DELIVERABLE

Knowledge Content Carrier Architecture
METOKIS D10

Task Number - Name	**Task 2.1**
Issuing Date	**15th September, 2004**
Distribution	**METOKIS internal, P.O., Final version to be shared with aceMedia**
Document Web Link	
File name	**d10_metokis_kcca_final.doc**
Author/Editor	**Sunil Goyal, Wernher Behrendt, Rupert Westenthaler**
QA Mentor	**W. Maass (UNISG)**
Lead/Contact Person	**Lead: Sunil Goyal, Contact: wernher.behrendt@salzburgresearch.at**

Change History

Version	Status	Comments / Changes
2.4	**final**	Signed off for delivery to project officer on 15th September, 2004
2.3	draft	Significant enhancement of KCO Model , added glossary
2.2	draft	Added KCCA minimal specification and new KCO Section included, August 2004
1.8	**draft**	Significant enhancements to KCCA, August 2004
1.5	wip	initial version + partner input where specified, 31st July, 2004
1.0	**wip**	**Initial version, communicated to partners 18th July 2004**
0.1 -0.9	**wip**	**Presentations of work in progress to partners in April and June 2004**

Table of Content

1 Executive Summary

This document gives an overview of the design issues concerning the Knowledge Content Carrier Architecture (KCCA) in METOKIS, including the definition of Knowledge Content Objects (KCO) and the specification of a layered knowledge content transfer protocol (KCTP).

Chapter 3 provides a state of the art report of existing semantic middleware systems.

Chapter 4 introduces several content and domain standards that are of interest in relation to the three METOKIS Application Scenarios (Educational Content Production, Clinical Trials and Senior Executives in Retail Sector).

Chapter 5 summarises the relevant knowledge representation standards and modeling techniques and upper level ontologies.

Chapter 6 describes Knowledge Content Objects (KCO) and the KCO modeling scenarios related to the three application scenarios.

Chapter 7 and Chapter 8 describe Knowledge Content Carrier Architecture (KCCA) and Knowledge Content Transfer Protocol (KCTP) which enable sharing of Knowledge Content Objects amongst disparate systems.

Chapter 9 provides use case scenarios showing the working of KCCA within the three application scenarios.

2 Objectives

This deliverable contains the initial specification of the METOKIS knowledge content carrier architecture. The architecture comprises three elements:

- KCTP (Knowledge Content Transfer Protocol) - a layered protocol for distributed content storage, aggregation and delivery systems that can cope with a combination of knowledge-based information and media based information.
- KCO (Knowledge Content Object) - an object model for a "container structure" and its semantics. The KCO is intended to be interpreted by any software dealing with such knowledge and media resources and provides an open content standard based on existing, partial standards.
- KCCA (Knowledge Content Carrier Architecture) - An architecture that proposes typical components of content value chain systems and shows how they can be put to use.

At the heart of our investigation will be the operators needed by such a knowledge content environment. Grouping the operators sensibly will lead to the definition of components that fulfil typical tasks as a media repository management system.

Further details about the objectives of the METOKIS Architecture can be found in deliverable D02 (Quality Assurance Plan) and in the technical annex, e.g. on pages 8 - 14 for the overall reasoning behind proposing this type of middleware.

3 State of the Art - Semantic Middleware Systems

3.1 Introduction

This section gives an overview of the state of the art semantic middleware systems. It provides an overview of the various middleware systems that have been developed in the past within the context of semantic web systems.

Rather than providing a list of every known system, the approach has been to look into the semantic middleware and content management systems from the perspective of particular functionalities that each system provides. The semantic middleware systems focus on one or more of the below functionalities or modules:

- Database Integration
- Ontology & Reasoning
- Services & Workflow support
- Middleware Component Management
- Communication (HTTP, Message Oriented Middleware (MoM), SOAP)
- Presentation

The first part presents a number of semantic middleware systems, analysising and comparing them with respect to the above parameters. The second part of this chapter also discusses various techniques, methodologies and frameworks employed by semantic middleware systems for database integration and services support.

3.2 SCAM (Standardized Content Archiving Management) Framework

SCAM [SCAM01] is a content archive management system, developed under the supervision of the KMR group, in cooperation with the Swedish National Agency for Education. The SCAM Framework provides storage & metadata access for a variety of applications that can be built on top of it. The SCAM framework uses RDF for metadata description. It consists of the following core parts:

Repository: Specific EJBs provide functionality for administration of user groups, supporting access and query of metadata, organization of metadata as well as storage of content, including a limited WebDAV [WebDAV99] [WebDAV01] functionality for accessing content.

Middleware: The middleware in SCAM acts as a container to the specific EJB's and provides HTTP access to the repository.

For management of metadata, the SCAM Framework defines *SCAM Records* & *SCAM Contexts*. SCAM Record is defined as an anonymous closure of a RDF graph computed from a given non-anonymous RDF resource. The closure is computed by following properties in the direction of subject to object until a non-anonymous RDF resource or RDF literal is found. Within a RDF Graph, a SCAM record is uniquely identified by the URI reference of the non-anonymous RDF resource from which the anonymous closure is computed. SCAM records coincide with URIQA's [URIQA] concise bounded description (CBD) except for the fact that CBDs do not include reifications.

SCAM contexts are defined as aggregations of SCAM records. The aggregation depends on the particular application. SCAM contexts enable the SCAM system to define administration features (access policies) etc. at the SCAM Context level. The system allows multiple SCAM records for the same resource via a marker to keep properties such as author, access rights or metametadata about the resource.

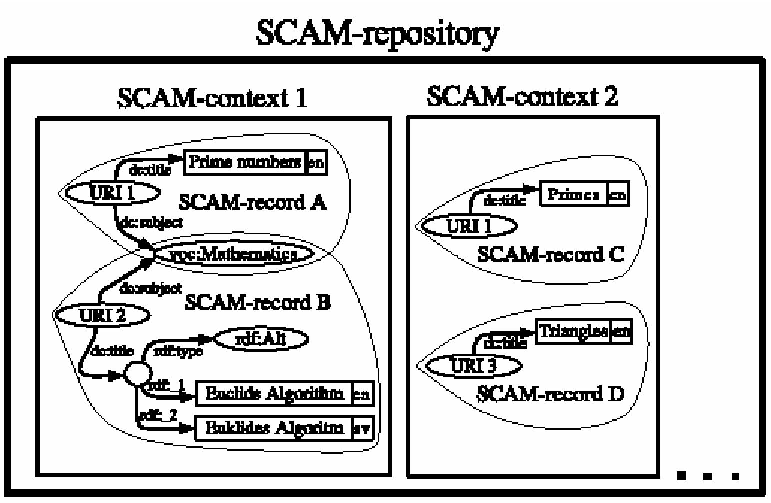


Figure 1: SCAM Records and SCAM Contexts that exist within the SCAM- Repository.

The SCAM Framework is implemented on a J2EE [J2EE] platform using JBoss [JBOSS] as the application server. It uses JENA2 [JENA] as triple store with a relational database as the backend.

The SCAM framework also uses a form of *Access Control Lists* at SCAM Record level to define access rights (read, write, lock resources). It uses RDQL and QEL [QEL] for querying RDF resources. Similar to the support of ACLs the SCAM repository also provides *support for meta-metadata* (creation date etc.) at the level of SCAM records. It uses SHAME (Standardized Hyper Adaptable Metadata Editor) [SHAME] at the presentation end.

3.3 Generic Interoperability Framework (GINF)

GINF proposes a universal interface allowing the generic representation of protocols, languages and data. The framework is based on a mediation infrastructure for interoperability in digital information systems. GINF is currently a working proposal by Sergey Melnik et al [GINF99] within the Digital Library Project at Department of Computer Science, Stanford University, USA.

GINF proposes a layered architecture providing a generic representation of protocols (HTTP vs IIOP), data (object oriented vs relational) and languages (data manipulation SQL vs OQL). It uses RDF as a graph structure for mapping messages containing data, protocol and language information amongst components.

Data: this component uses RDF models serialized as XML for representing data models specific to application domains. It uses the notion of RDF to define resources using URI's and defining appropriate metadata (e.g. Dublin Core) etc.

Protocol: In addition to data, the GINF framework represents the message communication of a simple request-response protocol in RDF.

Languages: The framework requires all objects including those "hidden" in queries to be uniquely identifiable. It requires query languages to be transparent for querying resources because most of the time, the identity of objects is encapsulated within data manipulation languages. e.g. SQL queries like "Select title, name from book" are represented in RDF by defining an equivalent RDF schema for SQL.

Information containing data, protocol and language is then intermixed within an RDF graph along with other layers containing information about sessions, load balancing etc. as needed depending on a particular system.

Mediation Infrastructure: GINF uses mediation via a canonical wrapper approach for integration with heterogeneous data sources. Canonical Wrappers provide uniform interface to components using generic representation of data, protocol and languages, but deals only with syntactic heterogeneity.

Mediators then use these canonical wrappers to do semantic translation amongst multiple domains between canonical wrappers, mediators and clients.

Layered Architecture: GINF uses namespaces to divide the application logic into modules that implement semantically coherent functions. A layered architecture is built using "Processing entities" where each processing entity supports a specific vocabulary (i.e. a certain set of concepts defined by a particular namespace). Processing Entities can create, modify and exchange RDF models. Any information in the RDF model which is not understood by a given Processing Entity is considered as invisible to the Processing Entity. The figure below shows multiple Processing Entities working on a particular RDF model.

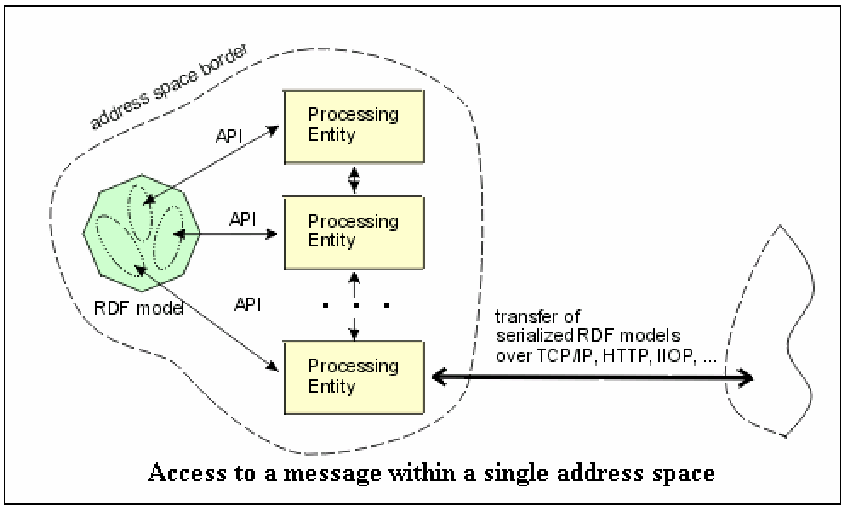

address space border
API
Processing Entity
API
Processing Entity
RDF model
API
Processing Entity
transfer of serialized RDF models over TCP/IP, HTTP, IIOP, ...
Access to a message within a single address space

Figure 2: An RDF model consisting of particular vocabularies is interpreted by multiple Processing Entities where each Processing Entity understands a particular vocabulary.

3.4 SEAL - SEMANTIC PORTAL

The SEAL (Semantic Portal) system provides infrastructure support for building semantic portals. The SEAL (SEAL I [SEAL01] & SEAL II [SEAL02]) system has been built by AIFB Karlsruhe. The overall system consists of a knowledge repository along with a reasoning engine provided by the OntoBroker [Decker01] system. The Seal Architecture distinguishes between three types of agents: Software Agents query the system for a collection of facts in RDF; General Users who access the content via the web and finally the Community Users who can both, access and modify the content.

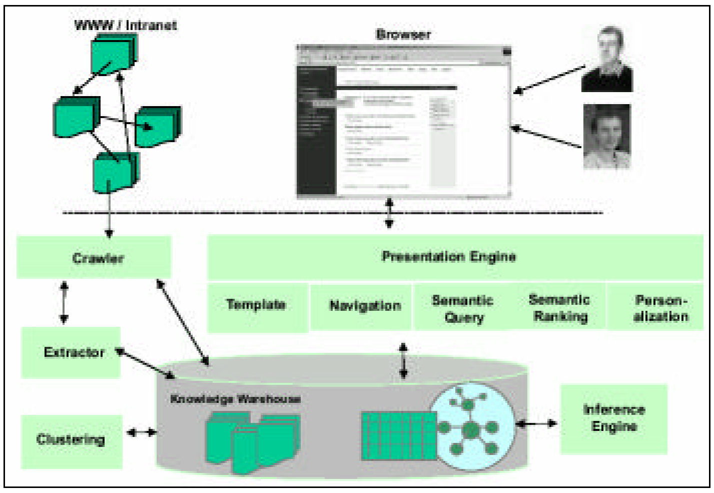


Figure 3: Overall System Architecture of SEAL II

The core modules of SEAL-II are:

OntoBroker: The Ontobroker [Decker01] system is a deductive, object-oriented database system providing compilers for different languages to describe ontologies, rules and facts. It is used as an inference engine within the architecture.

Knowledge warehouse: It serves as repository for data represented in the form of F-Logic statements. It hosts both the ontology and the data instances. It does not distinguish between schema and non-schema data.

Extractor: The extractor recognizes natural language texts and annotates them automatically to contexts. It uses the Porter Stemming Algorithm [Porter80] for word stem generation and then uses a lexicon for mapping words into concepts.

Ontology focused Crawler: It judges relevancy of documents, taking into account the ontology from the knowledge warehouse.

Ontology based Clustering: This component clusters unstructured documents into particular groups taking into account the knowledge provided by the ontology.

Presentation:
- *Template module:* The template module generates HTML forms for concepts that a user may instantiate. The data is used by the navigation module to produce the corresponding web pages.
- *Navigation module:* It provides complex graph-based semantic hyper-linking based on ontological relations between concepts in the domain.
- *Query module:* The query module provides an interface for the F-Logic based query interface of Ontobroker. Query events can be triggered when particular nodes in the tree are navigated.

Semantic Ranking & Personalization:
- *Ontology lexicon:* It is used for building adaptive web sites and its current implementation distinguishes between English and German localisations.

SEAL also provides components for semantic ranking of resources as well as semantic personalization within a portal using semantic bookmarks and log files.

3.5 SWIM - Semantic Web Integration Middleware

The SWIM framework [SWIM01] is a Semantic Web integration middleware for integrating heterogeneous data sources in the semantic web world and supporting reasoning services on top of it. The SWIM Middleware has been built by ICS FORTH in Greece. The key focus in SWIM is to support mappings between different data models, as well as, reformulation and optimization of queries expressed against mediation schemas and views. SWIM is based upon Datalog (see e.h. [Abiteboul95]).

The figure below describes the SWIM architecture. At the lowest layer there are multiple heterogeneous data sources such as Relational, XML. The SWIM middleware works on particular domain or application ontology expressed in RDFS and uses mapping rules to translate between source data models and the virtual RDF models expressed in RDF. It can then define personalized views on RDF using RDF View Language (RVL) [RVL03].

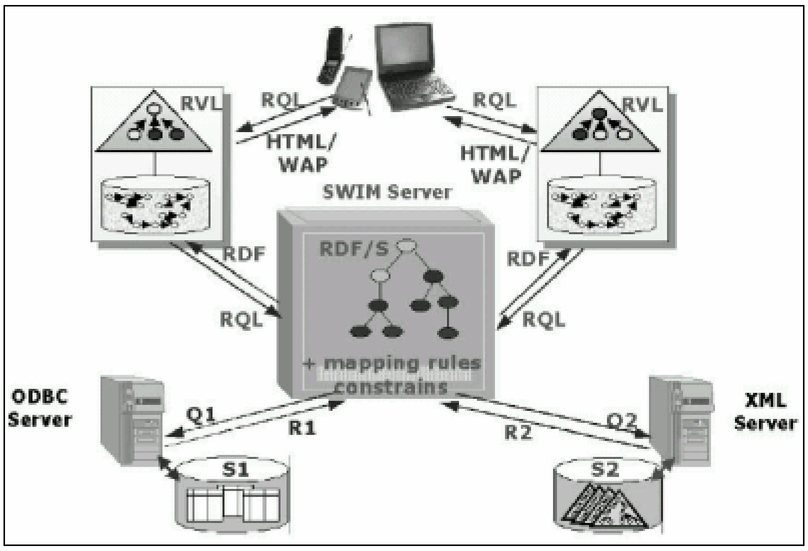


Figure 4: SWIM Architecture

The SWIM framework consists of the following key components:

Mappings: The SWIM middleware enables user to specify XML to RDF and Relational Database to RDF mappings. These mappings are expressed using Datalog. It also verifies that the mappings conform to the ontologies.

Query Mediation: It defines an internal framework based on Datalog-like rules to express RDF/S semantics and uses this framework to translate both queries and mappings into the internal framework.

Query Reformulation & Optimization: Based on the mappings and the query mediation framework, the SWIM middleware reformulates RQL queries into equivalent XML or Relational Database (SQL) queries. It also does optimization on the reformulated queries.

View Specification: It enables specifying personalized views over the mediator schema in RDF and RQL queries on the views.

3.6 KAON - Karlsruhe Ontology and Semantic Web Framework

KAON [KAON03] [KAON04] is a framework based on RDF which provides tools for storage, discovery, management and presentation of ontologies and metadata. KAON is an open-source project and has been developed jointly by Institute AIFB and the Research Center for Information Technologies (FZI Karlsruhe, Germany).

The KAON middleware provides an abstraction for ontology and data access and also provides component management functionalities. The Client Applications either connect directly to the KAON Middleware by the KAON API to access ontologies and knowledge bases or directly access the storage data via the RDF API. An External Services Layer provides the KAON middleware access to databases, inference engines etc.

There are five key functional layers (as shown in the figure below) within the KAON Framework:

Connector Layer: Provides APIs to connect to the KAON server both locally as well as remotely via a network protocol.

Security Layer: It provides interceptors for authorized access and provides logging facilities.

Management Layer: This layer provides the basic components of an Application Service such as Transactional Management which ensures the ACID transactional properties.

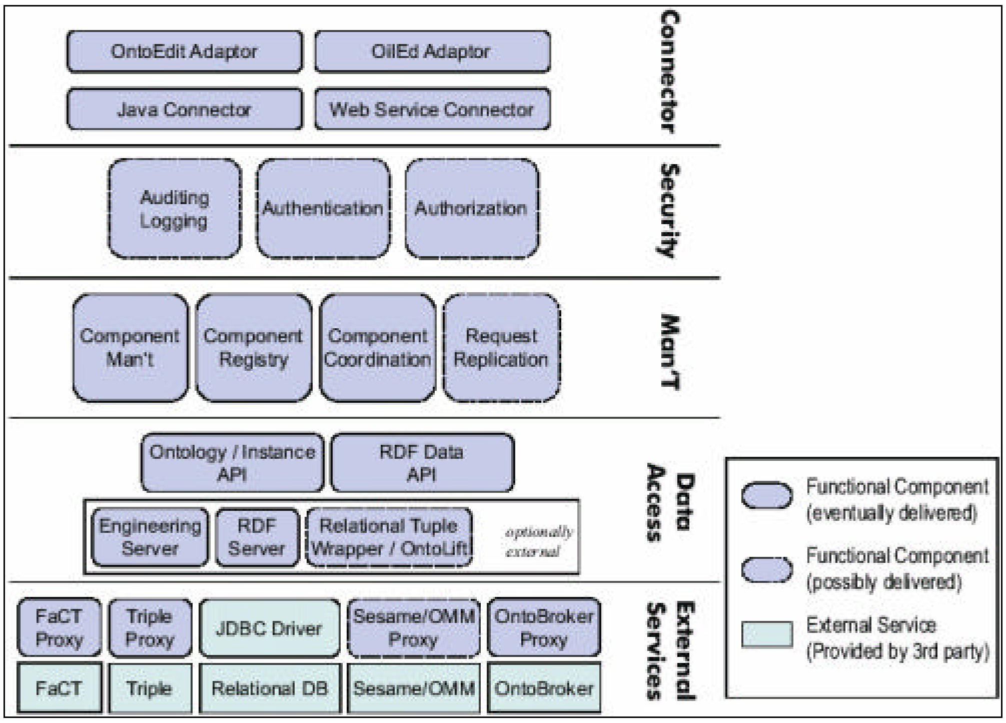


Figure 5: The functional layers of the KAON Architecture

Data Access Layer: This layer provides functionality for accessing and updating data by multiple users. It provides access to both the ontologies as well as the RDF data stored within the databases. The RDF API enables access to data stored in RDF Data Models and the KAON API provides an API to access and update ontologies.

External Services: This layer consists of external services which enables KAON server via using proxy components to manage external resources such as databases etc. The persistence the RDF models is managed within this layer. It also provides access to Reasoning Engines such as FACT [FACT98] etc.

3.7 MIKSI - Semantic & Service Oriented Integration Platform

MIKSI [MIKSI04] provides a semantic middleware architecture based on Service Oriented Architectures (SOA) using web services technology. The MIKSI platform has been developed by NIWA Web Solutions Vienna, Austria. The MIKSI platform extends the current Web Service Architectures and uses primarily RDF(S) for providing description of resources and messages. The key components of the MIKSI system are:

Semantic Data Model: The MIKSI platform uses RDFS as the data model for describing application domain ontologies and uses persistent model of the JENA RDF Framework for data storage. The persistent data models are then available within the J2EE EJB components.

MIKSI Service Oriented Architecture (SOA): The MISKI SOA is based on architectural standards such as SOAP and WSDL. Most of the services are atomic services performing particular functionality. It uses Business Process Execution Language for Web Services (BPEL4WS) [BPEL01] from IBM for defining composite business processes and their workflows.

MIKIS Architectural Components: It provides components for system management tasks, user authentication etc. There are three components:

- *MIKSI Parametric Service Invocation (Invoker):* MIKSI processes are not directly callable by the client but are accessible via parametric SOAP calls which are used by user session processes to determine which business process is to be invoked.

- *MIKSI Atomic Services:* These consist of functional atomic components described in WSDL and accessible via SOAP message exchange.

- *MIKSI User Session Processes:* this is a stateful BPEL process that manages user sessions.

MIKSI Messages: MIKSI uses SOAP as a transport layer but uses RDF serialized messages (MIKSI Messages) for communication. MIKSI messages conform to a lightweight MIKSI Message ontology. Valid instances of MIKSI SOAP Messages are generated by the client or server, serialized into plain text, packed into a SOAP Message as simple strings and is sent across. The actual light weight ontology for messages is not available publicly.

3.8 Semantic Message Oriented Middleware

Message Oriented Middleware (MoM) provides a loosely coupled asynchronous communication model between distributed applications. The current publish/subscribe mechanisms use keywords to match advertisements with subscriptions. Semantic Message Oriented Middleware [SMOM] provides the capability for semantic description and supports semantic matching.

The central middleware layer consists of two layers: Semantic Broker layer and the JMS [JMS] (J2EE) provider layer. The Semantic Broker acts as an interface between the JMS provider and the application clients and handles all publish/subscribe requests of the clients. It also organizes and maintains the topics of the JMS provider, matches and maps the clients' topic descriptions with the JMS topics, and generates and receives the JMS messages for the JMS provider. The delivery of the messages is done by the JMS provider.

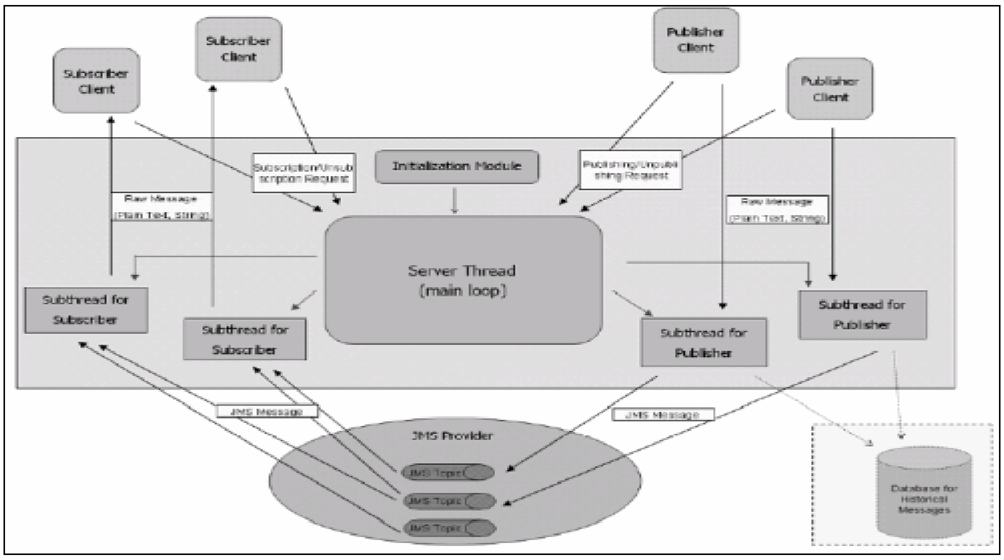


Figure 6: Semantic Message Oriented Middleware

In a publish/subscribe scenario, a client submits a DAML topic description as an advertisement or subscription to the semantic broker. The semantic broker searches existing topic descriptions in the system for matches, and then maps the advertisement or subscription of the current client to the JMS topics.

The knowledge base in Semantic MoM consists of topics domain consisting of DAML+OIL [DAML+OIL] ontologies, concept descriptions of subscribers, and instance descriptions of publishers. The DAML+OIL ontologies contain the definitions of concepts and roles of the topic knowledge domain, as well as the hierarchy structure and relationships between those concepts and roles. A DAML+OIL reasoning engine does the inference and the necessary matching.

It also provides secure communication between the semantic broker and client applications. It uses Secure Socket Layer (SSL) communication for the authentication of semantic broker and communication encryption. It also provides username and user information management for user privilege control.

3.9 Comparison of Semantic Middleware Systems

In this section we discuss and compare the Semantic Middleware Systems presented previously. Each of the current existing semantic middleware system is geared towards the domain in which it is currently being used. Some systems are strong in database integration and others are more geared towards reasoning or on a services based framework. The table (Table 1) below provides a comparison of the various systems.

	SCAM	GINF *	SEAL	SWIM	KAON	MIKSI
Data Model	RDF/S	RDF/S serialized as XML	RDF, F-Logic statements	Datalog	RDF/S	RDF/S
Database Integration	No	X[1]	Yes	Yes (via RVL)	Yes (via OntoMat-Reverse)	No
Query Languages	RDQL, QEL	Query with RDF Schema	F-Logic Query	RQL	RQL, QEL, RDF API, KAON API	RDQL
Ontology Support	Metadata Schemes	X	OIL, F-Logic statements	RDF/S, Datalog	RDF/S	RDF/S, OWL
Reasoning Engine	JENA	X	OntoBroker	Datalog	FaCT, OntoBroker, Triple	JENA (RDF/S,OWL)
Database	JENA (RDF Repository)	X	Relational, Documents,	Relational, XML, RDF	Relational, RDF, XML	RDF using Relational DB (JENA)
Middleware Framework	J2EE	X	X	No	J2EE	J2EE
Services Framework	No	No	No	No	SOAP(WSDL) Connector	SOAP (WSDL)
Workflow Engine	Limited Workflow	No	Limited Workflow	No	No	BPEL
Presentation Engine	SHAME, JSP	No	Semantic presentation (navigation, template, query)	No	KAON - Portal	X
Rights Access	Restricted ACL using EJB	X	X	No	JAAS (Java Authentication and Authorization Service)	EJB Components
Granularity	SCAM Record Level	RDF/S	F-Logic Statements	Datalog, RDF/S	RDF	RDF/S
Communication	HTTP,EJB, limited WebDAV,	X	HTTP	HTTP	HTTP, SOAP, RMI, Messaging	SOAP
Text Extraction	No	No	Yes	No	Yes (External module TextToOnto)	No
Key Contribution	SCAM Record and SCAM Context	Mediation Infrastructure for interoperability	Semantic Portals	Database Integration	Middleware Component Management	Service Oriented Architecture

X[1] indicates that the specific middleware does not specify the functionality in question.
* GINF is only a working proposal. No implementation currently exists.

Table 1: Comparison of Semantic Middleware Frameworks.

Semantic MoM has been excluded from the above matrix, as this acts as a specific component within a middleware and isn't a mandatory requirement for a semantic middleware system.

KAON looks to be the most mature middleware system in terms of providing and integrating multiple middleware components. It supports the ACID (Atomicity, Consistency, Isolation, and Durability) tests for ontology updations; supports localization, internationalization, authorization, logging support. But at the same time it lacks some other core functionalities like support for workflow engines, task execution etc.

Some of the above systems mainly work on structured data (with well defined RDF Schemas), while systems like SEAL, KAON provide support for document extraction, clustering, classification but they lack a sound framework for management of content objects (multimedia creation, storage, management, delivery, lifecycle management).

Following are some of the key requirements, which we believe are important for building a semantic multimedia content management platform:

- **Accessibility**: External systems should be able to very easily use middleware components through standard web protocols. Stress should be on dynamic integration via schema matching rather than just providing yet another API.
- **System self-description**: this is the ability to query a system in order to find out what it supports. Most of the functionalities within a middleware system are either locked in configuration files or are not defined explicitly, at all. Increasing the self-descriptive features of systems will increase the ability of external systems to know what a given system supports and to change its behavior accordingly.
- **Database Integration**: Provides support for integrating heterogeneous databases and query languages.
- **Reasoning**: Support for reasoning engines including support for ontologies, rules and facts is a necessary component for any semantic web application.
- **Security**: Authorised Access
- **Task Execution**: Support for execution of task specifications, rather than executing bespoke application code.
- **Multimedia Content Management:** This includes management of multimedia, including creation, storage, delivery, lifecycle management, rights management, trading and sharing of knowledge assets.

3.10 Frameworks for Service/Workflow Support

3.10.1 Web Services (Web Service Description Language WSDL)

WSDL [WSDL01] is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

Types: a container for data type definitions using some type system (such as XSD).

Message: an abstract, typed definition of the data being communicated.

Operation: an abstract description of an action supported by the service.

Port Type: an abstract set of operations supported by one or more endpoints.

Binding: a concrete protocol and data format specification for a particular port type.

Port: a single endpoint defined as a combination of a binding and a network address.

Service: a collection of related endpoints.

Specific binding extensions for protocols and message formats can be specified in WSDL. The current WSDL specification (WSDL 1.1) specifies bindings for SOAP 1.1, HTTP GET / POST and MIME.

3.10.2 OWL-S (DAML-S)

OWL-S [OWLS01] provides a semantic markup of Web Services which enables users and software agents to discover, invoke, compose, and monitor Web Services. The OWL-S initiative originated from the DARPA Agent Markup Language programme and the services ontology was previously known as DAML-S.

OWL-S defines an upper ontology for services which enables machines to process semantic descriptions. Service refers to a general concept in OWL-S and acts as a reference to each distinct published service. A Service is described by three properties: *Service Profile* - Service Profile refers to the semantic description of what a particular service a user or a software agent provides; *Service Model* - Service Model provides a semantic description of how the service works; and *Service Grounding* - the Grounding describes how to access that service.

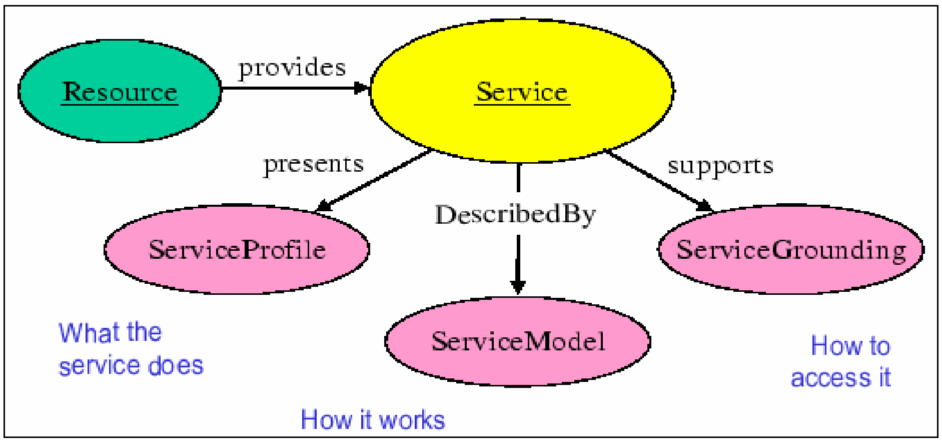


Figure 7: Top level of Service Ontology in OWL-S

Service Profile
The Service Profile provides a high level description of the Service consisting of:

- *Service Name, Contacts & Description*: Human readable information of a service
- *Functionality Description*: Provides description of the service, the necessary parameters, input, output, preconditions and effects.
- *Profile Attributes*: Refers to the quality guarantees that are provided by the service, classification of the service, etc.
- *Service Parameters*: Service parameters names and values.
- *Service Category*: Defines the category of the service based on some classification.

Service Model
In OWL-S a service can be viewed as equivalent to a Process. A process in OWL-S can either generate & return some new information or it can produce a change in the world. The latter transition is described by the preconditions and effects of the process. A process has the following properties associated with it:

- *Participants:* the Agents that participate in the Process (e.g. client, server etc.).
- *Inputs and Outputs:* these specify the data transformation of the Process.

- *Preconditions & Results:* refer to any pre-condition that has to be true in order for the process to be executed.
- *Conditioning Outputs & Effects:* Outputs describe the resulting information and effects describe the particular change in conditions in the world as a result of executing the Process.

There are two kinds of Processes in OWL-S: Atomic Processes & Composite Processes. Atomic Processes are single step processes that are directly invokable and which have no sub-processes. Composite Processes are decomposable into composite and non-composite processes. OWL-S provides the necessary control constructs such as Sequence and If-Then-Else etc for modeling the necessary data flow between Composite Processes. OWL-S also defines Simple Processes which like Atomic Processes are single step services but are not associated with a grounding and hence are abstract and non-invokable.

Service Grounding
Service grounding acts as a mapping from an abstract to a concrete specification of a service. It specifies the details of how to access the service - details having mainly to do with protocol and message formats, serialization, transport and addressing. OWL-S currently supports binding using WSDL (Web Services Description Language). AN OWL-S atomic process corresponds to a WSDL operation and the set of input and output of an OWL-S atomic process corresponds to WSDL's concept of a message. The types (OWL classes) of the inputs and outputs of an OWL-S atomic process correspond to WSDL's extensible notion of abstract type.

3.10.3 WSMO (Web Services Modelling Ontology)

WSMO [WSMO01] (Web Services Modelling Ontology) defines an ontology for modeling Web Services. WSMO is currently being developed by the DIP Integrated Project [DIP04] consisting of a consortium of 50 partners from industry and academia. WSMO aims to provide a semi automated solution for integrating Enterprise Application systems.

WSMO is a part of modeling environment called WSMF [WSML] (Web Service Modeling Framework) and WSMX (Web Services Execution Environment) [WSMX]. The core components of the framework include:

- Ontologies: Ontologies provide the formal semantics to the information.
- Goals: Specify the objectives that a client may have when consulting a Web Service.
- Mediators: Used as connectors to provide interoperability facilities among the rest of components.
- Web Services: Represent the functional part

Ontologies: the WSMO Ontology consists of the following components:

Non-functional Properties: These consist of properties (such as Dublin Core Metadata set), versioning information and specific information to represent Quality of Service (QoS) for a Web Service.

Used Mediators: Mediators (ooMediators) enable modular ontology design by enabling importing of ontologies defined elsewhere.

Axioms: these are considered to be logic expressions enriched by some extra-logic information.

Concepts: these provide an abstract view over real-world and artificial artifacts within a domain. Within WSMO, concepts provide definition of super concepts, attributes, methods their domain, range and parameters.

Relations: Relations model interdependencies between several concepts and their instances.

Instances: Instances model particular instances that exist for a concept.

Goals: Goals in WSMO consist of non-functional properties, used mediators, post-conditions and effects. They don't include pre-conditions.

Mediators: There are four kinds of mediators (classified in two groups) that exist in WSMO:

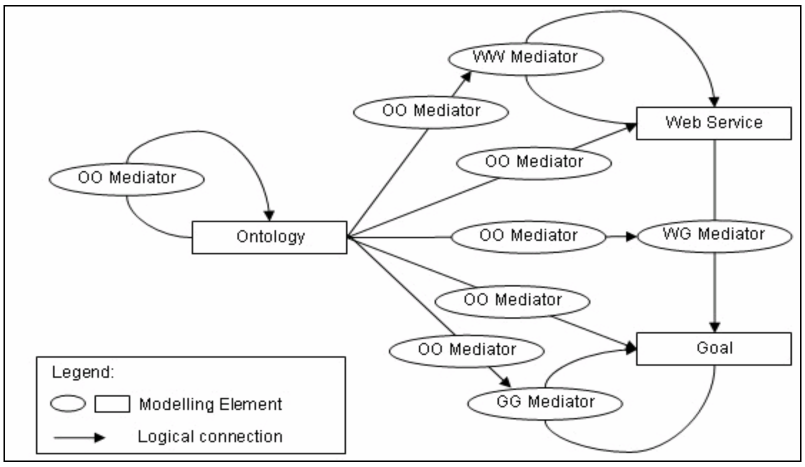


Figure 8: Mediators in WSMO

Refiners: ggMediators - mediators that link two goals. This link represents the refinement of the source goal into the target goal. *ooMediators* - mediators that import ontologies and resolve possible representation mismatches between ontologies.

Bridges: wgMediators - mediators that link web service to goals. They explicitly may state the difference (reduction) between the two components and map different vocabularies (through the use of ooMediators). *wwMediators* are the mediators that link two Web Services.

Web Services: Web Services description in WSMO consists of non functional properties, used mediators, capability and interfaces. Capability refers to service capability and defines pre-conditions, post-conditions, assumptions, effects, non functional properties and used mediators for a service and enables a system to have self-contained web service descriptions. An interface describes how the functionality of the service can be achieved. *Choreography* decomposes a capability in terms of interaction with the service (service user's view) and *Orchestration* decomposes a capability in terms of functionality required from other services (other service providers' view).

3.10.4 IRS II (Internet Reasoning Service)

The IRS [Crubezy02] [Motta03] framework supports the publication, location, composition and execution of heterogeneous web services, augmented with semantic descriptions of their functionalities. IRS is based on the UPML framework [Fensel99]. The UPML framework distinguishes between following components:

Domain Models: describe the application domain.

Task Models: provide generic task description, input and output types, pre-conditions and goals.

Problem Solving Methods (PSMS): These provide abstract, implementation independent descriptions of reasoning processes which can be applied to solve tasks in specific domains.

Bridges: These provide mappings between different model components of an application.

Each of the above component is specified via an ontology. The task method distinction in the above framework enables explicit separation between service types and service providers. The IRS framework uses the above semantic description model to realize semantic web services infrastructure. The IRS architecture consists of IRS Server, IRS Publisher and IRS Client which communicate via the SOAP Protocol.

IRS Server stores the knowledge level descriptions (represented internally as OCML) using the UPML framework of tasks, PSM's and domain models. Web Service mediation and composition are supported by task preconditions and goal expressions. The integration of semantic descriptions with Web Services is done via SOAP bindings. PSM's in IRS enable the system to do method to task mapping and to build mediation services.

IRS Publisher links the web services to their semantic descriptions and it also provides a way to generate a set of wrappers which can turn standalone java or lisp programs into a web service described by PSM.

IRS Client can invoke web services via APIs which are task centric. An IRS-II user simply asks for a task to be achieved and the IRS-II broker locates an appropriate PSM and then invokes the corresponding web service.

3.10.5 BPEL4WS (Business Process Execution Language for Web Services)

The Business Process Execution Language for Web Services is an industry initiative led by BEA Systems, IBM, Microsoft, SAP AG, Siebel Systems to drive and ensure interoperability for the description and communication of business processes based on web services. It has been submitted to the OASIS e-business standards body in April 2003. It combines the earlier efforts of WSFL from IBM and XLANG from Microsoft and provides a mechanism for Web service composition.

BPEL4WS builds on the core Web service architecture by layering on top of XML Schema, WSDL, and XPATH. Processes in BPEL4WS are based on Web Service interfaces exclusively.

Business processes within BPEL can be described in two ways: Executable Processes and Abstract Processes. Executable Processes help define a new Web Service which as a composition of existing Web Services and represents invokable services. Abstract Processes describe business protocols specifying the potential sequencing of messages exchanged by one particular partner with its other partners to achieve a business goal. These represent the non-deterministic behavior of a business protocol and hide the potentially complex details of an internal Executable Process.

BPEL4WS was released along with two others specifications: WS-Coordination and WS-Transaction. WS-Coordination describes how services can make use of pre-defined coordination contexts to subscribe to a particular role in a collaborative activity. WS-Transaction provides a framework for incorporating transactional semantics into coordinated activities by using WS-Coordination.

3.11 Other Semantic Web Frameworks/Systems

3.11.1 MAFRA - A Mapping Framework for Distributed Ontologies

URL: http://www.fzi.de/wim/eng/publikationen.php?id=810 , http://sourceforge.net/projects/hmafra
Project: Ontologging
Type: Open Source
Research Groups: FZI University of Karlsruhe, Germany; ISEP Instituto Superior de Engenharia, Instituto Politecnico do Porto, Portugal

MAFRA [Maedche02] provides a conceptual framework into the distributed ontology mapping process covering the ontology mapping lifecycle. Taking into account the decentralized nature of the web, it takes a distributed approach for schema mediation as compared to centralized mediation (global as view or local as view) approaches. The conceptual architecture consists of five key modules: Lift & Normalization (syntax and language heterogeneities); establishing Similarity between source & target ontologies; making Semantic Bridges between the source and the target ontologies; execution (static or dynamic) for generation of target instances based on semantic bridges and finally post-processing to check and improve the transformation process. It provides support for ontology evolution and consensus building for maintaining the semantic bridges and provides a GUI tool for supporting the building of such mappings.

Semantic Bridge Ontology (SBO) provides ontology for building semantic bridges between different ontologies. SBO is expressed in DAML+OIL. The execution engine takes the mappings to generate the necessary instances.

3.11.2 QEL (RDF Query Exchange Language)

URL: http://edutella.jxta.org/
Project: Edutella
Type: Open Source
Research Groups: Learning Lab Lower Saxony, University of Hannover, Germany; Database Group, Stanford University, USA; DFKI GmbH, Kaiserslautern, Germany; Centre for user oriented IT Design, Royal Institute of Technology, Stockholm, Sweden; Department of Information Science, Uppsala University, Sweden.

QEL (Query Exchange Language) [QEL] has been developed as part of the Edutella project. It is used to distribute queries to various RDF repositories, where the query is transformed to the repository query language (e.g. SQL, RQL).It abstracts from various RDF database storage languages (SQL) and user level query languages (RQL, TRIPLE) and provides syntax and semantics for querying against distributed peer to peer repositories holding RDF metadata.

QEL differs from many other query languages for RDF data in that it is based on relational calculus, the basis for datalog semantics, rather than relational algebra (the basis for SQL). QEL contains a number of built-in datalog predicates that are used to query the RDF data. There are Matching predicates (creates new bindings of variables from RDF data sources) and Constraint predicates (constrain values that have been already found).QEL supports a limited form of negation which enables to restrict query results in semantic web data. QEL queries support Datalog like rules and also provide support for Outer-Joins on RDF databases. Results in QEL are returned in a tabular format, one row at a time which makes it possible to return results in independent batches.

QEL provides Datalog and RDF syntax bindings for the query language.

3.11.3 D2RQ & D2RMap

URL: http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rq/,
http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rmap/D2Rmap.htm
Project: D2RQ
Type: Open Source, LGPL
Research Groups: Chris Bizer, Institut für Produktion, Wirtschaftsinformatik und OR, Fachbereich Wirtschaftswissenschaft - Freie Universität Berlin, Germany.

D2RQ is a declarative language to describe mappings between relational database schema and OWL/RDFS ontologies. The mappings allow treating information in a non-RDF database as a virtual, read-only RDF graph. D2RQ has been implemented as a part of the JENA Framework. A D2RQ Graph treats one or more local relational databases as virtual read-only graphs and rewrites Jena API calls or RDQL queries into equivalent application specific SQL queries. The result set of the SQL queries is transformed back into RDF triples.

D2R Map provides a one way export of Relational Databases into RDF. D2R Map supports different Jena model implementations like the Kowari [KOWARI] Metastore (a pure graph based RDF database provided by Tucana Technologies).

3.11.4 URIQA (URI Query Agent)

URL: http://sw.nokia.com/uriqa/URIQA.html
Project: URIQA
Type: Unknown
Research Groups: Nokia

URIQA (URI Query Agent) [URIQA] is a model for knowledge discovery on the web. It introduces an extension to the present web architecture, which is used to indicate to a web server that it should resolve the specified URI in terms of knowledge about the resource denoted by that URI rather than in terms of a representation of the resource in question. Semantic Web Agents on the web can query web servers for resources by asking for the concise bounded description of the resource.

A *concise bounded description* (CBD) [CBD] of a resource (with a given URI) is a set of statements in RDF and can be characterised as follows:

- All statements where the subject of the statement denotes the resource in question; and

- Recursively, for all statements included in the description thus far, for all anonymous node objects, all statements where the subject of the statement denotes the anonymous resource in question; and

- Recursively, for all statements included in the description thus far, for all reifications of each statement, the concise bounded description of each reification.

The Concise Bounded Description has been updated to solve the "URI bloat" [Dawes04] problem. The inclusion of anonymous nodes (the second point above) now follows the below constraint:

Recursively, for all statements included in the description thus far, for all anonymous node objects, include a *inverse functional bounded description* of the anonymous resource as follows:
If there exists at least one statement having the anonymous resource as subject and where the predicate is an owl:InverseFunctionalProperty, then

- Include only those statements having the anonymous resource as subject and where the predicate is an owl:InverseFunctionalProperty; and
- If the object of such a statement is an anonymous node, include the inverse functional bounded description of that anonymous resource.

Else,

- Include all statements where the anonymous resource is the subject; and
- If the object of such a statement is an anonymous node, include the inverse functional bounded description of that anonymous resource.

URIQA extends the present web architecture by defining new HTTP methods: MGET (Returns statements contained in the CBD for a Resource), MPUT (Inserts statements contained in the CBD of the Resource) and MDELETE (Remove statements contained in the CBD of a Resource). It also provides a simple semantic web interface providing the above descriptions using HTTP GET and POST. The current implementation of URIQA is provided as part of RDF Gateway, a product by Intellidimension (http://www.intellidimension.com/). It is available free for use in non-commercial projects.

3.11.5 JOSEKI (JENA RDF Server)

URL: http://www.joseki.org
Project: Part of JENA RDF Toolkit
Type: Open Source, BSD License
Research Groups: HP Labs, Bristol, UK.

Joseki [JOSEKI] is a server for publishing RDF models on the web. The Models have URLs and these are accessible by querying using HTTP GET. Joseki provides a RDF WebAPI that enables extracting of RDF sub-graphs from the published RDF models. The operations supported by JOSEKI are:

1) GET (a simple HTTP GET to a model URL returns the whole model).

2) Query, for various query languages encoded either as part of the HTTP Get request or encoded in RDF and communicated via HTTP Post.

3) Update operations, Add & Remove operations enable adding and removing of statements from RDF models.

The Joseki core engine is independent of HTTP but it currently provides only HTTP binding. It enabled addition of new query languages and new operations without modifying the core system.

4 Content Standards & Domain Standards

This section gives an overview of various content & metadata standards related to the three application domains (Education, Senior Executives, Clinical Trials) in Metokis. The section provides an overview of standards existing in the Learning sector, Publishing, Clinical Trial, Rights Management and Multimedia.

4.1 Learning Standards

We look at the LOM, SCORM, EML and PALO standards.

4.1.1 Learning Object Metadata (LOM)

The LOM [LOM01] [LOM02] standard specifies a conceptual data schema that defines the structure of a metadata instance for a learning object. A learning object is defined as any entity - digital or non-digital that may be used for learning, education or training. The LOM standard defines a basic conceptual schema for defining learning objects.

It defines nine different categories:

- *General* category groups the general information that describes the learning object as a whole e.g. title, catalog, language, description, keywords etc.
- *Lifecycle* category groups the features related to the history and current state of this learning object and those who have affected this learning object during its evolution e.g. version, status, contributor etc.
- *Meta-Metadata* category groups information about the metadata instance itself describing how the metadata instance can be identified, who created this metadata instance, how, when, and with what references.
- *Technical* category groups the technical requirements and technical characteristics of the learning object e.g. size, location, operating system, browser, installation requirements etc.
- *Educational* category groups the educational and pedagogic characteristics of the learning object e.g. difficulty level, age group, interactivity, learning time, context etc.
- *Rights* category defines the intellectual property rights and conditions of use for the learning object e.g. cost, copyright etc.
- *Relation* category defines relationship between multiple learning objects if these exist e.g. kind of relationship – Dublin Core relations, isPartOf, isBasedOn, requires etc.
- *Annotation* category provides comments on the educational use of the learning object and provides information on when and by whom the comments were created. This allows authors to share their annotations or assessment of particular learning objects.
- *Classification* category describes this learning object in relation to a particular classification system or taxonomy.

Conclusion: The LOM standard despite its broad coverage is specifically geared towards the learning sector. Concepts like Lifecycle, Rights, Meta-Metadata, Annotation are generic for any kind of content and are independent from any domain. Secondly LOM standard is semantically poor as compared to ontologies as one cannot define semantically rich relations occurring within multimedia content. The LOM standard via using Dublin Core and its own schema defines the "multimedia" part of the content, but it lacks the richness provided by "proper" multimedia standards like MPEG7 & MPEG21.

4.2.2 Shared Content Object Reference Model (SCORM)

SCORM [SCORM04] defines a metadata standard and a runtime environment for metadata in the learning sector. The SCORM Content Model describes the SCORM components used to build a learning experience from learning resources and how the low-level sharable learning resources are aggregated into higher-level units of instruction.

The SCORM Content Model is made up of:

- *Assets*: Electronic representation of media, e.g. text, image, video, xml documents etc.

- *Shared Content Object (SCO)*: A collection of one or more Assets representing a learning resource. SCO communicates with a Learning Management System (LMS) using the IEEE ECMAScript Application Programming Interface for Content to Runtime Services Communication.

- *Content Organization:* A Content Organization is a map that represents the intended use of the content through structured units of instruction (Activities). Learning taxonomies with hierarchical level of Activities (course, chapter, module etc.) can also be represented.

The SCORM Content Model defines a packaging structure (via manifest files) and a metadata structure for all of the above. It defines additional application metadata profiles along with using the LOM metadata for learning objects. The model also defines sequencing and a presentation component for SCOs.

The SCORM Run Time Environment Model details the requirements for launching content objects, establishing communication between LMSs and SCOs, and managing the tracking information that can be communicated between SCOs and LMSs.

Conclusion: The SCORM model takes the shift from defining pure metadata models to also defining an environment where such content can be accessed and made use of. This leads to the benefits that the system comes as a packaged unit and the content once created, is ready for delivery. But it comes at the added cost of an infrastructure, which is geared towards pure SCROM objects and is closed to the external world. The SCORM model also lacks the semantic richness that enables interpretation of content.

4.2.3 Educative Modelling Language (EML)

"An EML is a semantic information model and binding, describing the content and process within a ‘unit of learning’ from a pedagogical perspective in order to support reuse and interoperability."

The key concept in EML ([EML1]) is a "unit of learning". A unit of learning describes the learning design, the resources and the services needed in order to achieve one or more interrelated learning objectives. A unit of study cannot be broken down to its component parts without losing its semantic and pragmatic meaning and its effectiveness towards the attainment of the learning objectives. There exists a variety of EMLs each having a different information structure of its own:

CDF - Course Description Format by Swiss Federal Institute of Technology (EPFL)
OUNL- EML by Open University of the Netherlands (OUNL)
LMML - Learning Material Markup Language by University of Passau, Germany (UP)
PALO - by UNED University, Spain
Targeteam - by Universität der Bundeswehr, München (UB)
TML - Tutorial Markup Language by ILRT, University of Bristol, UK (ILRT)

SCORM is not included as an EML language as it doesn't include any pedagogical semantics. EML can possibly be used as a technical framework into which the semantics of other standards can be integrated.

4.2.4 PALO

PALO ([PALO1] [PALO2]) is an Educative Modeling Language (EML) designed to provide a high level definition of educational courses. The model is based on a set of templates consisting of a set of elements provided by the PALO language. PALO content model revolves around creating a course-specific repository of semantically linked material.

It consists of the following layers:

- *Educational Content*: It consists of plain or formatted text, standard learning formats like IMS, from an external repository or is defined using a domain model. It provides a semantic relationship between educational content by defining interrelationship between *concept*, *examples*, *explanation* and *prerequisites*.

- *Activity and co-operative model:* Tasks in PALO are defined in learning context consisting of a set of components and associated actors. It assumes two key actors: student and teacher. It defines simple tasks, essays and questionnaire. An essay is a composition of heterogeneous tasks and questionnaires provide support for quizzing. Tasks cover not just the description of work to do but also aspects like available resources and tools needed to carry out the activity.

- *Structure:* This layer provides an explicit structure to the contents of information model by providing hierarchical decomposition. The hierarchical elements of PALO enable one to establish pedagogical dependencies between different parts of the environment. It consists of 3 parts:
  - Directory Schema: It helps define directories (consisting of objectives, credits, instructions, and requisites).
  - Module Schema: Module defines a minimal unit of study containing learning content.
  - Part Schema: A second level hierarchy similar to module but providing functionality to group tasks.

- *Sequencing and Scheduling:* It provides timing dependencies between different components (prerequisites), defining deadlines and schedules over components. e.g. A deadline can be associated to a document.

- *Management:* This contains information for managing the delivery format, managing location of tools & repositories as well as metadata (Dublin Core).

Conclusion: PALO language provides definition, organisation (structure) and scheduling of learning content for defining educational course modules. But it does not define an environment for actual use of content. Even though it provides primitives for modelling course modules, it does not deal with issues like rights management, multimedia content description (MPEG 7) and presentation.

4.3 Publishing Standards

4.3.1 Publishing Requirements for Industrial Metadata (PRISM)

PRISM [PRISM] defines an XML metadata vocabulary for syndicating, aggregating, post-processing and multi-purposing content. It is based on standards like DublinCore [DC01], NewsML [NEWSML], NITF (News Industry Text Format) [NITF], ICE (Information & Content Exchange) [ICE01], RSS [RSS01] [RSS02], XrML (extensible Rights Markup Language) [XRML01]. PRISM compliant applications generate metadata that can be processed by RDF processing applications.

The PRISM Working Group specifies the following functional elements:

- *General Purpose Elements:* These form the generic descriptive PRISM metadata including Dublin core properties (title, creator, description etc), basic categories etc.

- *Provenance:* These elements describe the supply chain for a resource to indicate what the source material for a resource was and through which organizations the resource has passed. It tracks distributor, ISSN number, issue name, source, publisher etc.

- *Time Stamps:* This provides metadata for major milestones in the life of a resource e.g. creation, modification, expiration, publication, release etc.

- *Subject Description:* This describes the subject matter (people, places, events, things etc.) referred or described by the resource.

- *Resource Relationships:* This provides metadata for describing relationships amongst resources e.g. containment, versioning, formats, translated version etc.

- *Rights and Permissions:* It provides the rights and permissions vocabulary.

- *Controlled Vocabulary:* Many elements in PRISM-approved or PRISM-extended namespaces take values that are intended to come from *controlled vocabularies*. Controlled vocabularies are lists of terms that are updated through a defined and managed procedure.

- *PRISM In-Line Markup:* Important information, such as dates and the names of people, places, and things, occurs in the text of an article. Some organizations prefer to mark that data in-line rather than create a large set of subject description elements. PRISM provides inline elements to mark such data.

Conclusion: PRISM provides a generic framework for describing content related to publishing sector. It provides extensive set of primitives for resource description, content tracking, rights & permissions and provides limited extensibility via controlled vocabularies. PRISM currently lacks a strong integration with multimedia description standards and its use within domain specific content.

4.3.2 NewsML

NewsML [NEWSML] has been developed by the IPTC, an international consortium of news publishers and vendors. It is designed to provide a media-independent, structural framework for multi-media news. NewsML is an XML based framework for providing representation of electronic news items, collections of such items, the relationships between them, and their associated metadata. It provides a framework both for the interchange of news as well as management of news items. NewsML is media neutral and doesn't make any assumption about media encodings (audio, text, video). The key XML elements provided by NewsML are:

- *Catalogs:* used to identify vocabularies as well as topics within the document

- *Topic Sets:* list topics, as defined by the IPTC Topic Types e.g. events, people

- *NewsEnvelope:* contains information about how the NewsML document is being used within a business workflow or contractual relationship between news provider and receiver

- *NewsItems:* a publishable, multimedia unit of news. A NewsItem contains one or more ContentItems, the raw components of the news article.

- *NewsManagement:* contains information about a NewsItem's type, history and status, as well as its relationship to other NewsItems.

- *News Components:* The NewsComponent is a container for news objects. It is used to identify the role of news objects in relation to one another and to ascribe meta-data to them. A news object may be a NewsEnvelope, NewsItem, NewsComponent or ContentItem.

- *ContentItem:* A news object that contains, or provides a pointer to, a data object that carries renderable content (such as text, images, video, audio etc) intended for presentation to humans.

- *Metadata:* NewsComponents may have different types of meta-data: administrative (e.g. file name), rights (e.g. copyright) and descriptive (e.g. language).

- *NewsLines:* expose aspects of the meta-data to humans.

4.4 Clinical Trial Standards

4.4.1 CDISC Protocol Elements Standards

The CDISC [CDISC] standard portrays the structure of the clinical trial's protocol, i.e., the elements that should be included in a protocol, their hierarchical relations, their recommended order and some of their characteristics as defined by their attributes. CDISC takes into consideration the entire life cycle of the protocol (during the design, the execution and the data analysis stages of the trial). Not all the elements suggested in this model are accredited already by HL7 (Health Level 7). A few additions and slight modifications to some of its sections might be done within the project.

CDISC - Clinical Data Interchange Standards Consortium (http://www.cdisc.org/) is an incorporated non-profit organization. It is an industry consortium with FDA liaison committed to the development of worldwide industry standards to support the electronic acquisition, exchange, submission and archiving of clinical trials data and metadata for medical and biopharmaceutical product development.

HL7 - Health Level 7 (http://www.hl7.org/) is a non-profit volunteer organization accredited by American National Standards Institute (ANSI) as a SDO (Standards Developing Organization). Mission of HL7: *"To provide standards for the exchange, management and integration of data that support clinical patient care and the management, delivery and evaluation of healthcare services. Specifically, to create flexible, cost effective approaches, standards, guidelines, methodologies, and related services for interoperability between healthcare information systems".*

4.4.2 CDISC Glossary Version 2

This terminology standard refers to the design, and the management of clinical trials. It contains all the special terminology that might be used in the trial's protocol, as provided in reference sources (e.g., ICH, FDA, ACT, HL7); or, as defined by the CDISC Glossary Group, or by the PR Group, both working in collaboration with the HL7.

It includes 2 parts: Terms and definitions [CDISC01] for clinical trials and Abbreviations and acronyms [CDISC02] related to the same.

4.4.3 The HL7 RIM (Reference Information Model) Standard

HL7 RIM [HL7RIM] is a static model of health and health care information as viewed within the scope of HL7 standards development activities. The RIM is a large pictorial representation of the clinical data (domains). It includes class and state-machine diagrams and is accompanied by use case models, interaction models, data type models, terminology models, and other types of models to provide a complete view of the requirements and design behind HL7 standards.

4.4.4 ICH Guidelines

ICH (International Conference on Harmonization of Technical Requirements for Registration of Pharmaceuticals for Human Use) has developed a series of procedural standards that act as guidelines for various stages of the clinical trials. It is based on the FDA regulations and has an international status as obligatory and as the cornerstone for regulatory bodies and ethical committees.

Among theses guidelines the most known and relevant to Metokis is the ICH E6 – Guidelines for Good Clinical Practice (GCP). ICH E6 is a standard for the design, conduct, performance, monitoring, auditing, recording, analysis and reporting of clinical trials that provides assurance that the data and reported results are credible and accurate and that the rights, integrity, and confidentiality of trial subjects are protected. It includes a glossary of terms and specific guidelines [ICH01] for the structure of the protocol, and of the IB (Investigator Brochure).

ICH - International Conference on Harmonization of Technical Requirements for Registration of Pharmaceuticals for Human Use (www.ich.org/UrlGrpServer.jser?@_ID=276&@_TEMPLATE=254) is a group of representatives from drug regulatory authorities in the US, (FDA) EU (EMEA) and Japan (MHLW, KIKO) and pharmaceutical organizations (i.e., PhRMA, EFPIA and JPMA).

4.5 Media Standards

4.5.1 MPEG-7

MPEG-7 [MPEG-7] is an ISO/IEC standard for describing features of multimedia content, which was developed by the Moving Picture Expert Group (MPEG). It provides a flexible and extensible framework for describing audio-visual content in such a way, that users can browse, search and retrieve content more efficiently than they could be using text-based search engines.

It standardizes a set of descriptors, a set of description schemes, and the Description Definition Language. A descriptor (D) is a representation of a feature that defines the syntax and semantics of the feature representation. A description scheme (DS) specifies the structure and semantics of relationships between components. These components may be either descriptors or description schemes.

MPEG-7 standard consists of the following components:

- *MPEG-7 Systems*: It includes the binary format for encoding MPEG-7 descriptions and the terminal architecture.
- *MPEG-7 Description Definition Language:* allows the creation of new Description Schemes (DS) and Descriptors (D).
- *MPEG-7 Visual:* It consists of basic structures and Descriptors that cover basic visual features: color, texture, shape, motion, localization, and face recognition.
- *MPEG-7 Audio:* It consists of both low level features for describing audio content (e.g. spectral, parametric, and temporal) and high level features such as sound recognition, indexing etc.
- *MPEG-7 Multimedia Description Schemes (MDS):* MDS provides descriptor schemes for: Content Description, Content Management, Content Organization, Navigation & Access and User Interaction.

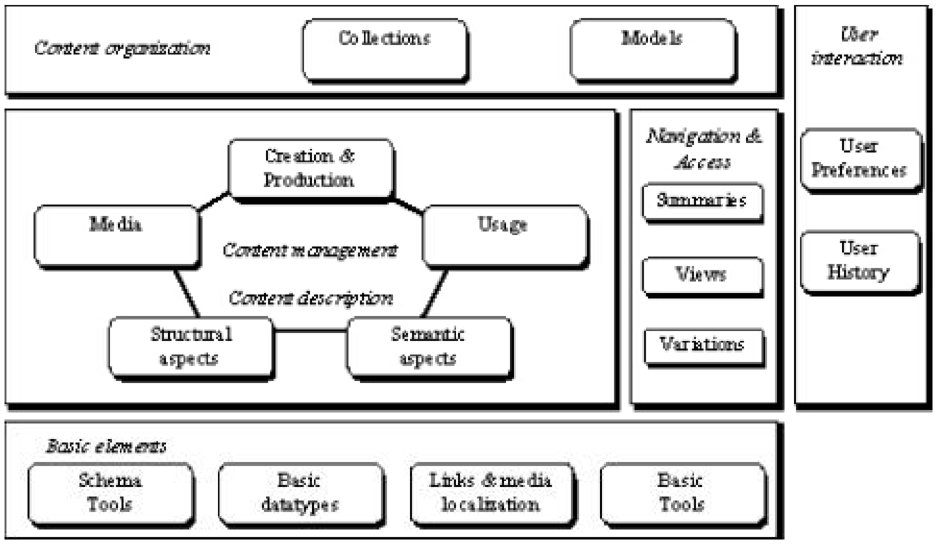


Figure 9: Overview of MPEG-7 Multimedia Descriptor Schemes
Source: MPEG-7 Overview (version 9) ISO/IEC JTC1/SC29/WG11N5525

4.5.2 MPEG-21

MPEG-21 [MPEG-21] aims at defining a framework for multimedia delivery and consumption. MPEG-21 is based on two essential concepts: "Digital Item" - a fundamental unit of distribution and transaction and the concept of "Users" interacting with Digital Items. The Digital Items have standard digital representation, identification & metadata (e.g., a video collection, a music album) and the Users can be considered the ones who either produce (content creators, publishers) or consume (persons, organisations) Digital Items. The standard defines the following components for building up a multimedia framework:

- *Digital Item Declaration (DID):* DID describes a set of abstract terms and concepts to form a useful model for defining Digital Items. The DID Model defines digital items, containers, fragments or complete resources, assertions, statements & annotations on digital items.

- *Digital Item Identification (DII):* The DII deals with unique identification of complete or part of Digital Items by encapsulating Uniform Resource Identifiers into the Identification DS. It also enables the identification of Digital Items via a Registry Authority.

- *Intellectual Property Management and Protection (IPMP):* It deals with management and protection of intellectual property within MPEG-21.

- *Rights Expression Language (REL):* REL helps declare rights and permissions using the terms as defined in the Rights Data Dictionary. An MPEG REL grant consists of: the principal to whom the grant is issued; the right that the grant specifies; the resource to which the right in the grant applies and the condition that must be met before the right can be exercised.

- *Rights Data Dictionary (RDD):* The Rights Data Dictionary (RDD) comprises a set of uniquely identified Terms to support the REL. RDD is designed to support mapping and transformation of metadata from the terminology of one namespace into that of another namespace.

- *Digital Item Adaptation:* It enables adaptation of digital content to preserve quality of user experience taking care of user, terminal or network characteristics.

- *Reference Software:* It deals with the architecture for processing Digital Items.

- *File Format:* MPEG-21 Digital Item consist of content of multiple formats textual (XML) and binary (still images).

4.6 Digital Rights Management Standards

4.6.1 <indecs> Metadata Framework

The <indecs> [Rust00] [Indecs00] [Indecs02] project provides a metadata framework for rights management for any type of creation; integrates descriptive metadata with commercial transactions and rights. The model separates and identifies three core entities: Users, Content, and Rights. Users can be any type of user, from a rights holder to an end-consumer. Content is any type of content at any level of aggregation. The Rights entity is an expression of the permissions, constraints, and obligations between the Users and the Content. The model provides a flexible mechanism for assigning rights to any combination or layering of Users and Content.

The framework proposes:

- a *generic attribute structure* for all entities;

- *events* as the key to complex metadata relationships;

- a *metadata dictionary* for multimedia intellectual property commerce;

- unique identifiers (*iids*) to be assigned to all metadata elements;

- the need for *transformation* processes to express the same metadata at different levels of complexity for different requirements.

4.6.2 IPR Onto

IPROnto [IPRONTO] defines an ontology for Intellectual Property Rights. The ontology consists of two parts: the static part defines concepts related to IPR and the dynamic part applies these concepts within a content life cycle putting actors, events and rights into context. IPROnto merges the existing efforts in Digital Rights Management such as <indecs>, DMAG [DMAG], Imprimatur [IMPRIMAT], WIPO [WIPO] using SUMO as an upper level ontology.

The root of the IPROnto ontology is Entity, which may be Physical or Abstract. A Physical entity may be an Object or a Process, which in turn might be an Event or a Situation. Agreements, including contract or license etc. come below events. LegalConcepts defining IntellectualPropertyRight, LegalEntity form a part of Abstract Entity.

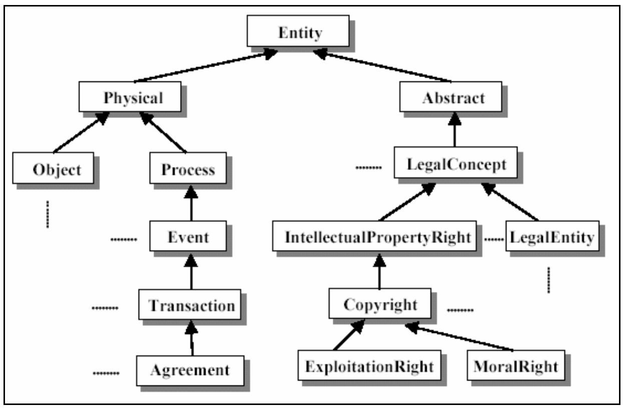


Figure 10: Core elements of IPROnto

The key concepts of IPROnto are:

- *Agreement:* Contacts and Licenses related to IPR form a part of the Agreement protocol in IPR Onto. They deal with contact by an author or a purchase contract or a distribution license etc.

- *Intellectual Property Right:* These provide right of ownership such as patents, copyrights, trademarks etc.

- *LegalEntity:* It refers to concepts defined by law, statute or international convention. Concepts refer to legal entities such as corporates, or individual persons. Roles such as Creator, Media Distributor, Customer etc. also form a part of LegalEntity.

5 Knowledge Standards & Upper Ontologies

5.1 Introduction

This section consists of two parts. The first part deals with various Knowledge Representation Standards and the second part deals with knowledge modelling techniques for modelling knowledge content objects.

5.2 Knowledge Representation Standards

5.2.1 KIF (Knowledge Interchange Format)

Knowledge Interchange Format (KIF) [KIF] is a formal language for the interchange of knowledge among disparate computer programs. The language has declarative semantics. It is possible to understand the meaning of expressions in the language without appeal to an interpreter for manipulating those expressions. KIF is logically comprehensive -- it provides for the expression of arbitrary sentences in predicate calculus. The language also provides for the representation of knowledge about the representation of knowledge. This allows one to make all knowledge representation decisions explicit and permits one to introduce new knowledge representation constructs without changing the language.

KIF syntax consists of 3 layers: Characters, lexemes (combination of characters) and expressions (combination of lexemes).

KIF supports logic operators: functional terms, logic terms, truth values, inequalities and sentences (Relational, Logical & Quantified). It also supports operations on numbers and provides the ability to define finite sequence of objects. There are 128 distinct characters known to KIF, corresponding to the 128 possible combinations of bits in the ASCII encoding. A string in KIF is defined as a list of characters. KIF also provides support for metaknowledge via naming expressions.

5.2.2 Conceptual Graphs

The Conceptual Graphs specification [ISO01] is an ISO draft version, which specifies the syntax and semantics of conceptual graphs. It aims to express meaning in a form that is logically precise, human readable, and machine processable. Conceptual Graphs provide full first order logic plus meta language capabilities.

A Conceptual Graph is a structure consisting of two kinds of nodes: concepts and conceptual relations. A conceptual graph is a bipartite graph in which there are no arcs between a concept and another concept, and no arcs between a conceptual relation and another conceptual relation. All arcs either go from a concept to a conceptual relation or from a conceptual relation to a concept. The arcs in conceptual graphs are always directed. Additionally, concepts and conceptual relations are typed.

The specification also introduces the concept of a “Module” which can be used to specify knowledge structures or ontologies. A module defines basically three things: First, the hierarchy of the types of concepts and conceptual relations, secondly the catalogue of individuals, which includes all entities that represent individuals that may be explicitly referenced in the module, and finally the outermost context, which asserts some background knowledge about those individuals. Only individuals which are contained in the catalogue can be referenced in the module. Individuals which are not catalogued might be implied by existential quantifiers in some concepts of a module.

CGIF (Conceptual Graph Interchange Format) is intended for transfer of knowledge between systems that use CGs as their internal representation. CG has the same model-theoretic semantics as Knowledge Interchange Format (KIF). They have the same expressive power and as such anything represented in CG or KIF can be translated to a logically equivalent form in the other.

5.2.3 XML Topic Maps (XTM)

The XTM specification provides an abstract model and XML grammar for interchanging Web-based topic maps. The XML Topic Maps (XTM) 1.0 specification was written by the TopicMaps.Org, which is an independent consortium of parties interested in making the Topic Maps Paradigm applicable in the World Wide Web. The Topic Maps Paradigm is fully described in the ISO/IEC 13250:2000 “Topic Maps” standard [ISO99].

The key concepts in Topic Maps are *topics*, *occurrences* and *associations*. A topic is a resource within the computer that stands for, or *reifies*, some real-world subject. It is a resource that acts as a proxy. A topic has assigned topic characteristics. There are three items, which can belong to the topic characteristics: a topic *name*, a topic *occurrence*, and a *role* that the topic plays in an association. The topic characteristics constitute the structure of the topic, and thus are the basis for topic map navigation and querying. A topic is specified by zero, one, or several names and can have occurrences.

An occurrence is any information that is specified to be relevant to a given subject. Occurrences must be resources, that are either addressable via Uniform Resource Identifier (URI), or can be placed inline as character data. Or if the occurrence is placed inline, a small piece of information, as the date of creation, can be expressed.

Topics can participate in relationships, called associations, in which they play roles as members. In other words, associations consist of members, and these members are topics. The topics participating in an association play a specific role. There exist no limit on the number of members of an association and an association itself does not specify any directionality. The directionality is determined by the type of the relationship, and by the roles the member play.

5.2.4 RDF/RDFS (Resource Description Framework)

Resource Description Framework (RDF) [RDF99] [RDF02] provides means to model meta-data about the resources on the web. The basic RDF model describes resources, properties and statements. Resources are all things being described, for example a web page, or a part of a web page, an object stored in a database, or an object which is not directly accessible via the web, i.e. a natural person or a printed book. Resources are identified by a resource identifier, which is a Uniform Resource Identifier (URI) plus an optional anchor id. Properties, which are resources as well, are specific aspects, characteristics, attributes and relations used to describe a resource. Property names must be associated with a schema.

RDF Schema (RDF's vocabulary description language) is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. In the RDF Schema language it can be expressed how one property relates to other properties, and which values are the permitted values of a property. A statement is the base element of an RDF model. It is a triple of subject, predicate and object. The subject is a specific resource, the predicate is a named property, and the object, which is the value of the predicate, is a second resource or a literal. Thus, a statement describes the semantic relationship between two resources. All the statements (or triples) result in a directed graph, where nodes and arcs are resources, which are labeled with URIs.

RDF & RDF Schema provides the following constructs for building ontologies and defining relationship between resources.

- *Classes:* Provides desciptions about Resources, Classes, Instances, Literals etc. e.g. rdfs:Resource, rdfs:Class, rdfs:Literal, rdfs:Datatype, rdf:XMLLiteral, rdf:Property.

- *Properties:* An RDF property is a relation between subject resources and object resources e.g. rdfs:range, rdfs:domain, rdf:type, rdfs:subClassOf, rdfs:subPropertyOf, rdfs:label, rdfs:comment.

- *Container Classes and Properties:* RDF containers are resources that are used to represent collections via rdfs:Container, rdf:Bag, rdf:Seq, rdf:Alt, rdfs:member, rdfs:ContainerMembershipProperty.

- *RDF Collections:* This provides constructs for defining closed collections, i.e. a list has only fixed number of members. rdf:List, rdf:first, rdf:rest, rdf:nil

- *Reification Vocabulary:* Defines vocabulary for reification. rdf:Statement, rdf:subject, rdf:predicate, rdf:object

- *Utility Properties:* Provides some utility properties for defining resources e.g. rdfs:seeAlso, rdfs:isDefinedBy, rdf:value

5.2.5 Web Ontology Language (OWL)

OWL [OWL] is part of the W3C recommendations for building ontologies related to the Semantic Web. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web. OWL is a revision over other more expressive ontology languages such as DAML+OIL.

OWL provides three increasingly expressive sublanguages depending on the varied use of semantics by the users and implementers. *OWL-Lite* supports users targeting mainly a classification hierarchy and simple constraints. *OWL-DL* provides maximum expressivity while retaining computational completeness & decidability. OWL-DL includes all OWL language constructs but they can be used only with certain restrictions to guarantee completeness & decidability. *OWL-Full* provides users maximum expressivity and syntactic freedom of RDF but with no computational guarantees. OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF.

OWL-Lite provides the follow constructs:

- *RDF Schema Features:* Consists of concepts in RDF Schema Language e.g. owl:Class (Thing, Nothing), rdfs:subClassOf, rdf:Property, rdfs:subPropertyOf, rdfs:domain, rdfs:range, owl:Individual.

- *Property Restrictions:* Defines restrictions over property values e.g. *owl:*Restriction, owl:onProperty, owl:allValuesFrom, owl:someValuesFrom

- *Class Intersection:* Defines intersection between classes using *owl:*intersectionOf

- *Datatypes:* Supports XML datatypes.

- *(In)Equality:* Supports equality of classes, properties and instances via owl:equivalentClass, owl:equivalentProperty, owl:sameAs, owl:differentFrom, owl:AllDifferent, owl:distinctMembers

- *Restricted Cardinality:* Supports cardinality e.g. owl:minCardinality (only 0 or 1) , owl:maxCardinality (only 0 or 1), owl:cardinality (only 0 or 1)

- *Versioning:* Provides supports to versioning of OWL documents e.g. owl:versionInfo, owl:priorVersion, owl:backwardCompatibleWith, owl:incompatibleWith, owl:DeprecatedClass, owl:DeprecatedProperty

- *Property Characteristics:* owl:ObjectProperty, owl:DatatypeProperty, owl:inverseOf, owl:TransitiveProperty, owl:SymmetricProperty, owl:FunctionalProperty, owl:InverseFunctionalProperty

- *Header Information:* Includes header information for OWL files e.g. owl:Ontology, owl:imports

- *Annotation Properties:* Includes properties for annotation e.g. rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:isDefinedBy, owl:AnnotationProperty, owl:OntologyProperty

OWL-DL and OWL-Full provides additional constructs while supporting all OWL-Lite constructs:

- *Class Axioms:* owl:oneOf, dataRange; owl:disjointWith; owl:equivalentClass (applied to class expressions), rdfs:subClassOf (applied to class expressions)

- *Arbitrary Cardinality:* Supports cardinality of any degree using owl:minCardinality, owl:maxCardinality, owl:cardinality

- *Boolean Combinations of Class Expressions:* owl:unionOf, owl:complementOf, owl:intersectionOf

- *Filler Information:* owl:hasValue

OWL DL requires type separation (a class can not also be an individual or property, a property can not also be an individual or class). This implies that restrictions cannot be applied to the language elements of OWL itself (something that is allowed in OWL Full). Furthermore, OWL DL requires an explicit separation between properties that are either ObjectProperties or DatatypeProperties which is relaxed in OWL-Full.

5.2.6 Semantic Web Rule Language (SWRL)

Semantic Web Rule Language (SWRL) [SWRL04] is a proposal (submitted to W3C) to combine ontology languages such as OWL DL and OWL Lite with the Unary/Binary Datalog rule languages such as RuleML [RuleML]. The proposal extends the set of OWL axioms to include Horn-like rules.

An OWL ontology in the abstract syntax contains a sequence of axioms and facts. Axioms may be of various kinds, e.g., subClass axioms and equivalentClass axioms. It is proposed to extend this with rule axioms. A rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a (possibly empty) set of atoms.

Both the antecedent (body) and consequent (head) consist of zero or more atoms. An empty antecedent is treated as trivially true (*i.e.*, satisfied by every interpretation), so the consequent must also be satisfied by every interpretation; an empty consequent is treated as trivially false (*i.e.*, not satisfied by any interpretation), so the antecedent must also not be satisfied by any interpretation. Multiple atoms are treated as a conjunction. Note that rules with conjunctive consequents could easily be transformed (via the Lloyd-Topor transformations) into multiple rules each with an atomic consequent.

Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x,y are either variables, OWL individuals or OWL data values.

SWRL uses an XML Concrete Syntax, which is a combination of the OWL Web Ontology Language XML Presentation Syntax with the RuleML XML syntax, for defining SWRL rules. It enables a user to freely mix rules and ontology axioms and to use arbitrary OWL classes (e.g. descriptions) as predicates in rules.

5.3 Content Modelling Techniques

This section describes different techniques and approaches for content & knowledge modelling for integrating multimedia content with knowledge for defining Knowledge Content Objects.

Multiple content standards occurring within one or multiple domains pose a tremendous challenge for defining generic knowledge objects. The ABC model is one such attempt to harmonise multiple domain ontologies via an upper level ontology integrating multimedia content with knowledge. A number of other upper level ontologies like SUMO, DOLCE bring together multiple ontologies and align them with their own upper level ontology.

Along with domain modelling, there are multiple methodologies to store and retrieve knowledge from content. AdobeXMP provides a platform for storage and retrieval of metadata from within the multimedia file by enabling storage of RDF encoded knowledge in binary format. EMMO (Enhanced

Multimedia Meta Object) model provides an object oriented representation for linking multimedia data with ontologies. The INKASS '"Information Objects" provide a conceptual model defining facets or functionality that is needed for defining Knowledge Content Objects.

5.3.1 ABC Model

The ABC ontology [ABC01] [ABC02] [ABC03] can be used to model physical, digital and analogue objects held in libraries, archives; abstract concepts such as intellectual content and temporal entities such as performances or lifecycle events that happen to an object. In addition the model can also be used to describe other fundamental entities that occur across many domains such as: agents (people, organisations, instruments), places and times.

The ABC models the following categories:

- *Temporality Category:* This category models time (Situation, Events and Actions) within the ABC model. Situation provides the context of time dependent properties of entities. Events mark a transition from one situation to another. Actions provide the mechanism for modelling responsibilities of agents for events.

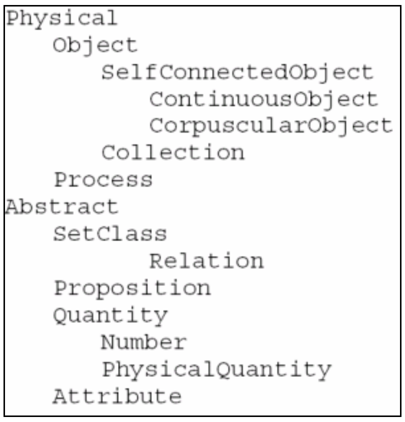
- *Actuality Category:* The Actuality ontology category encompasses entities that are sensible - they can be heard, seen, smelled, or touched.

- *Abstraction Category:* The Abstraction category makes it possible to express concepts and ideas. It helps to express the notion of Work as a means of binding together several manifestations of an intellectual expression.

The ABC ontology has been used to glue together domain specific ontologies with multimedia. The event aware model of ABC has been linked together with Museum ontology (CIDOC CRM) [CIDOC02], Biomedical domain (ON9.3), MPEG-7 and MPEG-21.

5.3.2 SUMO Ontology

The Suggested Upper Merged Ontology (SUMO [SUMO01]) is an upper level ontology that has been proposed as a starter document for The Standard Upper Ontology Working Group, an IEEE-sanctioned working group of collaborators from the fields of engineering, philosophy, and information science. The SUMO provides definitions for general-purpose terms and acts as a foundation for more specific domain ontologies.



```
Physical
   Object
      SelfConnectedObject
         ContinuousObject
         CorpuscularObject
      Collection
   Process
Abstract
   SetClass
         Relation
   Proposition
   Quantity
      Number
      PhysicalQuantity
   Attribute
```

Figure 11: Top Level SUMO Ontology

The root node of the SUMO is ‘Entity’, and this concept subsumes ‘Physical’ and ‘Abstract’. The 'Physical' category includes everything that has a position in space/time, and the 'Abstract' category includes everything else.

- *Physical:* The SUMO ontology embodies a 3D orientation (or "endurantists”) by making ‘Object’ and ‘Process’ disjoint siblings of the parent node ‘Physical’. The ‘Object’ concept consists of two disjoint sub concepts ‘SelfConnectedObject’ and ‘Collection’. A ‘SelfConnectedObject’ is any ‘Object’ whose parts are all mediately or immediately connected with one another.

The concept of ‘SelfConnectedObject’ is partitioned into two concepts: ‘ContinuousObject’ and ‘CorpuscularObject’. A ‘ContinuousObject’ is an ‘Object’ in which every part is similar to every other in every relevant respect. For example, substances like water and clay would be subclasses of ‘ContinuousObject’. ‘Collections’ consist of disconnected parts, and the relation between these parts and their corresponding ‘Collection’ is known as ‘member’ in the SUMO. Unlike ‘Classes’ and ‘Sets’, ‘Collections’ have a position in space-time, and ‘members’ can be added and subtracted without thereby changing the identity of the ‘Collection’. Some examples of ‘Collections’ are toolkits, football teams, and flocks of sheep.

Processes in SUMO forms a part of the physical world. PSL (Process Specification Language) [PSL01] has been incorporated into SUMO for defining processes.

- *Abstract:* The class ‘Abstract’ subsumes four disjoint concepts: ‘Set’, ‘Proposition’, ‘Quantity’, and ‘Attribute’. ‘Set’ is the ordinary set-theoretic notion, and it subsumes ‘Class’, which, in turn, subsumes ‘Relation’. A ‘Class’ is defined as a ‘Set’ with a property or conjunction of properties that constitute the conditions for membership in the ‘Class’, and a ‘Relation’ is a ‘Class’ of ordered tuples. The class of ‘Attributes’ includes all qualities, properties, etc. that are not reified as ‘Objects’.

  The ‘Quantity’ is divided into ‘Number’ and ‘PhysicalQuantity’. The 'Number' exists as a count independent of an implied or explicit measurement system, and 'PhysicalQuantity' refers to a complex consisting of a ‘Number’ and a particular unit of measure.

5.3.3 DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering)

Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE [DOLCE01]) is an upper level ontology developed within the WonderWeb project (http://wonderweb.semanticweb.org). DOLCE provides a *cognitive basis* for ontology development in the sense that it aims at capturing the ontological categories underlying natural language and human commonsense. The figure below shows the taxonomy of basic DOLCE categories.

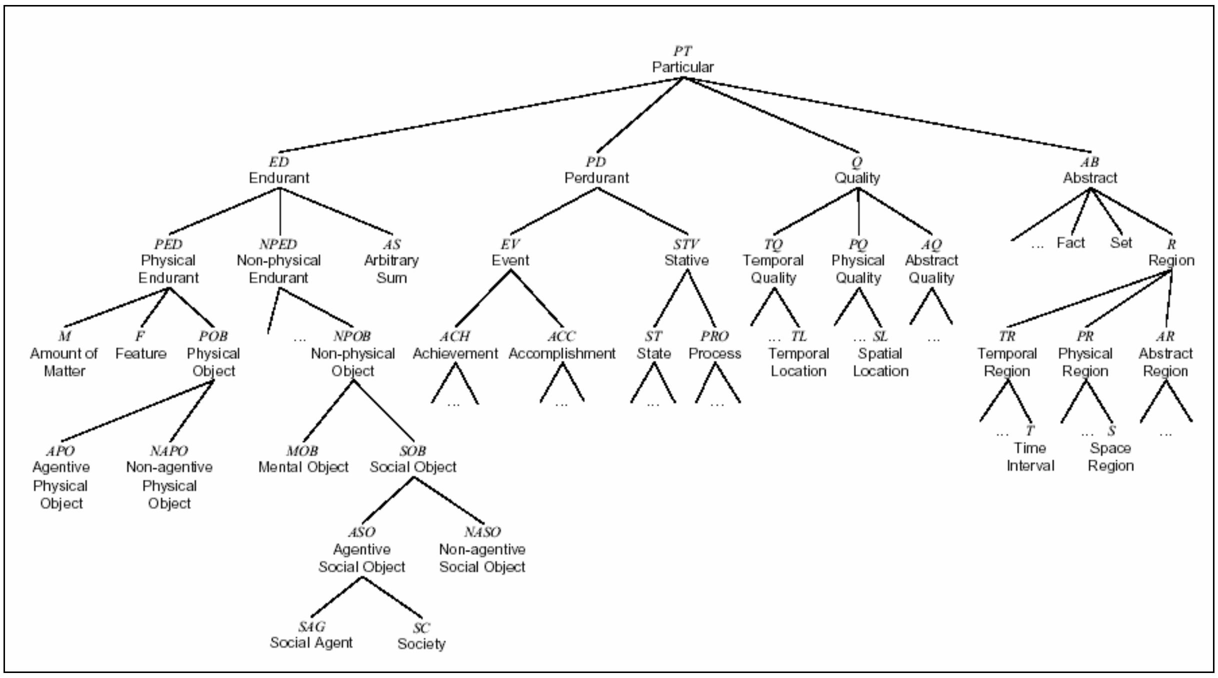


Figure 12: Taxonomy of DOLCE basic categories

The DOLCE ontology distinguishes between the following classes:

- *Endurants & Perdurants:* The difference between enduring and perduring entities is related to their behavior in time. Endurants are *wholly* present (i.e., all their proper parts are present) at any time they are present. Perdurants, on the other hand, just extend in time by accumulating different temporal parts, so that, at any time they are present, they are only *partially* present, in the sense that some of their proper temporal parts (e.g., their previous or future phases) may be not present.

- *Qualities & quality regions:* Qualities can be seen as the basic entities one can perceive or measure: shapes, colors, sizes, sounds, smells, as well as weights, lengths, electrical charges. The "value" (e.g., a particular shade of red) related to the qualities describe the position of an individual quality and form a part *quality space*. Space and time locations occur as specific qualities.

- *Abstract Entities:* Abstract entities do not have spatial nor temporal qualities, and they are not qualities themselves. Quality Regions act as Abstract Entities within the model.

Relationships model different relations that exist between the entities.

- Parthood & Temporary Parthood
- Dependence & Spatial Dependence
- Constitution
- Participation
- Quality Inherence & Quality value

5.3.4 WordNet

WordNet [WordNet01] [WordNet02] is an on-line lexical reference system where English nouns, verbs, and adjectives are organized into synonym sets, each representing one underlying lexical concept. WordNet divides the lexicon into five categories:

Nouns
Verbs
Adjectives
Adverbs
Function verbs

WordNet organizes lexical information in terms of word meanings, rather than word forms. Therefore, for organisation, semantic relations are used. Some of the relations that are used to form WordNet are defined below:

- *Synonym:* There are several definitions for synonym. One definition is that, two expressions are synonymous if the substitution of one for the other never changes the truth-value of a sentence in which the substitution is made. Another definition of synonym relative to a context is: two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value. Synonym is a lexical relation between word forms and it can be said to be symmetric.

- *Antonym:* The antonym of a word *x* is sometimes *not-x*, but this definition can not be generalised. Antonymy is a lexical relation between word forms and it is symmetric.
  Ex: rise - fall

- *Hyponymy/Hypernymy:* It is a semantic relation between word meanings. It is also called as subordination/ superordination, subset/superset, or the ISA relation. Hyponymy is transitive and asymmetrical. *x* is said to be a hyponymy of *y* if native speakers of English accept the sentence constructed as "An *x* is a (kind of) *y*."
  Ex: *tree* is a hyponymy of *plant*
  plant is a hypernymy of a *tree*

- *Meronymy/Holonymy:* It is a semantic relation which can also be called as part-whole or HASA relation. *x* is said to be a meronymy of *y* if native speakers of English accept the sentence constructed as "An *x* is a part of *y*".

5.3.5 Adobe XMP (Extensible Metadata Platform)

Adobe XMP [XMP01] provides a basic data storage platform along with providing metadata schemas for storing metadata in RDF, provides storage mechanism and defines a basic set of schemas for managing multimedia (versioning, media management, etc.)).

Data Model: The data model is derived from RDF and is a subset of the RDF data model. It provides support for:

- *MetaData Properties*: metadata consisting of a set of properties. Properties are associated with a resource. Properties have a property name (legal XML names) and have a property value e.g. the property authorOf Moby Dick is Herman Melville.
- *Schemas*: Definition of standard schemas and also allows extension of new schemas.
- *PropertyValues*: Property values can be simple types (literals), structures and arrays. Simple types are strings, dates, integer etc.
- *Structured Properties*: A structured property consists of one or more named fields. Its represented via an anonymous node in RDF sense e.g. A document has a maximum page size - Dimension (width, height, unit). The data model also allows Unordered arrays – Bag (RDF), Ordered Arrays – Seq (RDF) and Alternative Arrays – Alt (RDF).
- *Property Qualifiers*: Any individual property value may have other properties attached to it; these attached properties are called *property qualifiers*. They are in effect “properties of properties”; they can provide additional information about the property value. E.g. Moby Dick dc:creator Herman Melville. Now a role can be attached to Herman Melville - Role “author” via property qualifiers.
- *Language Qualifiers*: Similar to property qualifiers, language qualifiers can be attached to a property. Role is “xml-lang”.

Storage model*:* The implementation of the data model**.** It provides a serialization of the data model as RDF in a binary format via XMP packets. The metadata can be embedded inside the files (images, documents). Currently it supports the following formats: JPEG, GIF, tiff, html, xml.

Schemas*:* It consists of predefined sets of metadata property definitions that are relevant for a wide range of applications varying from media management, versioning and specific schemas Adobe uses within its tools etc. External schemas can be added as and when needed.

5.3.6 Enhanced Multimedia Meta Objects (EMMO)

EMMOs [EMMO02] [EMMO03] encapsulate meaningful relationships between multimedia objects and maps them into a navigable structure. An EMMO (Enhanced Multimedia Meta Object) is a self-contained object that can be created and worked on collaboratively and can be traded or exchanged over the Internet. The model defines the following entities:

- *Associations*: Particular relationships that exist between entities.
- *LogicalMediaPart*: Logical media parts describe parts of media objects or whole media objects on a semantic level.
- *Ontology Objects*: Ontology objects in the EMMO Model may on the one hand represent concepts from a particular ontology which are only referenced from the model or on the other hand they may be concepts that are themselves stored in the EMMO Model.
- *Emmo*: Emmo acts as an encapsulation over all of the above entities including itself. Therefore, emmos can contain other emmos.

The actual media objects are described via an MPEG-7 description (Media Profile) and a bridge (connector) links the Logical Media Part to the Media Profile. The EMMO model provides versioning support at entity level by providing predecessor successor relationships amongst entities. By having Associations as first class objects, it also provides the possibility to express statements about statements.

5.3.7 Information Objects

The INKASS project [Maass02] [Inkass01] (Intelligent Knowledge Asset Sharing and Trading) makes an attempt at providing a comprehensive model of knowledge resources, which enables organizations to trade knowledge objects of any sort in B2B eCommerce market places. The model comprises at present of 11 ontology "facets" which specify different aspects of the information object (IO).

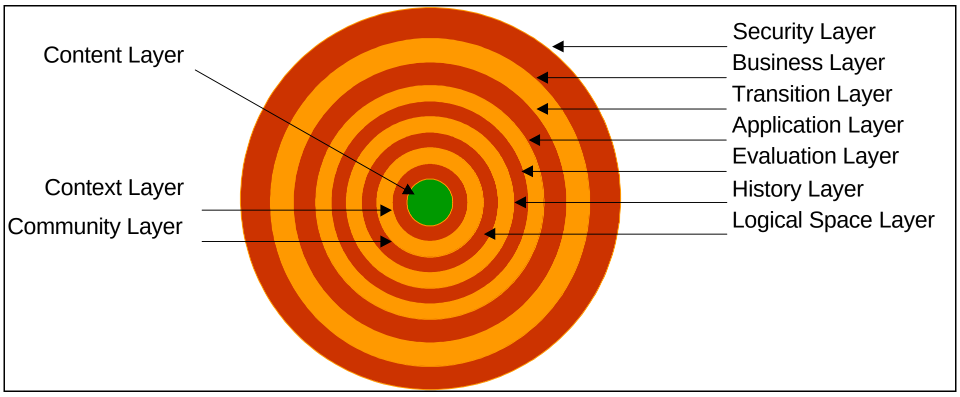


Figure 13: Information Ontology for Knowledge Assets

- *Content Facet:* Defines the core content of the Information Object.
- *Context Facet:* Describes the use and application of Information Object in an organization.
- *Community Facet:* Describes the community of agents interacting with an IO.
- *Domain Facet:* Ensures all content specific statements about an IO are understandable and interpretable.
- *History Facet:* Contains information about creation, modification and change history of an IO.
- *Evaluation Facet:* Contains information to access the quality of an IO.
- *Method Facet:* Contains information about technical provisions required to apply some knowledge described by an IO.
- *Transition Facet:* Describes how the application of some knowledge may affect or change the application environment.
- *Business Facet:* Information required to establish trading functionalities.
- *Legal aspects*: IPR Information to conduct legal transactions.
- *Security Facet:* Information required to ensure that the whole transaction on the web is secure.

6 Knowledge Content Objects (KCO)

Having discussed relevant research projects and standards in the previous sections, we will now present our current design of the components that make up the METOKIS Architecture. It seems best to start the description with the objects for which the architecture will be developed.

6.1 Motivation

The background model for METOKIS is that "actors" have "knowledge" and receive "information". Furthermore, the actors have access to a large resource space (typically the WWW) from which they can draw further information. The actors make use of their existing knowledge, weave into the existing knowledge the new information and eventually, "interpret" their growing "knowledge space" with respect to their current or future "action space". When the interpretation is done with reference to a future action space then it is called "planning". When the interpretation is done with reference to the current action space then it is called "doing". Using the WWW, actors can interact with other actors irrespective of whether they are humans or software. The way to communicate (i.e. have controlled and controllable interaction) is by exchanging meaningful statements between the actors. Human beings are capable of using natural language for this task. When machines are involved, surrogates must be found for natural language and for the notion of meaningful statements.

METOKIS could be seen as an attempt to create an environment for software objects that include media for human consumption as well as a translation of those media for machine consumption. This way, the new software objects become a (surrogate) means of communication. The analogy would be that the actors are writing a special sort of letters to each other. The inner structure of these letters helps the machines to separate out what is meant for them and what is for the humans to interpret. This special sort of "letter" - which can be exchanged between humans and humans, humans and machines, as well as machines and machines - we call a *knowledge content object*.

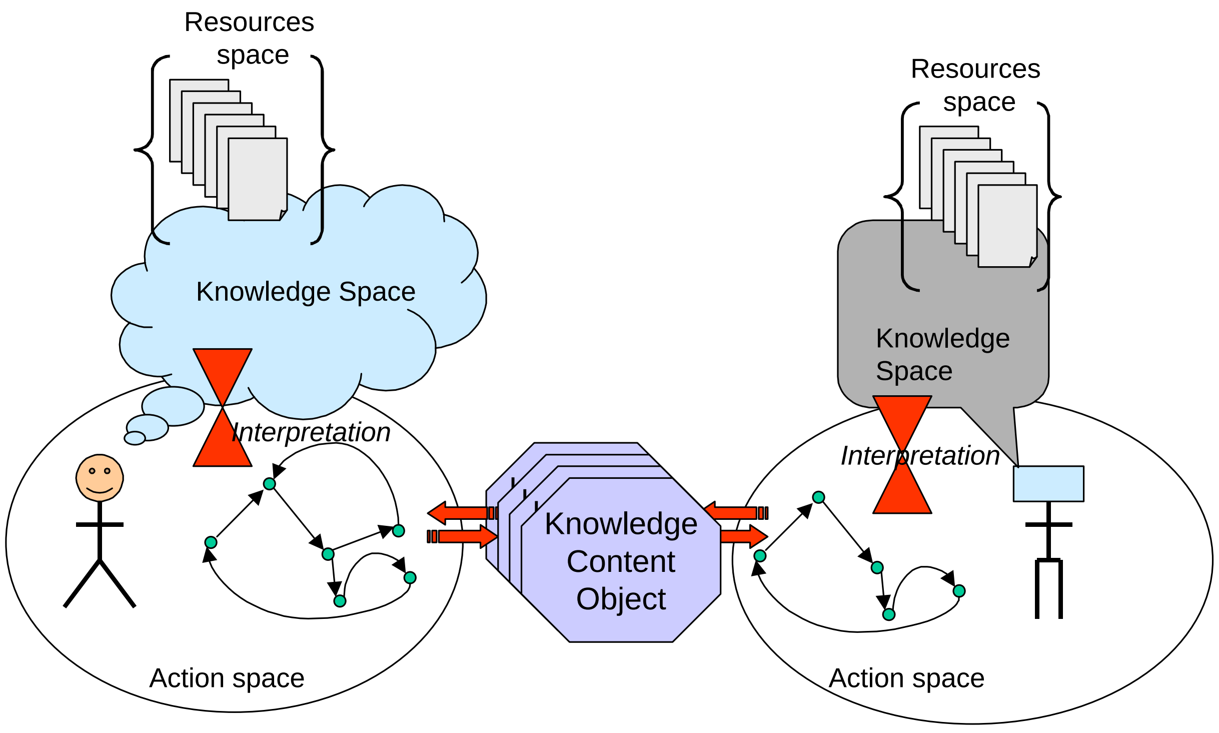


Figure 14: The Role of Knowledge Content Objects in an Environment of Resource-, Knowledge- and Action spaces

6.2 METOKIS Intended Environment

The above diagram illustrates how knowledge content objects are aimed at crossing the boundaries between human and system action spaces, by providing close-to-equivalent descriptions of aspects of the real world, to system actors (e.g. software agents or web services). The above situation can be modelled in terms of a distributed architecture (see section 7). The *intention* of this environment is far-reaching: essentially, we want to work towards the vision of ambient intelligence where machines and humans exchange information/knowledge at will and seamlessly. In the METOKIS *project*, we want to focus on the provision of a minimal infrastructure needed to exchange knowledge and media in such mixed human-machine environments, through non-proprietary protocols and formats.

In METOKIS, the *Action Space* is modelled in the *Task Taxonomies*. The Resources Space is any web-enabled resource, and the Knowledge Space has two manifestations: *human knowledge* (some of which may be encoded and stored in the Resources Space) and *system knowledge* (some of which may be encoded and stored in the Resources Spaces, and some of which may be encoded in the actual current state of the acting system).

The process of *Interpretation* uses knowledge constructs in order to behave (i.e. do things) as afforded by the actor's Action Space. Interpretation happens in two directions: knowledge content objects - that are received through the action space - can be incorporated into the Knowledge Space or conversely, internal knowledge constructs can be transformed into sequences of actions within the confines of the actor's action space. This may include the transformation of a resource into an internal knowledge structure, the transformation of that internal knowledge structure into a knowledge content object, and finally, the transmission of a knowledge content object to some other actor.

In METOKIS, the process of Interpretation is delegated to *direct implementations* of the domain specific user applications. Thus, the task models will remain *implicitly encoded* in these applications. However, it is conceivable - and could subject of further work - to extend the system by a full-blown task execution environment which can interpret task specifications and which employs sophisticated reasoning engines to execute the tasks according to their specifications, rather than the bespoke applications that are currently the state of the art.

In the remaining sections of this chapter, we describe the inner structure and the intended semantics that are carried by knowledge content objects.

6.3 Elements of a Knowledge Content Object

Knowledge content objects take further, some of the intuitions of Topic Maps, but are giving much more emphasis to logic-based knowledge representation schemes such as Conceptual Graphs, as well as existing meta data standards such as MPEG-7. KCOs merge the concepts developed within CULTOS and INKASS to build knowledge enhanced multimedia content objects which can be shared and interpreted by middleware systems and applications. The focus is not just on definition of knowledge content objects but also on how these objects can play a significant role within the system considering their lifecycle within systems. The next sections of this chapter will define the KCO and how the KCO semantics can play an active role within multimedia systems and tools.

KCO's should foster knowledge exchange both where structured data and communication over structured data is involved such as in Web Services and also in un-structured data such as multimedia content, multimedia documents etc. which are kept within content repositories and are shared amongst systems. The focus for KCO or Knowledge Content Objects is to be able to hold enough semantics to be able to foster knowledge exchange amongst middleware platforms.

KCO's should support knowledge reuse situations such as (1) shared work production, (2) shared work practioning, (3) expertise seeking by novices and (4) secondary knowledge mining. If KCO's are exchanged between contexts with different conceptualisations, KCO's should support this exchange by referencing to the original ontological representations and - if translation mechanisms are available - to representations of the targeted context. Linking mechanisms as discussed in INKASS, allow mappings of KCO's with associated ontologies.

The vision of a KCO is to provide all information that is required to automatically process structured and unstructured content by receiving mechanisms. For unstructured content, this vision of strong typing can only be approximated but with the KCO we intend to extend the flexibility and the processability of unstructured content. In this sense, strong typing is emulated by providing secondary, semantic information on the content that provides strongly typed structured and processible information.[1]

By conceptually aligning the data structures of an "information object" (INKASS) and an EMMO (CULTOS), we believe that a KCO needs to carry the following semantics (some of the semantics are new and were neither present in Information Objects nor EMMOs).

KCO	Propositional Content	
	Time based spatial rendition	
	Interaction based spatial rendition	
	Multimedia metadata	IPR information
		Media properties
		Content classification scheme
	Usage context	User task
		User Community
		Usage History
	Business semantics	License
		Contract
		Pricing
		Negotiation
		Trading
	Trust	
	Access semantics	user authorisation
		processing policies
	KCO self description	

Table 2: Overview of KCO Structure

6.3.1 Propositional Content

This is a graph structure which allows the definition of a semantic network over a set of media assets. For example, we can express that a specific scene in a film is a parody of a section of text in a novel and that the protagonist of the novel is called "Don Quixote". The structure is also self-describing in terms of knowledge representation language used for the specification of the semantic network.

- **RootURI** - a single URI denoting the place from which the segments of the knowledge content object can be accessed. The RootURI is at the same time, the object identifier for the KCO.
- **MediaTokenURIs** - the set of tuples **`mediatoken`**`(RootURI, MediaTokenURI)` at which representations of actual Media can be found.
- **SegmentTokenURIs** - the set of tuples **segmenttoken**`(MediaTokenURI, SegmentTokenURI)` at which representations of the actual segments can be found.
- **MediaURIs** - the set of tuples **`mediauri`**`(MediaTokenURI, MediaURI)` at which each of the media files associated to an abstract media can be found.
- **KnowledgeAssociations** - the set of structural links (multigraph edges) that can be associated to any two MediaTokenURIs or SegmentTokenURIs. The format is **`knowassoc`**`(<MediaTokenURI>|<SegmTokenURI>,`
`EdgeType,  (<MediaTokenURI>|<SegmTokenURI>)`

[1] In contrast to programming language, strong typing here is not related to compile time but to the event of content publication.

- **KnowledgeAnnotations -** the set of logic statements - referring to some ontology - associated with each KnowledgeAssociation.
  ```
  knowannotation(<MediaTokenURI>|<SegmTokenURI> |
  <KnowledgeAssociation>, OntologyTerm, OntologyRef, KRLRef,
  <MediaTokenURI> |<SegmTokenURI> | <KnowledgeAssociation>)
  ```

OntologyRef means a reference to one of several possible ontologies by which a content can be annotated. KRLRef denotes the knowledge representation language used to express OntologyTerm. (This allows multiple KRL annotations of the same resource).

6.3.2 Time-based spatial content rendition specification

Given a propositional content, we need to specify how that content is intended to be rendered under certain circumstances. For instance, if we had a semantic structure for telling jokes then we would probably choose a rendition that starts with the intro, sets up an expectation, and then delivers the punch line. Our jokes could be semantically annotated in this fashion. In order to be rendered correctly, we would now need a specification that states for all jokes, that they are best told in the order intro, expectation, punch-line.

A further specification mechanism could be to use a mapping between the knowledge structures and e.g. the SMIL description language [SMIL], thus creating a multimedia presentation from applying rendering rules to the knowledge structure that overlays the media network.

We expect other projects to develop specification languages for time-based spatial content and therefore, do not attempt to re-invent such a language in METOKIS.

6.3.3 Interaction based spatial content rendition specification

Given a propositional content, we need to specify how that content is intended to be used. For example, if we had a psychological study on computer games and we wanted to let users play an episode of a game and then let them answer questions before carrying on playing, then we need to be able to specify that rendition, e.g. "for each episode e, ask the user to answer the questionnaire q(e)".

We expect other projects to develop specification languages for interaction based spatial content rendition, and we foresee such specifications to be carried in the overall content description.

6.3.4 Multimedia Metadata Description

In line with the ideas of EMMOs, a KCO is primarily a description of some (mixed!) media content but it is not necessarily tied to any specific instances of the media referred to in the description. Whether or not a tight coupling is desirable depends on the *intended usage* of the media in conjunction with the description. The description then specifies how tight the coupling is intended to be and how strongly that coupling is intended to be enforced[2].

It is a technical option for a KCO to carry the actual multimedia content (audio, video, documents etc.) "on-board" (known as the "fat EMMO" option). This should be done by setting the MediaURI relative to the KCO's RootURI.

For each MediaURI, it is possible to have a Metadata Description which will be at least partially compliant with one or more of various standards (e.g. MPEG-7, MPEG-21, indecs, Dublin Core, Adobe XMP or EXIF - a metadata standard for image files of digital cameras). However, we propose that KCOs comply with a unified metadata ontology that allows mediation between the different, overlapping content description standards. The most likely starting point for such a unified meta data ontology is the ABC model. We must strive for information preserving mapping i.e. information that the KCO Model does not understand must be retained so as to enable interpretation by an external application. The Metadata Description intends to address the following aspects:

[2] For example, the current discussion of shutting down USB based memories or limiting MP3 players so that they cannot be used for illegal copying could benefit from such distinctions. The self-descriptive content may carry clear information about its intended usage, in the same way as a road traffic sign carries clear information about the maximum speed allowed. The current tendency of the industry is to activate a limiter on the user's machine (enforcement) whereas in the real world, some traces of free will, can still be detected in the legislation, i.e. your car will not be disabled by the "owner" of the road traffic sign, but you may get fined for speeding. The technologists in content related research need to understand their social responsibility in this.

- Intellectual and original provenance of the content
- Properties of the media that encode the content
- Classification of the content according to traditional description schemes

Note that this aspect of the KCO tries to answer questions such as: "who is your creator?", "what are you made of?", and "what are you about?". These questions address *endurant* aspects of the KCO.

6.3.5 KCO Usage Context

This aspect of KCOs is derived from the INKASS model of Information Objects. For KCOs, we distinguish:

- **`userTaskContext`**`(TaskOntology, Usertask, <UsageSpecification>)` - given a task ontology and a task label derived from that ontology, this tuple specifies that the knowledge and media content of this KCO is associated with the user task as specified. The usage specification can express one of these semantics concerning the content of a KCO: `[(+/-)` **`can`** `| (+/-)` **`must`** `| (+/-)` **`is-recommended-to`**`] be used in <UserTask>.`

  The user task and the usage specification define the situative embedding in which the content can be used. The primary "context ontology" is represented by the task definitions which are supported by the domain application.

- **`userCommunityContext`**`(UserGroup, UserRole, RightsVector)`
  the community ontology contains constraints that are related to the organization which can use and interact with a KCO. User Roles are defined together with their rights and obligations. The RightsVector is an N-tuple of specification attributes for rights and obligations which can each take different but enumerated values that are then interpretable by a KCCA compliant system.

- **`usageHistoryContext`**`(KCO, TrailsModel, UserTrails)`
  This includes the history of the media, in which context it has been used along with versioning information to keep track of the usage. The usage may be described according to different trailsmodels and each user trail is an instantiation of such a model. Again, it may well be desirable for the KCO not to carry its user trails on-board, but to have them kept in a trusted repository. Of course, it must be possible to have a null-model for trails which entails that no user data whatsoever is being kept about this KCO.

6.3.6 KCO Business Semantics

All business relevant attributes are described by business ontologies. This layer includes information such as on applicable pricing schemes and negotiation schemas. It also contains information on contractual issues and links into corresponding contracts.

- **`kcoLicenseScheme`**`(RootURI, LicenseSchemeURI)`
- **`kcoContract`**`(RootURI, ContractSpecification)`
- **`kcoPricing`**`(RootURI, PricingScheme)`
- **`kcoNegotiationScheme`**`(RootURI, NegotiationScheme)`
- **`kcoTradingProtocol`**`(RootURI, TradingProtocol)`

Industry specific business ontologies currently emerge that will provide the basis for libraries of business ontologies. Business conceptualisation, such as provided by UDDI and ebXML, are considered as starting points. Both are meta-models for business ontologies that are instantiated by domain specificiations.

Legal (Rights management, IPR, copyright): Legal or regulatory ontologies contains information about legal aspects such as intellectual property rights and copyrights (→ WORM 2004).

6.3.7 KCO Trust Semantics

The purpose of trust semantics is twofold: on the one hand, users can evaluate the trustworthiness of a resource if other users can leave (genuine and verifiable) endorsements. On the other hand, any kind of quality feedback is also of interest to the owner of the KCO as it allows improvements on the basis of trustworthy evaluation by users.

Evaluation (deals with quality of information represented by a KCO): Ratings, reviews, and other qualifications about the content of an information object are described by evaluation ontologies. Evaluations are highly context-dependent. For instance, ratings that are applicable in one domain might be irrelevant in another domain. Background mappings between evaluation ontologies and in particular ratings are required.

There is a relationship between user trails and user feedback. Both will need to be used to express various dimensions of trust. At present, research into trustable knowledge and content objects is in its infancy and therefore, specification of the KCO trust semantics will be deferred to a later version of KCOs. It should be noted that the issue is of interest and importance, e.g. in the application case of assessments of news stories by senior executives.

6.3.8 KCO Access Semantics

KCO access semantics are not related to the content and utility of a KCO but to control and technical processing issues.

Security/Access Permissions: The security layer contains information about how a KCO can be used regarding security issues. If a KCO is encoded, the security layer describes the kind of security protocol by which it can be accessed.

6.3.9 KCO Domain Semantics

This layer carries a self-description of the KCO (i.e. the meta-level description or ontology schema of KCOs). It is subject to further research what the properties of this KCO element should be.

6.4 First-Level KCO Operators

In this section we describe the operational semantics for KCOs at the level of generic operators that take the whole KCO and its first level structure as operands. These operators are:

ADD (kco, database)
REMOVE (kco, database)
UPDATE (kco, component, database)
QUERY(kco, components, query-expression, database)
MERGE (kco1, kco2, database)
RENDER(kco, target-application)
CONVERT(kco1, kco2, database)

The ADD and REMOVE operators work on whole KCOs.

The UPDATE operator changes the designated component of a KCO in the database.

The QUERY operator acts similar to a database cursor and specifies which components of a KCO are to be accessed by the query expression.

The MERGE operator takes two KCOs and fuses their components as indicated below:

Operand 1	Operand 2	Return Value	Description
kco1	**kco2**		
propositional content 1	propositional content 2	prop1 AND prop2	Logical AND of the propositions, it is up to applications to test for logical inconsistencies
time-based spatial rendition 1	time-based spatial rendition 2	UNDEFINED	Interaction is needed in which the time scales of kco1 and kco2 get synchronised

interaction based spatial rendition 1	interaction based spatial rendition 2	UNDEFINED	Interaction is needed in which the interaction schemes of kco1 and kco2 get synchronised
Metadata-IPR 1	Metadata-IPR 2	IPR1 AND IPR2	The most stringent interpretation of IPR1 and IPR2 is adhered to
Metadata-MediaProperties 1	Metadata-MediaProperties 2	MProp1 OR MProp2	Either the properties of media in KCO1 or those of media in KCO2 hold.
MetaData-Classification 1	MetaData-Classification 2	MD-Class1 OR MD-Class2	Either the classification of a media item according to the classification 1 holds, or according to the classification scheme 2
Context-userTask 1	Context-userTask 2	userTask1 OR userTask2	We hypothesise that two KCOs are only merged when it is believed that the resulting KCO will be usable in both task contexts
Context-user Community 1	Context-user Community 2	userCommunity1 OR userCommunity2	see above - both communities are assumed to be able to use the merged KCO
Business-License 1	Business-License 2	(License1 AND License2) OR UNDEFINED	Both license regulations need to be satisfied or an update will be needed
Business-Contract 1	Business-Contract 2	(Contract1 AND Contract2) OR UNDEFINED	Both contracts need to be satisfied or an update to the contract will be needed
Business-Pricing 1	Business-Pricing 2	PRICE1 AND PRICE2 OR UNDEFINED	The conservative semantics is that the combination of two KCOs costs as much as the sum of the two.
Business-Negotiation 1	Business-Negotiation 2	Negotiation of KCO1 if scheme(KCO1) INCLUDES scheme (KCO2), KCO2 if it INCLUDES KCO1, UNDEFINED otherwise	If the two negotiation schemes differ from each other, then the discrepancies have to be resolved manually
Business-Trading 1	Business-Trading 2	Trading scheme of KCO1 if scheme(KCO1) INCLUDES scheme (KCO2), KCO2 if it INCLUDES KCO1, UNDEFINED otherwise	If the two trading schemes differ from each other, then the discrepancies have to be resolved manually
Trust 1	Trust 2	Trust of KCO1 if trust(KCO1) < trust (KCO2), trust (KCO2) otherwise	The conservative semantics would be that the combined trust in the resulting KCO is at least as big as in the least trusted KCO
Access-user authorisation 1	Access-user authorisation 2	Access rights of KCO1 if access(KCO1) < access (KCO2), access (KCO2) otherwise	as above.
Access-processing policies 1	Access-processing policies 2	Processing rights of KCO1 if processing rights (KCO1) < processing rights (KCO2), processing rights (KCO2) otherwise	as above
KCO Ontology 1	KCO Ontology 2	Ontology of KCO1 if version(KCO1) < version (KCO2), KCO2 otherwise	as above

Table 3: Operational Semantics of the KCO-MERGE Operator

The RENDER operator only acts on the Propositional Content element and renders it in accordance with the specifications.

The CONVERT operator takes one of the KCO Elements and transforms the source element into a target element. The operation is defined primarily from the logic description to all other elements and the converse, from all others to the logic description. This allows that e.g. a KCO trading information can become a KCO content which is itself tradable. Likewise, a logic description of some trading information carried in one KCO can be converted into the actual trading information pertaining to some other KCO.

6.5 Second-level (Element level) KCO Operators

We now define the semantics of KCO operators that take the inner structure of KCO elements as operands.

6.5.1 KCO Propositional Content Operations

The functional group of knowledge instance manipulation allows for sub-graphs of the KCO to be identified (same as setting a cursor in a database) and statements within the identified scope (i.e. the sub-graph) to be altered.

Operator	Operands / Parameters	Return Value	Description
Query	KCO-ID, "LD", "all"	Logic description	Retrieves the logic description i.e. the ontological statements associated with this KCO.
Query	KCO-ID, "LD", Q-TERM	Query result	Retrieves an aspect of the logic description expressed by the query term
Add	KCO-ID, "LD", Subgraph	OK if successful, FAIL otherwise	Adds knowledge items to the LD, relative to the scope of the KCO sub-graph. If no match can be found then the operation FAILs
Delete	KCO-ID, "LD", Subgraph | "all"	OK if successful, FAIL otherwise	Deletes the logic descriptions belonging to the specified subgraph LD from a KCO. If no match can be found then the operation FAILs
Update	KCO-ID, "LD", Subgraph	OK if successful, FAIL otherwise	Replaces the current instances of the specified sub-graph with the instances attached to the Subgraph parameter. This operation can fail if the content in question is locked
Merge	KCO-ID1, KCO-ID2, "LD", Subgraph-1, Subgraph-2,	KCO-ID3, MergedSGraphs	Fuses the two knowledge subgraphs and puts them in a new KCO (KCO-ID3)
convert	KCO-ID1, KCO-ID2, "LD", <TRG> Subgraph-1, Subgraph-2,	OK | PARTIAL | FAIL	Takes a logic description which describes (possibly only in part) the content of another element of a KCO (e.g. the business services) and converts it into the target (<TRG>) KCO element schema. The precise semantics will depend on the exact element schema.

Table 4: Knowledge Domain Operations

Specific Semantics of Ontology Access Operations
A defined KCO has two ways in which it can be grounded in an ontology. The first is by reference to a designated external ontology. The second is by reference to an "on-board" ontology that only holds for the KCO itself. Any KCO can be queried for information concerning its ontological mark-up. Each of the query predicates can be further parameterised.

Operator	Operands / Parameters	Return Value	Description
query	KCO-ID, "LD", "ontorefs"	List of Ontology URIs	retrieves URIs of relevant Ontologies (on-board ontologies are marked as "local")
query	KCO-ID, "LD", "ontology", <OntoURI>	Ontology graph from <OntoURI> | NULL	Returns the specified ontology used in the KCO or NULL if the specified ontology is not used in the KCO

Table 5: Specific Ontology Access Operations

There are no add, update, delete, merge, render or convert operators defined because this type of manipulation would not be sanctioned externally.

6.5.2 Time based spatial content rendition operations

Operator	Operands / Parameters	Return Value	Description
Query	KCO-ID, "TSC", "all"	Logic description	Retrieves the time based spatial rendering specification which defines how media associated to specific logic statements should be rendered over time.
Query	KCO-ID, "TSC", Q-TERM	Query result	Retrieves an aspect of the rendering expressed by the query term
add	KCO-ID, "TSC", Subgraph	OK if successful, FAIL otherwise	Adds a rendering specification sub-graph. If no match can be found then the operation FAILs
delete	KCO-ID, "TSC ", Subgraph | "all"	OK if successful, FAIL otherwise	Deletes the rendering specification belonging to the specified subgraph LD from a KCO. If no match can be found then the operation FAILs
update	KCO-ID, "TSC", Subgraph	OK if successful, FAIL otherwise	Replaces the current instances of the specified sub-graph with the instances attached to the Subgraph parameter. This operation can fail if the content in question is locked
merge	KCO-ID1, KCO-ID2, "TSC", Subgraph-1, Subgraph-2,	KCO-ID3, MergedSGraphs	Fuses the two rendering specification subgraphs and puts them in a new KCO (KCO-ID3)
render	KCO-ID, "TSC", TargetFormat	RenditionData	render is an operator which acts on a KCO or on a multimedia meta object and interprets the rendering information with respect to a target format (e.g. SMIL or FLEX$^{(TM)}$)
convert	KCO-ID1, KCO-ID2, "TSC", "LD", OntoRef, Subgraph-1, Subgraph-2,	OK | PARTIAL | FAIL	Takes a TSC and converts it into a KCO LD schema, with reference to the ontology specified by OntoRef. The precise semantics will depend on the exact TSC element schema.

Table 6: Time based spatial rendition - Operations

6.5.3 Interaction based content rendition specification operations

These operators are essentially the same as the time based content rendition, except that the content here specifies how the system is planned to interact with a user in relation to this knowledge content object. We expect to include models here, which are developed in the realms of educational or games technology. The identifying parameter is "ISC" for interactive spatial content. It should be noted that for example in e-learning applications, student / system or student / teacher / system interactions may need to be specified and these may differ over the same content. By separating this description layer from other rendering information, reuse is enabled. METOKIS may do some initial modelling of this, in the course of developing the educational application case.

6.5.4 Multimedia Metadata Description and Services

We base this element of the KCO definition on the work done by Hunter, Lagoze and Doerr who developed ontologies [ABC01, ABC02, ABC03] for integrating MPEG-7 (media meta data), Dublin Core (cataloguing meta data) , Indecs (IPR meta data) and the CIDOC CRM (domain specific cataloguing model for cultural heritage). Their work shows the strengths of using ontologies to arrive at a canonical model across overlapping standards (which are also just models), and we put this body of work to the test. Our hypothesis is that media properties, cataloguing information and IPR information is a reasonably well bounded domain for knowledge content of any sort to bring these together in one conceptual "package". The linking of this generic information to specific domain models (e.g. "how to extinguish fires on oil wells") should - in our view - be done in a transparent, yet explicit fashion, which is why we separate "propositional content" (i.e. what is represented by the media) from "metadata" (i.e. how the content is shaped, packaged, formatted, labelled and sold). The functionality of the content metadata description services is two-fold: firstly, it implements a management layer for media resources and secondly, it offers interfaces to meta data management systems by using the internal ontology that streamlines implementations of multiple, overlapping standards.

Operator	Operands / Parameters	Return Value	Description
query	KCO-ID, "MMD", "all"	MMD Instantiation | NULL	Since MMD is a mandatory element, the only "failure" of the "all" query can be a NULL value if no instantiation has been found
query	KCO-ID, "MMD", Q-TERM	MMD Inst. | NULL | FAIL	If the Q-Term does not match with the schema then a FAIL is returned. If no values can be found, NULL. Else the instantiation of the Q-Term is returned (subgraph)
add	KCO-ID, "MMD", Sub-graph	OK | FAIL	If the subgraph to be added does not match the MMD ontology (i.e. if at least one sub-graph label differs) then the operation fails. This is to ensure that only consistent sub-graphs are added.
delete	KCO-ID, "MMD", Subgraph | "all"	OK | FAIL | FAIL-LOCKED	If sub-graph does not match: FAIL If sub-graph instances are locked - FAIL-LOCKED (user may be able to "manually" delete the non-locked instances of the sub-graph)
update	KCO-ID, "MMD", Subgraph	OK | FAIL | FAIL-LOCKED	See above.
merge	KCO-ID1, KCO-ID2, "MMD", Subgraph-1, Subgraph-2,	OK | FAIL-RESOLVE	If two MMDs are merged and the sub-graphs have different values attached to identical attributes then the user has to resolve the conflicts "manually", or accept failure of the operation
render	KCO-ID, "MMD", TargetFormat	NULL	The render operator is not applicable to meta data. If the knowledge contained in the MMD should be rendered then the MMD should first be converted into a logical description "LD".
convert	KCO-ID, <SRC-COMP>, <TRG-COMP>	OK | PARTIAL | FAIL	OK if a full conversion succeeded, PARTIAL if at least one element was converted (to be accepted by the user); FAIL if zero conversions.

Table 7: Metadata Description Operations

6.5.5 KCO Usage Context and Operations

Since we have three aspects to cover in this KCO element, the operators need to have specified semantics for these aspects:

- **`userTaskContext`**`(TaskOntology, Usertask, <UsageSpecification>)`
- **`userCommunityContext`**`(UserGroup, UserRole, RightsVector>)`
- **`usageHistoryContext`**`(KCO, TrailsModel, UserTrails)`

Operator	Operands / Parameters	Return Value	Description
query	KCO-ID, "USG", "task"	KCO-ID, Task description | NULL	The query returns the task context in which this content is deemed to be usable. The higher in the hierarchy, the more general the intended usage of the content. NULL when no intended task context is specified
query	KCO-ID, "USG", "cmty"	KCO-ID, User:Group:Role:Rights tuples | NULL | FAIL	The query returns the *intended* community of users, the roles that would use the content and the rights one would give to the roles. Note that this should be interpreted as a recommendation (similar to ratings such as "parental guidance") and is not to be confused with the ACTUAL access rights that are defined as a separate KCO element.
query	KCO-ID, "USG", "hist"	KCO-ID, Usage:Content pairs	The query returns aggregations of how the content has been used and manipulated in the past (trails, history). Note that this element has significant privacy implications and may be fully blocked depending on legislation, policy etc.
add	KCO-ID, "USG", <SubElement>, Sub-graph	OK | FAIL	If the subgraph to be added does not match the USG SubElement ontology (i.e. if at least one sub-graph label differs) then the operation fails. This is to ensure that only consistent sub-graphs are added.
delete	KCO-ID, "USG", <SubElement>, Subgraph | "all"	OK | FAIL | FAIL-LOCKED	If sub-graph does not match: FAIL If sub-graph instances are locked - FAIL-LOCKED (user may be able to "manually" delete the non-locked instances of the sub-graph)
update	KCO-ID, "USG", <SubElement>, Subgraph	OK | FAIL | FAIL-LOCKED	See above.
merge	KCO-ID1, KCO-ID2, "USG", <SubElement>, Subgraph-1, Subgraph-2,	OK | FAIL-RESOLVE	If two USG-SubElements are merged and the sub-graphs have different values attached to identical attributes then the user has to resolve the conflicts "manually", or accept failure of the operation
render	KCO-ID, "USG", TargetFormat	NULL	The render operator is not applicable to the usage context. If the knowledge contained in the USG should be rendered then the USG should first be converted into a logical description "LD".
convert	KCO-ID, <SrcElement>, <TrgElement>	OK | PARTIAL | FAIL	OK if a full conversion succeeded, PARTIAL if at least one element was converted (to be accepted/rejected by the user); FAIL if zero conversions were successful

Table 8: KCO Usage Context Operations

6.5.6 KCO and Media Business Services

Multimedia Business Services involve getting information about contracts, licenses, legal terms and pricing to conduct the actual business transaction. The following tasks are involved:

Operator	Operands / Parameters	Return Value	Description
query	KCO-ID, "BIZ", "license" | "contract" | "pricing" | "negotiation" | "trading", QueryTerm | "all",	KCO-ID, BIZ-Graphs | NULL | FAIL	The query returns the business terms which are known for this knowledge content object, or NULL if no terms are known (or if no matches are made at instance i.e. value level), or FAIL if the QueryTerm does not match the schema of the SubElement (e.g. contract schema)
add	KCO-ID, "BIZ", <SubElement>, Sub-graph	OK | FAIL	If the subgraph to be added does not match the BIZ SubElement ontology (i.e. if at least one sub-graph label differs) then the operation fails. This is to ensure that only consistent sub-graphs are added.
delete	KCO-ID, "BIZ", <SubElement>, Subgraph | "all"	OK | FAIL | FAIL-LOCKED	If sub-graph does not match: FAIL If sub-graph instances are locked - FAIL-LOCKED (user may be able to "manually" delete the non-locked instances of the sub-graph)
update	KCO-ID, "USG", <SubElement>, Subgraph	OK | FAIL | FAIL-LOCKED	See above.
merge	KCO-ID1, KCO-ID2, "BIZ", <SubElement>, Subgraph-1, Subgraph-2,	OK | FAIL-RESOLVE	If two BIZ-SubElements are merged and the sub-graphs have different values attached to identical attributes then the user has to resolve the conflicts "manually", or accept failure of the operation
render	KCO-ID, "BIZ", TargetFormat	NULL	The render operator is not applicable to the usage context. If the knowledge contained in the BIZ should be rendered then the BIZ should first be converted into a logical description "LD".
convert	KCO-ID, <SrcElement>, <TrgElement>	OK | PARTIAL | FAIL	OK if a full conversion succeeded, PARTIAL if at least one element was converted (to be accepted/rejected by the user); FAIL if zero conversions were successful

Table 9: KCO Business Context Operations

Detailed Semantics of Trading operations

Trading functionality is needed for any contract-based transfer of KCOs between individuals and/or organisations. This is an example for more specific semantics which we will have to add for other KCO elements as well.

Call	Parameters	Description
Browse	KCO-ID, <SubElement>	Retrieves various descriptions derived from the ontological representations of the KCO elements, provided they are open to browsing

Signal	KCO-ID	Buyer signals that he is interested to buy this KCO; This request is transmitted to the person that is qualified as a negotiation partner (→ business representation)
Negotiate	KCO-ID, BUYER, SELLER, NEGO-SCHEME	Buyer and / or seller select an admissible negations mechanism that was retrieved from the business ontology. This call evokes a negotiation mechanism as referenced by the KCO. [Overruling mechanisms could be required]
Sign-Contract	KCO-ID, BUYER, SELLER, CONTRACT, NOTARY	A negotiated contract can be fed into an KCO by sign-contract. This adds the contract and a reference to the notary to the legal representation of a KCO.
Exchange		Transaction of a least two operations: exchange of two values such as goods and money

Table 10: Specific Semantics of Trading Operations

6.5.7 KCO Trust Operations

KCO Trust involves assigning user confidence ratings or certification values to multimedia resources and KCOs. Trust related services will involve the following operations:

Operator	Operands / Parameters	Return Value	Description
query	KCO-ID, "TRUST", "ratingmodels" | "ratings", <M> | "certificates" QueryTerm, <M> | "all"	KCO-ID, TRUST-Graphs | NULL | FAIL	The query returns the trust terms which are known for this knowledge content object, or NULL if no terms are known (or if no matches are made at instance i.e. value level), or FAIL if the QueryTerm does not match the schema of the SubElement (e.g. ratings schema)
add	KCO-ID, "TRUST", <SubElement>, Sub-graph	OK | FAIL	If the subgraph to be added does not match the TRUST SubElement ontology (i.e. if at least one sub-graph label differs) then the operation fails. This is to ensure that only consistent sub-graphs are added.
delete	KCO-ID, "TRUST", <SubElement>, Subgraph | "all"	OK | FAIL | FAIL-LOCKED	If sub-graph does not match: FAIL If sub-graph instances are locked - FAIL-LOCKED (user may be able to "manually" delete the non-locked instances of the sub-graph)
update	KCO-ID, "TRUST", <SubElement>, Subgraph	OK | FAIL | FAIL-LOCKED	See above.
merge	KCO-ID1, KCO-ID2, "TRUST", <SubElement>, Subgraph-1, Subgraph-2,	OK | FAIL-RESOLVE	If two TRUST-SubElements are merged and the sub-graphs have different values attached to identical attributes then the user has to resolve the conflicts "manually", or accept failure of the operation
render	KCO-ID, "TRUST", TargetFormat	NULL	The render operator is not applicable to the usage context. If the knowledge contained in the TRUST should be rendered then the TRUST should first be converted into a logical description "LD".
convert	KCO-ID, <SrcElement>, <TrgElement>	OK | PARTIAL | FAIL	OK if a full conversion succeeded, PARTIAL if at least one element was converted (to be accepted/rejected by the user); FAIL if zero conversions were successful

Table 11: KCO Trust Operations

6.5.8 KCO Access and Collaboration Management

The functionality of collaboration management deals with the issues arising from individuals as members of a group manipulating KCOs. This entails questions of sharing workspaces and KCOs, versioning and access/locking.

Call	Parameters	Description
LOCK	KCO-ID	Locks a KCO to be edited only by a single process. Includes Timeout.
UNLOCK	KCO-ID	Unlocks a KCO
GET	KCO-ID	Retrieves a KCO
ADD	KCO, KCO-ID, Database	Inserts the KCO into the target database/repository
DELETE	KCO-ID, Database	Deletes the KCO

Table 12: Collaboration Operations

6.6 Knowledge Content Objects in the application domains of METOKIS

Given this initial mental image of knowledge content objects, we try to illustrate their potential use in the application domains of the project, namely the annotation and selection of media information items for senior executives; the dynamic aggregation of learning objects and the creation of a protocol for clinical trials.

6.6.1 Senior Executive Information Objects

Templeton College hosts the Oxford Retail Futures Group (ORFG). This is a group of senior executives who meet several times a year to discuss the retail industry. The group is co-ordinated by a moderator who needs to set an agenda for the group's meetings over the forthcoming year.

KnowledgeView provides a news management system which via Rapid Browser (front end for News Delivery) provides an interface for journalists to manage the news process, and provides necessary tools to deliver news to the end-user.

The moderator will use KnowledgeView's Rapid Browser to aggregate and filter retail news. A blog will be used to help define likely topics and generate feedback from the executives. Finally, the agenda itself will be published to a wiki.

KCO description of News Articles

The KCO in the Senior Executive domain consist of news articles which are shared by multiple systems: a publisher publishes news articles, the Knowledge Views system acts as an aggregator and provides enhanced services based on that. The tools like WIKI/Blog will be able to retrieve the content from Knowledge Views system and will be able to provide a feedback channel.

A single KCO will consist of multiple multimedia documents consisting of text, audio, video or documents etc. The Metadata Description of KCO will contain multimedia description of the news articles consisting of metadata such as MPEG-7, Dublin Core. The actual logic description of the multimedia will be part of the Propositional Content. It is envisaged that the richness of the description will depend on provider to provider and some providers in future can provide KCO's for news articles directly. The KCO Usage context will contain the context in which this data can be used e.g. Only for personal use or public use or can be used by aggregators etc. The KCO Business Semantics will provide the contract and copyright information (e.g. creative commons license for digital items). KCO Trust Semantics will define the quality of the News Articles e.g. the actual rating can be dependent on external rating systems. The Access Semantics will define who can access and in which fashion a particular KCO can be accessed.

Description of Tasks related to news articles (aggregation/query etc.)

Rapid Browser as a system will have input from various feeds (news wires, email, RSS blogs). These news items will be classified (by subject area, person (& their organisation), agenda topic) and then displayed to the user in filters. Alerts will highlight specific news items which match a particular classification.

Rapid Browser and the Moderator (via blog and wiki) will be able to perform certain actions on the multimedia content kept in the Knowledge View's news management system. The table below mentions some of the services which will be provided by the KnowledgeView's system to be able to interact with external systems (blog / wiki).

Agent	Call	Parameters	Description
RapidBrowser	Get-Feed	Feed ID, feed type	Acquire news feed
RapidBrowser	Classify-Story	Story ID, classification	Create classifications for a story (subject, person or topic)
RapidBrowser	Display-Alert	Filter, story, classification	Highlight a story that matches a classification
Moderator	Add-Comments	Story	Annotate a story
Moderator	Define-Topic	Story, blog	Create a topic and publish it
Moderator	Publish-Agenda	Agenda, wiki	Post the agenda item to the wiki

Table 13: Application Services provided by the news management systems for the Senior Executive domain

6.6.2 Klett Education Platform

The Educational Metokis Platform (EMPF) focuses on the production of CBTs or WBTs. The objective is to create a platform that facilitates the software production following the rapid prototyping approach. It should be able to aggregate education content in variety of formats e.g. LOM, SCORM objects etc.

KCO description of a Learning Object

A KCO in the Klett Education Domain will consist of multimedia resources stored in a repository, any business plan objects that take part in the workflow of the Educational Metokis Platform. In case of learning objects the KCO will contain the LOM or SCORM metadata. It will also contain classification of media (assets) via ontology. The different syllabuses will be tagged via an educational domain specific ontology.

Description of tasks related to Learning Objects

The tasks related to the Klett Education Domain will be querying the repositories internal or external for education resources. The Tools will enable seamless integration of result sets from multiple repositories. The querying will involve querying the resources via an ontology, doing analysis of the query results and providing results to the users based on a relevance (putting higher relevant results on the top).

Call	Parameters	Description
Query	Term, Language, MultimediaType	Searching of results based on specific terms from an ontology. Language implies the language of the text, MultimediaType

		involves type of multimedia resource(image, documents, video etc.)
Query	Text,Language, MultimediaType	Searching of resources based on natural language text.
GET-Rank	Query, Resource	Gets a rank value of a query based on the query.

Table 14: Tasks provided by the Educational Metokis Platform.

6.6.3 Clinical Trial Design Objects

The Clinical Trial Application will provide a design tool for building clinical trial protocols. The key parts of the clinical trial design tool consist of designing a clinical trial protocol, doing compliance checking with standard clinical trial procedures and to provide a visualization interface for analysis of clinical trial data.

KCO description of Clinical Trial Objects

The Clinical Trial Objects contain information consisting of the procedure and the data related to a specific clinical trial. The information contained here is more structured as compared to the information of news articles or pure multimedia documents. The Clinical Trial ontology like HL7 describes the structured content of the Clinical Trial Objects. External Templates (ontology describing the compliance of clinical trial data e.g. which data fields are mandatory , which are optional; the rules building the particular user interface so that a user fills the data in a particular sequential fashion) along with Clinical Trial Ontology enables popping of specific databases with clinical trial data. The data when shared by multiple systems is shared as a KCO.

The Propositional Content of the KCO in Clinical Trial Objects will contain the actual clinical trial data referring to the particular ontology which has been used for building the Clinical Trial Object. The KCO Usage Semantics will contain the context in which this data can be used e.g. Only for private use or for government compliance etc. The KCO Business Semantics will provide the contract and copyright information. KCO Trust Semantics will define the quality of Clinical Trial Object e.g. if it is suitable for use for government regulations etc. The Access Semantics will define who can access the Clinical Trial Object KCOs.

Description of Clinical Trail Tasks

The Clinical Trial Application works as a client server application where the server repository holds information about the clinical trials, the data of the trials, compliance rules and authorization information. The middleware provides the necessary interfaces so that external systems (ones outside the scope of the tool) can contact the repository and perform necessary functions.

The table below mentions some of the interfaces that will enable the tool as well as any other external system to communicate with the middleware and query for clinical trial protocol data, perform compliance checking or is able to extract data depending on a pre-defined template.

Call	Parameters	Description
Query	TemplateTermID	Searching of templates for clinical trial protocol based on terms of a template vocabulary
Query	DataItemID	Searching of clinical trial data based on terms of a clinical trial ontology vocabulary
Check-Compliance	ClinicalTrialID, RuleID	Checking of compliance of a Clinical Trial with a given set of Rules
Check-Completeness	ClinicalTrialID, RuleID	Checking of completeness of a Clinical Trial with a given set of Rules
Extract-Data	ClinicalTrialID, TemplateID	Extracting Data from a clinical Trial based on a given template.

Table 15: Tasks provided by the clinical trial system.

7 KCCA - Knowledge Content Carrier Architecture

7.1 Introduction

The METOKIS Architecture defines a middleware platform for building semantic information systems, providing components and services enabling interoperability amongst varied content management platforms. The infrastructure aims to work towards:

- **Content Level Interoperation**: Applications are normally built around specific document models enabling little or no interoperability. Building ontologies and publishing well defined schemas is one of the key steps towards content level interoperability.

- **Knowledge Level Interoperation:** Content Standards to some extent enable smooth transition of content from one format to another but this at times results in loss of information. Given two known different schemas, dynamic translation and transformation of instances should be possible.

- **Task level Interoperation:** Tasks define how the content is used by users and systems. Semantic definition of tasks should enable systems to "understand" i.e. recognise the constraints posed by the context in which tasks are used in a workflow.

- **Workflow & Collaborative Work Interoperation:** Specific services based on content, knowledge and tasks at both domain and application level will foster interoperation and enable collaboration.

The Metokis platform will build a middleware platform for semantic content management systems. The Metokis platform consists of two key parts: KCCA (Knowledge Content Carrier Architecture) Platform and KCTP (Knowledge Content Transfer Protocol) Protocol. The KCCA Platform acts as a middleware providing support for building content management applications. The KCTP provides interaction and communication support between multiple Metokis or Metokis enabled systems. Section 8 describes the Knowledge Content Transfer Protocol in detail.

The KCCA Platform provides the basic middleware support for exchanging Knowledge Content Objects (KCOs) and for defining operations on KCOs. It provides support for semantic definition of tasks and will also provide specific services (tasks) in the multimedia content management sector. The three application domains of the project (Clinical Trials, Education and Senior Executives in Retail Sector) will use the KCCA Platform and the KCCA Services and also provide their own application specific services to be used by Application Tools and other external systems.

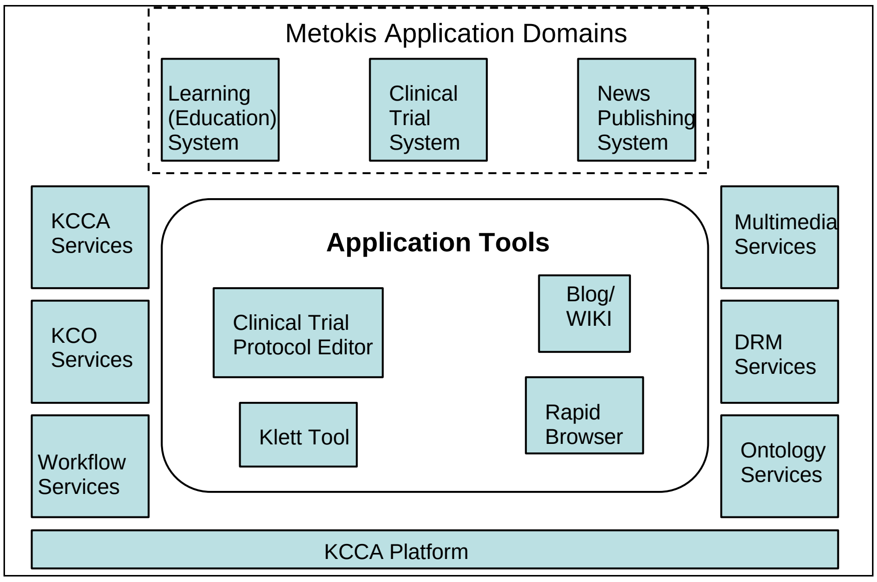


Figure 15: Overview of the KCCA Platform.

7.2 KCCA Middleware

There are four key components (as shown in the below Figure 16) of the KCCA Middleware:

- KCCA Repository
- KCCA Middleware Components
- KCCA Request Broker
- KCTP Protocol

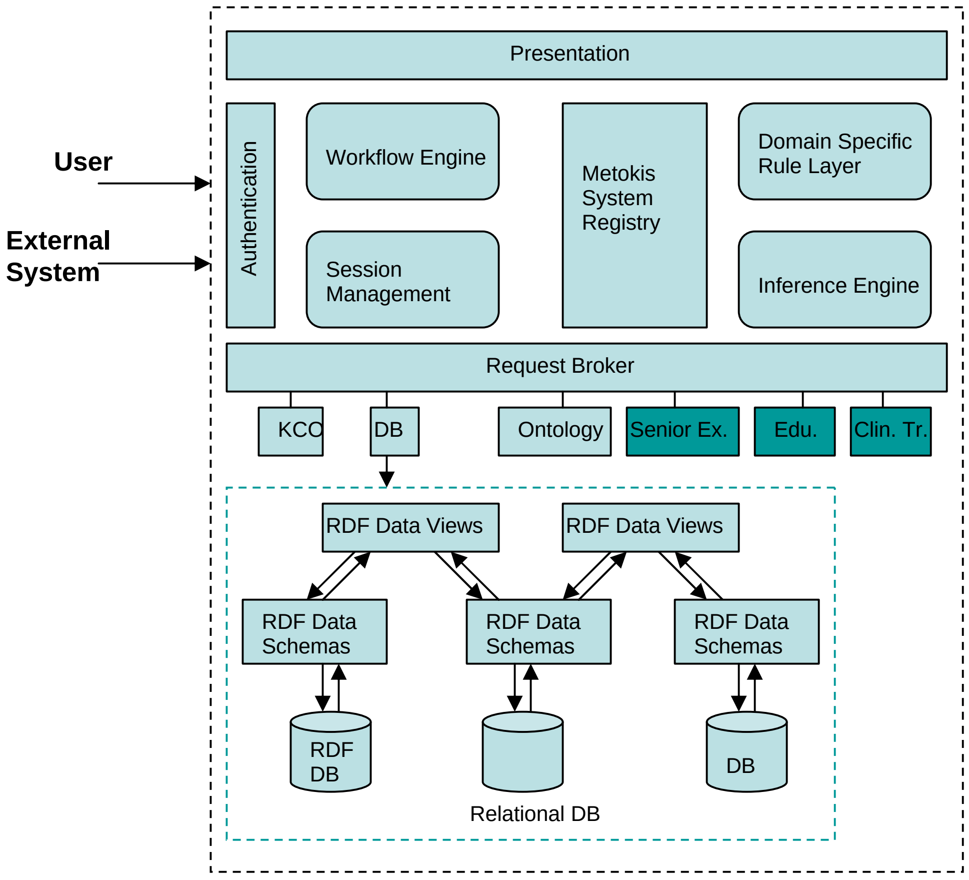


Figure 16: Overview of the KCCA Middleware

KCCA Repository: KCCA Repository provides interfaces with databases for storage of content, metadata and ontologies. It also acts as storage for KCOs (Knowledge Content Objects). The metadata within KCCA middleware is stored at RDF level and existing data within Relational or XML databases can be supported via mapping the non-RDF data models with equivalent data models in RDF. It also provides other repository functionalities such as views over data schemas or merging multiple RDF schemas/model etc.

KCCA Middleware Components: KCCA Middleware Components provide specific components and modules that enable building up of the actual middleware. The components include: Authentication, Workflow Engine, Session Management, Inference Engine, Rule Layer and System Registry.

KCCA Request Broker: KCCA Request Broker enables integration of middleware components and also provides support for both system and domain level services. The domain level services include services in the three application domains of Metokis, services related to Multimedia Systems (Digital Rights Management etc,), Registry Services etc. The system level services include services for accessing KCCA Repository, accessing KCCA Middleware components etc. It also includes KCO Services which provide access, query and manipulation of KCOs. The execution of KCCA Middleware Services is done by the Workflow Engine component within the KCCA Middleware Components.

KCTP Protocol: KCTP (Knowledge Content Transfer Protocol) Protocol provides access to KCCA Middleware to other external KCCA Systems. The KCCA Middleware system can exchange KCO's with other KCCA aware systems by a simple request/response protocol. The KCCA Middleware can also talk with other systems such as Ontology Servers etc. using the KCTP. Chapter 8 describes the KCTP in detail.

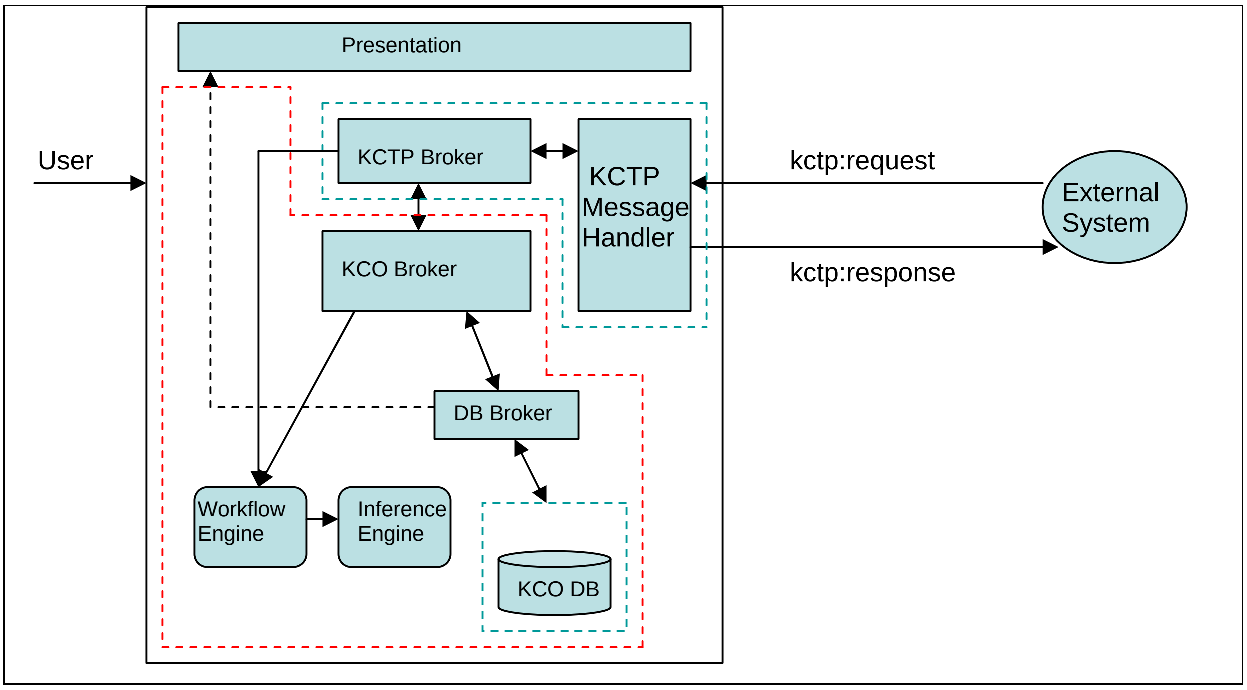


Figure 17: Request/Response within KCCA Middleware Systems via KCTP

A simple request/response protocol (Figure 17) enables exchange of KCOs within KCCA systems. Within the KCCA middleware, a KCTP Message Handler handles the KCTP message and then passes it to the KCTP Broker. The KCTP Broker interprets the message and if contains a KCO, passes it to the KCO Broker. KCTP Broker uses KCCA Middleware Components such as Workflow Engines or Inference Engines etc. to take the necessary actions. KCOs are stored within the database repository.

7.3 KCCA Repository

This section deals with how various databases, schemas & models can be integrated within the KCCA Middleware Architecture which then provides a basis for services that the Metokis System must support for interaction with the system. It provides support for data storage layer and enables external systems to interact with the storage layer. It provides storage both for KCOs as well as storage for other form of metadata/ontologies with a well defined schema in RDF.

There are wide variety of databases that are commonly used in information systems, the most popular being relational databases. Integrating heterogeneous data schema models within even a relational database via automatic schema mapping is still a huge problem. Variety of approaches (e.g. Warehouses, Mediators, Schema Mapping GAV (Global As View) /LAV (Local As View)) are practiced for integrating multiple heterogeneous data models:

Any data modeling within an information system occurs at four different levels: Instance Level, Schema Level, Data Model and Conceptual Model. Taking an example from Relational Databases, the rows of a given table (e.g. rows of an Employee table) are the instances; the Schema Model is the schema for the table (e.g. the Schema for an Employee table); the Data Model is the actual model of the database (e.g. Relational Database Model with a defined Relational Algebra) and a Conceptual Model (e.g. UML Modeling). At each of these four different levels there are heterogeneity issues and there have been a number of solutions proposed in the past at each of these four different levels.

Within the KCCA Architecture, we are pragmatic about solving generic interoperability issues at each of the above four levels. Within the KCCA Middleware we take an ontology (using OWL) as a starting point for conceptual modeling, and use RDF model as our data model for modeling the domains. The KCCA Middleware Architecture does not build its own component for automatic schema mapping between multiple Data Schemas but defines a framework in which external mapping solutions (e.g. MAFRA, D2RQ) can be plugged in.

The KCCA provides a basic infrastructure for integrating multiple heterogeneous databases. It provides the necessary interfaces for querying and updating RDF database schemas and RDF instances, thus enabling multiple database systems to be plugged in to the semantic systems world. The graph-based model of RDF provides flexibility to model other database schemas (Relational, Object Oriented, XML). The KCCA Middleware works both with RDF Databases directly (like Sesame, Redland etc), or else, uses a wrapper approach where an equivalent RDF schema for a domain model in a relational database exists and hence can work with a non-RDF database. Any existing warehouses or federated databases that need to be made available should also provide an equivalent RDF schema map from their domain model. KCCA will provide interfaces for integration of wrappers but will not deal fully with automatic schema mapping by itself. It will rely on external schema mapping frameworks ([SWIM01], [PIAZZA01], [D2RQ]) for this.

In an RDF world, domain schemas are published to the external world which is quite contrary to the traditional database world, where schemas are normally kept internal. At times sharing internal domain models is complex or is not required or is even prohibited as a matter of business principles. Therefore we take an approach where we split the internal database model explicitly from an external world-view. The approach is also used by Relational databases which enables users to define views over database schemas.

Within the architecture we split up the RDF schema into two different schema types depending on the respective roles each one plays. We call it Context Profile & View Profile.

Context Profile: Context Profile is a one-to-one mapping of an internal data model to an RDF data model. The Context Profiles will be normally kept internal within a system and might not be shared with the external world.

View Profile: View Profile is a view over one or more context profiles. The schema defined by the View Profile is the one that will be shared with the external world.

The figure below illustrates how Context Profiles and View Profiles are related by specific properties and how these relate to the notion of "Repository" as mentioned in Figure 2.

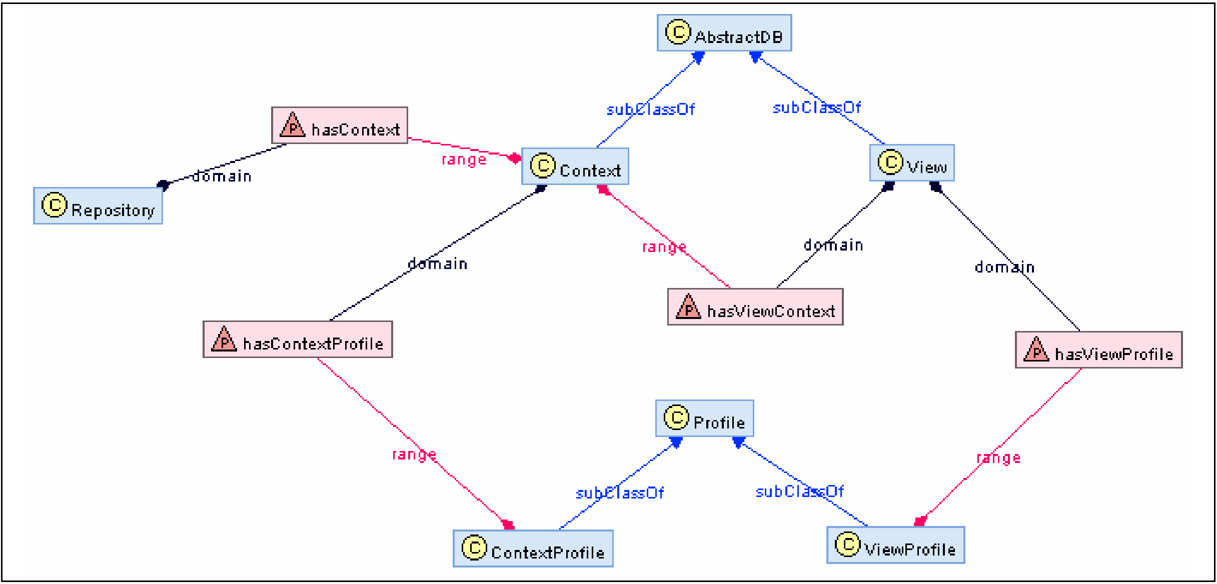
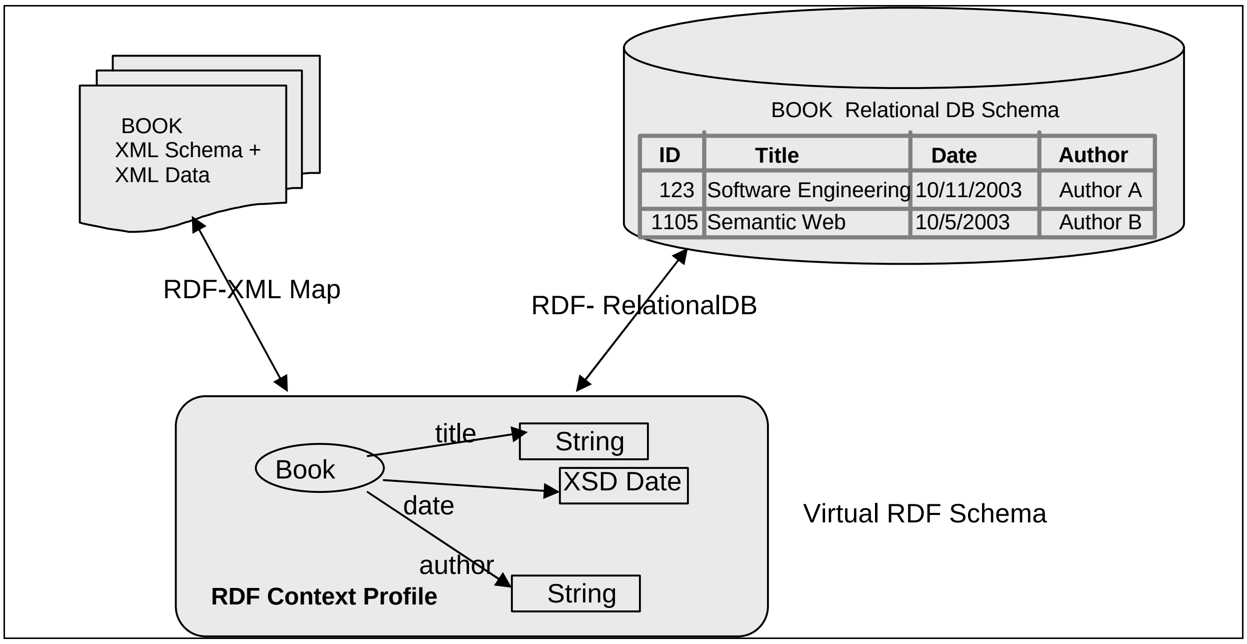


Figure 18: Notion of Context Profiles and View Profiles in KCCA.

The schema shown (in Fig. 18) is a partial schema which exists as part of the KCCA Architecture Schema in RDF/S. Labels marked "C" are concepts and Labels marked "P" are properties. The 'domain', 'range', 'subClassOf' are properties defined in RDFS.

KCCA models the notion of a database repository within the system by defining the notion of a 'Repository' within the KCCA Architecture Schema. A 'Repository' can have multiple 'Context's. A 'Context' refers to instances within a particular data schema (e.g. an Employee table holding information of employees within the context of a relational database). Every 'Context' within the KCCA system has a 'ContextProfile'. A 'ContextProfile' refers to the actual schema for the context (e.g. an Employee table schema). Views are defined in the KCCA Architecture by defining the concept of a 'View'. A 'View' within a KCCA system can be defined over one or more 'Context's. A 'ViewProfile' refers to the actual schema of the 'View'. 'AbstractDB' is an abstract superclass of 'Context' and 'View'. 'Profile' is an abstract superclass of 'ContextProfile' and 'ViewProfile'.

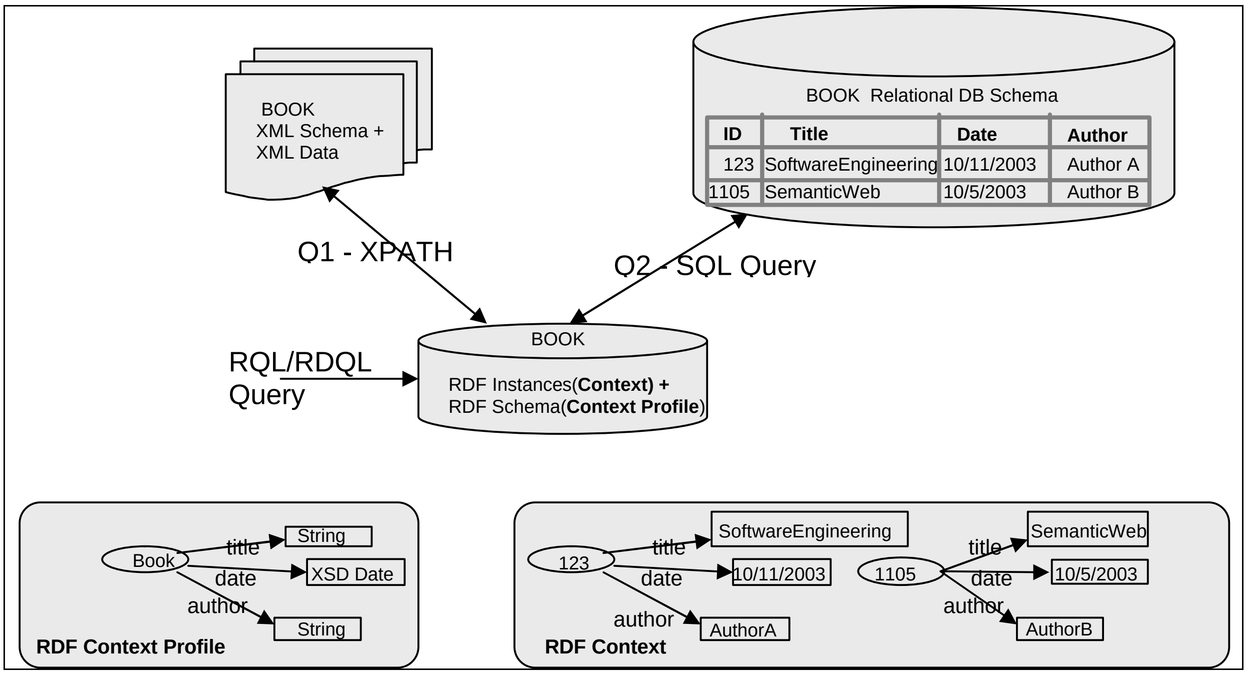
The figure below (Fig.19) describes integration of heterogeneous databases like Relational and XML Databases within the KCCA Middleware. In the example below, a relational database contains a particular schema about Books (ID, title, date and author). Similar information can also be present in an XML database. A 'ContextProfile' within KCCA defines schema of Books in RDF which is equivalent to the schema of Books defined in a XML database or a Relational database. The RDF-XML Map or the RDF-Relational Map are domain level mappings which map Books schema defined in a in XML database or a Relational Database with the 'ContextProfile' of the Books in RDF.



ID	Title	Date	Author
123	Software Engineering	10/11/2003	Author A
1105	Semantic Web	10/5/2003	Author B

Figure 19: RDF Context Profile providing a virtual map over an equivalent Relational table or an equivalent XML Schema.

In other words, the 'ContextProfile' within KCCA is a schema in RDF for an equivalent domain level schema in a Relational, XML or any other database. KCCA doesn't provide generic mappings at Relational or XML model level. A necessary binding between the 'ContextProfile' with the Relational Schema (RDF-RelationalDB Map) or the 'ContextProfile' with an XML Schema (RDF-XML Map) is necessary. Certain frameworks like SWIM, D2RQ enable building up of such mappings at schema level. D2RQ also provides query reformulation techniques where by read only RDQL (RDF Query Language) queries are transformed into equivalent SQL queries for relational databases and the SQL query results are then transformed back in RDF.



ID	Title	Date	Author
123	SoftwareEngineering	10/11/2003	Author A
1105	SemanticWeb	10/5/2003	Author B

Figure 20: Query transformation via RDF Context Profile

An RQL query is converted into equivalent SQL query for relational databases or into XPATH query for XML databases taking into account the mapping of RDF Context Profile with equivalent Relational or XML schemas. RDF Context Profile provides the RDF domain schema and RDF Context holds the actual data instances.

In systems, at times the internal data model is quite different from a domain model in terms of ontology depicting real life situations. Systems might not even want to share their internal data model with external providers. Differentiating a Context Profile and a View Profile makes this difference explicit and provides functionality that can be harnessed by external systems. This also gives the advantage that the View Profile can be closer to applications than to internal middleware systems or one can have multiple View Profiles depending on application needs. View Languages like RVL (RDF View Language in SWIM) currently define views but these do not form a core part of the middleware architecture. For simple data models, a View Profile can be the same as the Context Profile. The Figure below (Figure 21) shows a particular mapping between a View Profile and a Context Profile.

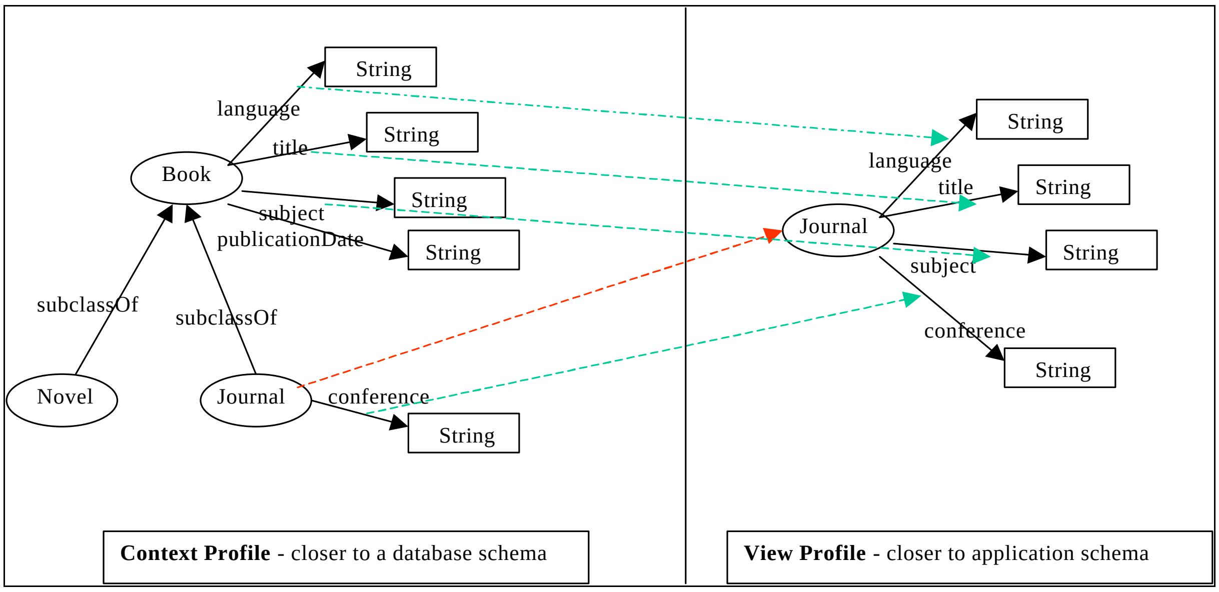


Figure 21: Context and View Profiles and their role with respect to databases and applications

A context Profile in the above figure (left hand side) represents a small model defining books, novels, journals and properties associated with them. An application which deals with just journals, doesn't need to know about novels or books or any other kind of book. The View Profile (right hand side) presents a small restricted view over the context profile defining Journals and provides a mapping between the concepts (marked with dotted red lines) and mapping between properties (marked with dotted green lines).

The Context Profile also gives an explicit advantage by having a one-to-one mapping of a data model in RDF with an internal data model kept in an external non-RDF database. This is equivalent to building canonical wrappers for integrating heterogeneous databases. Context Profile and View Profile respectively contain domain schemas with View Profiles being personalized domain schemas over Context Profiles. The notion of a Context in METOKIS is similar to the notion of models in Jena or as Graphs in TRIX [TRIX01]. The notion is also equivalent to the notion of SCAM Contexts (which describe it as an aggregation of SCAM resources). The difference is that systems like Joseki, SCAM etc. - do not distinguish explicitly between context and view as the same domain schema that is used inside the system is also shared outside with external systems.

7.3.1 KCO Storage

The KCCA Repository also provides storage for Knowledge Content Objects (KCOs). The KCO Schema acts as a View Profile within the Repository. The Context Profile for the KCO Schema is the same as the View Profile unless certain systems have any proprietary extensions to the KCO Schema. Such extensions can be supported by changing the Context Profile of the KCO and providing a suitable mapping between the Context Profile and View Profile of the KCO Schema. The particular instances of KCOs exist as 'Context' within the Repository.

7.4 KCCA Middleware Components

KCCA Middleware Components provide specific components and modules for supporting building of semantic web applications. The components include: Inference Engine, Workflow Engine, Authentication, Session Management, Registry and Domain Specific Rules etc. Most of the components exist as either stand alone components e.g. Reasoning Engines providing support for OWL, or workflow engines supporting task execution within a workflow (e.g. Business Process Execution Language (BPEL)) or as part of another web application framework (e.g. J2EE platform includes authentication and session tracking). In the section below, we discuss two of the components (Access Control & Presentation). Most of the components will be taken as off-the-shelf components and will be made available via the KCCA Request Broker.

7.4.1 Authentication

RDF itself does not inherently provide the notion of a graph or a model and only defines a restricted form of encapsulation. But systems need to have an explicit notion of models or graphs so that applications can work within the boundary of a particular model. Security & access policies need to be associated to the graph models. And systems also need to do support operations on multiple graph models - merge, combine, do reasoning, apply rule engines etc. Various approaches have been proposed to represent a model and to access resources for a model. Approaches vary from having a resource level view (URIQA concise bounded description) to aggregation of particular resources (SEAL) to a file based graph model (TRIX, JOSEKI).

Depending on the granularity of the model, the security mechanisms & access policies change. We take the approach of having a graph based granularity as an access mechanism rather than at an individual resource triples level. In case of the need for resource level security, a particular View Profile can filter a Context Profile based on security information with resource (triple) granularity defined in another Context.

KCCA does not say how an individual mapping to a model or graph is to be done. The only constraint is the views representing graphs (or models) know which resources they contain. One way of achieving this is to have an encapsulation or explicit reification. This increases the number of triples for each node present in the model. The other possibility is to have an internal model that has an explicit notion of a model or a graph.

KCCA will use a subset of Access Control Lists (ACL's) [W3CACL] for defining authorization and access control at a graph/model level for defining security & access policies.

7.4.2 Presentation

Presentation Layer in web information systems helps generate hypermedia presentations dynamically from resources stored within databases. The Web information systems retrieve information from a number of heterogeneous data sources and adapt the information according to user needs & user environment.

The presentation layer deals with issues, such as:

- Presentation Delivery - Content Format (HTML for web, SMIL, WML)
- Navigation Structure of the Presentation
- Composition of presentation resources dynamically
- User Interaction
- Personalization

There are a number of middleware frameworks & architectures providing presentation layer support for the above issues. Web Frameworks like Struts [STRUTS], Jetspeed [JETSPEED], Tapestry [TAPESTRY], Cocoon [COCOON] provide extensive support for web based hypermedia presentations and portal support. Semantic Web Frameworks such as SEAL (I & II), HERA [HERA01], OntoWebber [Jin01] etc. use ontologies to model information space and use ontologies to also define navigation, presentation and user interaction.

Within METOKIS, no specific component for presentation will be developed at the middleware level but wherever needed an existing presentation framework will be used within the context of each of the three application domains. For instance, the Clinical Trial application demands a specific user interface component which requires guided navigation based on an ontology. The presentation layer in this case will form a part of the Clinical Trial application which will interact with the KCCA middleware for retrieving information.

7.5 KCCA Request Broker

This section describes services that are made available via the KCCA Request Broker. These services form a part of the KCCA Architecture and enable external systems to query a Metokis System via using a service framework.

There are two kinds of services, one which form a core part of the KCCA middleware enabling querying, manipulating data; a registry server, mediator services and second are the domain level services (e.g. services related to Educational Content Publishing).

The KCCA System Services consist of:
Repository Services - enable data query and data manipulation of the KCCA Repository.
Session Services - provides state maintenance.
KCO Services - enable query, manipulation etc. on KCOs within a KCCA Middleware system.
Registry Services - a registry framework (e.g. like UDDI) using RDF.

The KCCA Domain level Services consist of:
Ontology Services - getting and manipulating ontologies for a KCO.
Digital Rights Management Services - support for Digital Rights Management.
Multimedia Content Management - specific services for multimedia content management.
Application Domain Services - e.g. Services related to Education, Clinical Trials and Senior Executives in Retail.

The specific request brokers corresponding to each set of services makes available domain level functionality within the KCCA Middleware.

In this section, we mainly concentrate on the KCCA System Services, which form a core part of the middleware architecture. The specific KCCA Domain level services and the specific operations needed for them have already been specified in Section 6.6.

7.5.1 Repository Services

Repository Services contains services that interact with the RDF Repository. These also form a part of the service description of the KCTP (Knowledge Content Transfer Protocol) as described in the next chapter (Chapter 8). Different repositories support variety of query languages (RDQL, SeRQL [SERQL], OWL-QL [OWLQL01], Concise Bounded Description). KCCA Middleware supports components for plugging in multiple query languages and performing query operations. The specific details for this are covered in Section 8.4 of KCTP-RDF Data Profile.

Operations on a repository can be divided into two basic services, *query* services and *data manipulation* services:

Query services:
query(View, QueryLanguage, Query) - implies querying a particular View model defined in the Repository using a specific Query Language. The View parameter refers to the specific View instance as defined in RDF Repository, the Query Language parameter refers to a particular QueryLanguage e.g. RDQL. Query is the actual query that is to be executed on the View.

Data Manipulation Services: This provides *Add*, *Delete* or *Update* operations on a repository.

add(View, QueryLanguage, Query) - modifying a RDF Data Model using a specific query language.
delete(View, QueryLanguage, Query) - deleting a RDF Data Model using a specific query language.
update(View, QueryLanguage, Query)- updating a RDF Data Model using a specific query language.

A system may or may not support both the Query Service and all operations of the Data Manipulation Services on every domain model. KCCA Middleware will provide services to interact with the domain models and know which operations are supported. Similarly a particular repository may support one or more query languages. The KCCA Middleware will enable to plug in support for multiple query languages and then provide services for querying.

Query Service & Data Manipulation will not only allow an external system to query for data in repositories but it will also provide support for higher level services such as Registry, Sessions, Encoding, Ontology Services, Rights Management Services etc.

7.5.2 Session Services

Session Services enable maintaining user state during a session. For more details on Session Profile, please see Section 8.4.2.

Session Data and User Data View Service

The Session Data View Service enables an external system or client to temporarily hold RDF Triples (RDF statements (subject, predicate, object) pairs) within a session. This is equivalent to holding cookie information (key-value pairs) in Servlets [SERVLET] or any other technology to maintain state over HTTP. A user gets back a temporary object called Session Data View (containing triples) valid for the current session on which the user can perform queries or update it.

The User Data View Service is a persistent form of Session Data View Service, in which an external system or a user application can store RDF triples in a persistent fashion inside a system. The User Data View Service returns a User Data View, which provides for instance initialization to his session whenever the user creates a new session.

View Adaptation Service

Ontology and domain schemas can become complex with hundreds of concepts and properties. Applications are normally concerned with only a small number of concepts and only a small part of the whole ontology. Even small applications need to know the complete domain schema to work on or to query a small part of this ontology. This makes life hard for application developers developing semantic web applications.

Defining an application-oriented view over the domain schema helps solve this problem to some extent. An application can define its own small view over the big domain ontology, and then use this view or schema within its internal system. A mapping between the view and the actual domain schema helps translate things between the view and the actual domain schema. View Languages like RVL (RDF View Language) provide a support for this. But most of the time the views are kept at the client and the necessary translation is done at the client side. Although this reduces the complexity within a particular client application, performing the necessary translations can still be rather complex.

The View Adaptation Service enables a client to store a view (an application specific view over a database schema) temporarily at the server for the session. During a particular session, a client can then make queries or do updates on the view held in the session. This reduces the complexity for the client as the client just concentrates on the view maintained within its application.

7.5.3 KCO Services

KCO Services provide implementation for the KCO operators. Section 6 describes the KCO tasks that are provided by the KCCA Middleware.

7.5.4 Registry Services

A registry service provider provides specific look up services. The Registry Service in METOKIS uses the existing infrastructure (consisting of KCCA Middleware and KCTP Protocol) for building look up services.

Registry Servers have a specific Registry Schema definition, which defines the data model of the Registry in RDF. So in this aspect Registry Schema is a domain schema which can be stored in a repository and one can have both Context Profile & View Profile for the registry schemas. For example there can be a Registry Schema for LDAP resources or UDDI registries at the syntactic level.

At the minimum a registry service provider needs to support the basic KCTP Protocol implying it should provide support to query about data encodings, supported protocols, any repositories (a specific registry repository) if any, query for the schemas that it has and other such services.

A Registry Server publishes the "Registry View Profile"(a View Profile of the Registry Schema) which is then made available externally for systems to query the Registry Server. Registry View Profile has an internal "Registry Context Profile" (Context Profile of the Registry Schema) which contains the system data model for storing registries. At the same time a registry service can provide specific specialized services which are specific for looking up resources.

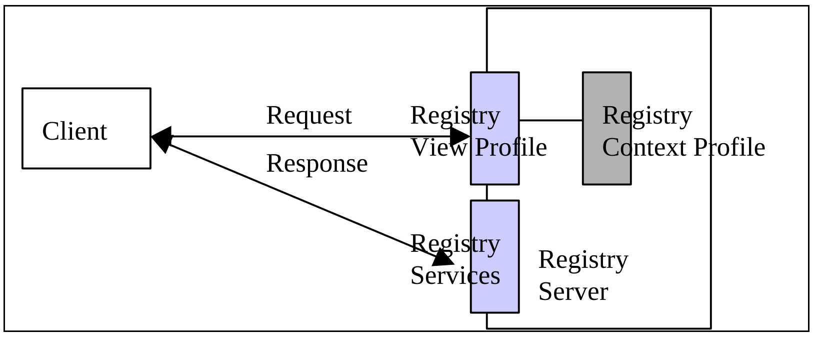


Figure 22: Registry Server & Client interaction

The KCCA Middleware along with KCTP enables the registry services to be implemented in variety of ways yet keeping the same semantic structure for querying a registry server. Registries can be built in multiple ways:

HTTP GET/POST Registry: A simple look up for resources can be provided via a minimal Registry Schema. Any client can access the Registry Server and can ask for particular METOKIS System Resources that the Registry Server knows of. A simple HTTP GET/POST request protocol acts as the binding protocol for communication.

UDDI Type Registry: UDDI based registries can be built on top of the METOKIS system where a specific UDDI View Profile and Context Profile exist and specific services available for UDDI can be published at RDF Data Profile Services. The protocol binding over multiple protocols can be provided e.g. SOAP, HTTP GET/POST etc.

Publish/Subscribe based Registry: Messaging systems can be used at the protocol layer to provide publish/subscribe based registry for resources.

Overlay Network Based Registry: In pure peer-to-peer scenarios, distributed hash tables or methods such as HyperCube routing [Schlosser02] can be used to build registry services.

8 KCTP - Knowledge Content Transfer Protocol

8.1 Introduction

Knowledge content objects enable sharing of content and knowledge across multiple platforms and systems. Multiple applications interact with these knowledge objects that hold information about complex relationships from a number of domains. The operations on knowledge content objects vary from as simple as accessing these objects, to doing complex manipulations and sharing. Different management systems from content creation applications, multimedia content management systems, content distribution systems, learning management systems - each work partially on the content objects depending on their specific functionality. KCCA - Knowledge Content Carrier Architecture via Knowledge Content Transfer Protocol (KCTP) provides a common middleware upon which multiple services providing varied functionality can be built and which will enable systems to partially interact with the knowledge content objects.

KCOs as described in Section 6, enable representation of content, metadata and knowledge by defining resource spaces, knowledge spaces and action spaces. Information systems making use of knowledge objects need to store, interpret, process, deliver and share such objects with other disparate systems. Defining and standardising content and metadata standards enable systems to process only standard metadata and the systems tend to suffer as soon as they receive something outside their domain. Apart from operating on standardised knowledge content objects, enabling partial understanding between systems is a key requirement for any semantic information system.

The primary motivation behind building KCTP - Knowledge Content Transfer Protocol is to define a standard way of sharing information. KCTP provides a description of the communication between multiple METOKIS Systems. The KCTP is defined via a small ontology expressed in RDF and is referred in the document as the KCCA Architecture Schema. Any system provider can provide his own implementation of the system taking into account the core functionality that needs to be fulfilled as described within the KCCA Architecture Schema and can also support extensions on top of it. The KCTP consists of two core parts:

KCTP Profiles: This provides a description of a protocol for sharing METOKIS system information, RDF Data, doing state oriented communication, describing tasks and services at semantic level and to share KCOs.

KCTP Request/Response Protocol: This provides an overview of the request response protocol for communication between multiple METOKIS enabled systems.

The next sections describe the KCTP Protocol in detail.

8.2 KCTP Protocol

KCCA provides a base architecture and middle-ware components for semantic web based infrastructure systems. The methodology we have used is on the principle that there are wide varieties of systems from simple to complex - and therefore any semantic web middleware should be adaptable and be able to fulfill the middleware role in multiple scenarios. KCCA focuses itself more on interfaces and requirements, which can be implemented using multiple technologies.

KCCA takes a protocol level view (KCTP) of defining an architecture where the protocol provides a standard way of sharing semantic information (see Figure 23). The KCTP Protocol consists of 2 layers the application layer and a session layer with reference to the OSI Network Reference Architecture [Tanenbaum02, p59]. KCTP via using vocabularies enables partial understanding amongst semantic middleware systems.

The KCCA architecture itself is independent of transport layer protocols and can be implemented over multiple transport layer protocols (e.g. HTTP, SOAP, Messaging etc.). The only requirement that KCCA must fulfill for KCTP is to support a request/response protocol where the requests and responses are serialized as RDF graphs. Section 8.8 provides an overview of the communication protocol.

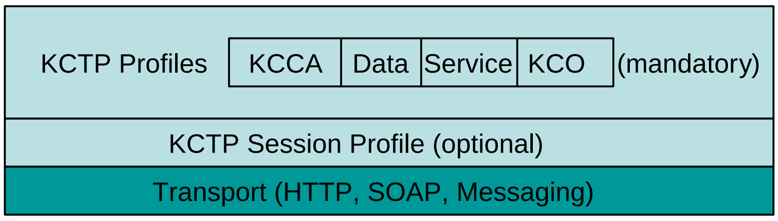

KCTP Profiles
KCCA
Data
Service
KCO
(mandatory)
KCTP Session Profile (optional)
Transport (HTTP, SOAP, Messaging)

Figure 23: KCTP - Knowledge Content Transfer Protocol layering in the OSI Layer.

KCTP enables partial understanding between multiple KCCA systems by defining KCTP Profiles which form a part of the KCCA Architecture Schema. Each KCTP Profile consists of a vocabulary (ontology) which provides a particular set of functionality within the KCCA system. A particular KCCA System provides an implementation of the KCTP in accordance to the defined semantics within each of the KCTP Profiles. The KCTP Session Profile is optional to be implemented by the KCCA middleware systems.

KCTP Profiles

The KCTP deals with the following profiles:

KCTP- KCCA Profile: KCCA Profile defines the KCCA System characteristics such as protocol bindings, data encodings, repositories etc.

KCTP-RDF Data Profile: This profile provides support for stateless operations for querying and updating data between multiple systems. The controlled vocabulary for the RDF Data Profile enables systems to Query Data, Update Data etc.

KCTP-Service Profile: This profile provides vocabulary support for service definition. It provides support for system services such as Data Manipulation Services, Registry Services, Collaboration Services, Digital Rights Management Services etc. Domain specific services belonging to the three application domains of Metokis: Education, Clinical Trials and Senior Executives also form a part of the service profile vocabulary.

KCTP-KCO Profile: The KCTP-KCO Profile enables sharing of KCOs within multiple METOKIS Systems. KCO Ontology with its defined set of tasks as defined in Section 6 forms a part of the KCTP-KCO Profile.

KCTP-Session Profile: This layer provides support for operations where state is necessary to be maintained across a session. This is useful in reasoning systems, workflow scenarios etc.

8.3 KCTP - KCCA Profile

The KCTP protocol enables multiple METOKIS systems to communicate with each other. KCTP itself is lightweight requiring at minimum a basic request/response protocol of RDF serializations.

The KCCA Profile provides the vocabulary that defines a particular KCCA system node and the functionalities (transport protocol, data encodings, repositories, services) supported by that particular KCCA Middleware system.

Within the KCCA Architecture Schema, a METOKIS system is represented by a 'MetokisSystem' concept. This refers to a particular system instance which holds the information and from which an external system can query for system information. Every METOKIS system has an access URI which enable external systems to access a particular METOKIS System node.

KCCA itself is independent of any particular protocol or data encoding. It does not provide any infrastructure for translating between multiple protocols (for instance translating IIOP calls to HTTP or vice versa). These protocols if needed can be plugged in to this architecture with the system providers supporting the basic request/response of RDF serializations on top of it.

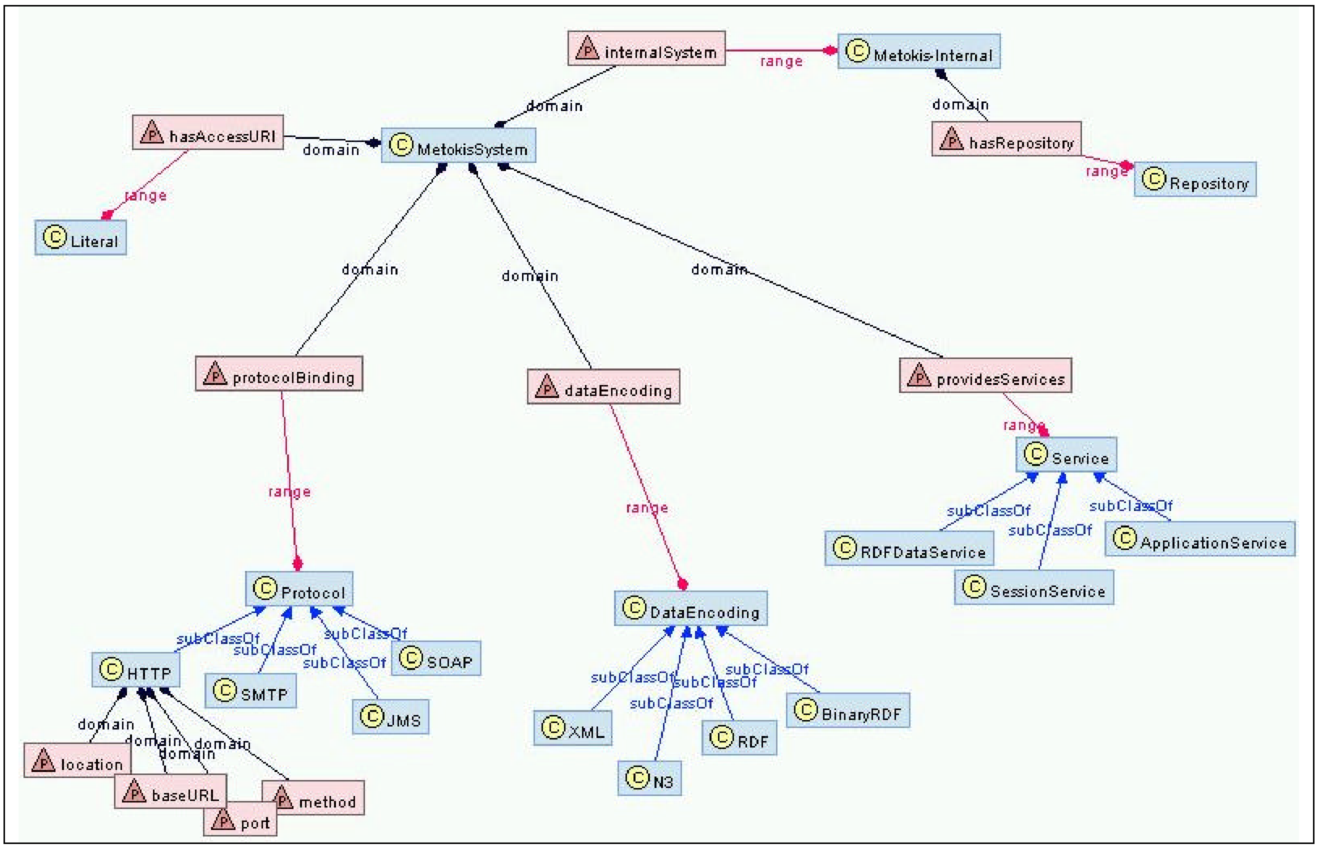


Figure 24: KCTP - KCCA Profile. Partial Schema of the KCCA Architecture Schema (in RDF/S) specifically showing data encodings, protocol bindings, services & internal properties of the KCTP Protocol. Note that all nodes (label "C") are concepts and all nodes (with label "P") are properties.

The particular protocol bindings that a system supports are referred by a 'protocolBinding' property. It specifies which particular protocol bindings the system currently supports. The particular data encodings that a METOKIS System supports are referred by the 'dataEncoding' property, which specifies which data encodings the system supports. A system can add specific data encodings and protocol bindings and provide a specific implementation for the included data encodings and protocol bindings. External systems can query a particular METOKIS System Node via a default encoding (RDF) and a default protocol binding (HTTP) to know what particular support exists for the system.

Any METOKIS System internally might have repositories or other specific services, user policies etc., which are represented via the 'Metokis-Internal' concept. Repositories within a system are represented via the 'Repository' node and a property 'hasRepository' links this to the 'Metokis-Internal' node. The 'KCTP- Data Profile' deals with operations on Repositories defined within this profile (KCTP - KCCA Profile). The 'KCTP - Service Profile' support is referenced as 'Services' within the KCTP-KCCA Profile. Services referring to RDF Data Profile, Session Profile and other Application specific Services are all referred to as Services.

8.3.1 Data Encoding

The KCCA Architecture Schema defines particular Data Encodings such as RDF, N3, XML, Binary RDF (e.g. RDF encoded in proprietary Binary format) by defining them as particular concepts and making these concepts subclasses of 'DataEncoding' concept. Every system has to support RDF data encoding as a mandatory requirement. A particular Metokis System can support one or more of these data encodings or define its own data encodings alongside the mandatory RDF data encoding. The only requirement for defining one's own data encoding is to extend the KCCA Architecture Schema by defining a concept referring to the 'DataEncoding' concept and then providing an equivalent implementation for the encoding.

8.3.2 Protocol Binding

The KCCA Architecture Schema also defines particular Protocol Bindings such as HTTP, SOAP, SMTP, JMS (Java Messaging Server) by defining them as particular concepts and making these concepts subclasses of 'Protocol' concept. The specific properties of the protocol are supported within the KCCA Architecture Schema. For example the HTTP protocol has the following properties associated with it:
method - GET/POST method of the HTTP request
baseURL - the base URL of the HTTP request
location - the location path of the HTTP request
port - the port number to which the HTTP request is sent.

The HTTP protocol is a mandatory requirement for the Metokis System. A particular Metokis System can support one or more protocols. As in the case of data encodings, to support multiple protocol bindings, the system implementers need to define which protocol bindings they want to support and provide implementations for these.

8.4 KCTP - RDF Data Profile

The KCTP - RDF Data Profile enables the integration of the RDF Repository (as described in Chapter 7) with the KCTP protocol. The RDF Data Profile enables management of an RDF Repository and enables support of query operations on the RDF Repository within a Metokis System.

The KCCA Architecture Schema integrates particular Query Languages such as Concise Bounded Description, RDQL, SQL by defining them as particular concepts. A METOKIS system can support specific Query Languages by declaring them as concepts with the KCCA Architecture Schema. This will then be available by external systems to query so as to find out which Query Languages a particular METOKIS System Node supports. Every Metokis System by default supports getting resources by Concise Bounded Description and RDQL. The KCCA Architecture Schema also defines what kind of Query Operations are possible on a particular repository (Delete, Add, Update, Query).

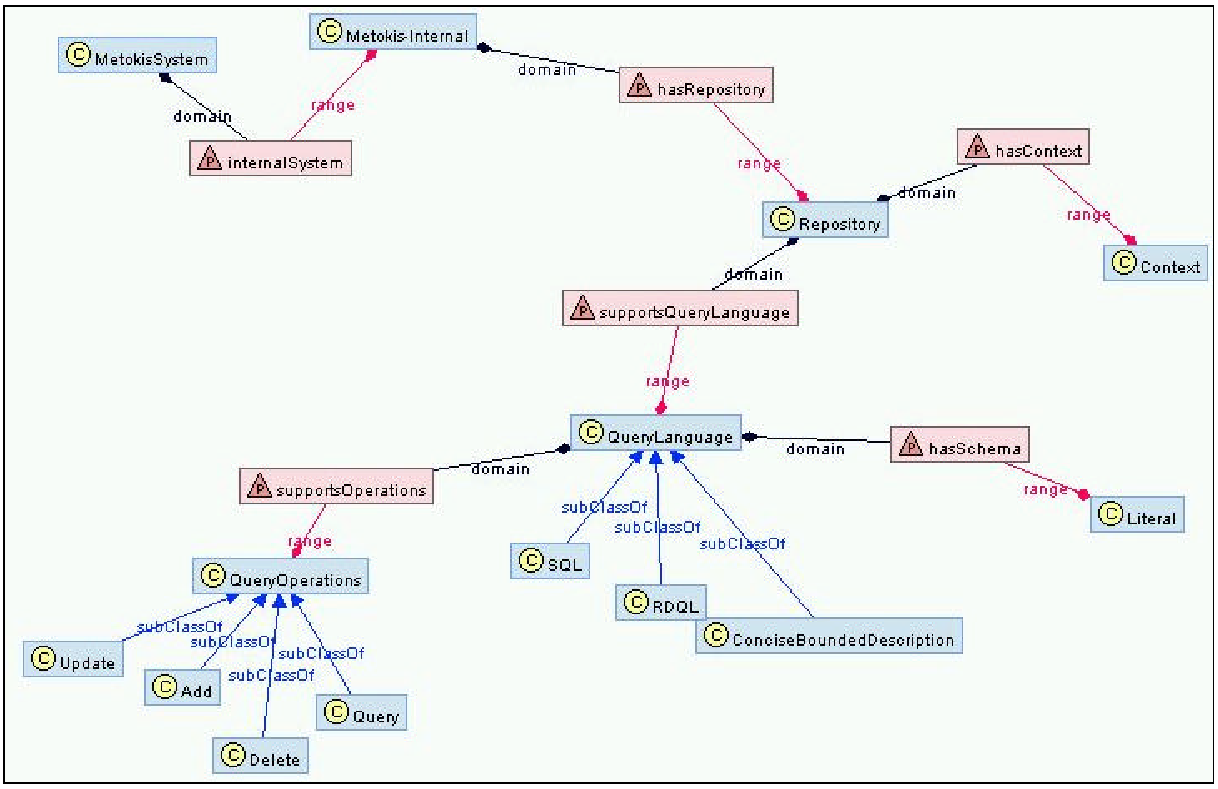


Figure 25: KCTP- RDF Data Profile. A part of the KCCA Architecture Schema showing support for multiple Query Languages and Query Operations that can be performed on a particular Repository.

8.4.1 Query Languages

Multiple Query Languages (RQL, RDQL, RDF Query Language, OWL-QL) exist for querying RDF databases. KCCA Architecture Schema doesn't define its own specific Query Language but enables the multiple query languages developed, to be used within the KCCA framework. All these multiple query languages can be represented in the KCCA Architecture Schema by extending the "Query Language" concept and providing an equivalent software implementation of the supported Query Language. KCCA Architectural Schema represents the specific Query Language Schema (e.g. RDQL Schema in RDF) represented in RDF to be represented via the "hasSchema" property. This will allow systems to query and obtain information about specific Query Languages supported by the system.

8.4.2 Query Operations

Query Languages vary in respect to the type of operation they perform. The variation exists from being a pure query language (read only queries) to data manipulation operations (modifying the data model via a Query). A repository can allow or disallow query functionality by specifying which functionality a particular Query Language supports. This enables external applications to know weather a system supports data manipulation or just read only queries.

8.5 KCTP - Service Profile

This section contains specific service profiles which METOKIS middleware provides using the KCCA Architecture Schema providing extension to the KCTP protocol for building semantic applications at services level. A semantic middleware architecture provides support for interaction with ontology management servers, multimedia content repositories, trading for knowledge content objects, workflow tasks, resource look up and mediation with other such systems.

Any specific domain services for example the three application scenarios in Metokis - Clinical Trials, Senior Executives in retail sector & Education service schemas also form a part of the Service Profiles. The RDF Data Profile provides a mechanism for querying and manipulating data operations. Service Profile layer in KCTP will help provide support at the Services layer. It will provide support for specific services for querying databases, session management, security policies, transactions or workflow management on which the domain specific services (services related to Education, Clinical Trials, Senior Executives in Retail) can be built upon.

There are two types of Service Profiles, *System Profiles* and *Domain Specific Profiles*:

KCTP System Profiles

KCTP System Profiles contains profiles for Metokis System Services. This will provide services for Registry, Repository (Query Service, Data Manipulation Service), Session Services, Data Encoding Services and Administration Services. The KCCA middleware will provide support for these system services.

KCTP Domain Specific Service Profiles

Domain Specific Service Profiles will include service definitions for specific domains such as Digital Rights Management etc. It will also include services for the three Metokis domains - Clinical Trials, Education Domain & Senior Executives.

Below we define some services (related to querying KCTP properties), which external systems can use to query the METOKIS System in order to retrieve specific information related to KCTP itself. The services below are currently envisaged to be implemented.

Methods or Services needed to interact with the Metokis System:

getDataEncodings() - getting Data Encodings (RDF/XML) supported by the Metokis System
getProtocolBindings() - protocol bindings (HTTP/SOAP) supported by the Metokis System
getAllRepositories() - getting all Repository Descriptions of repositories which this Metokis System can connect to.

The KCTP Service Profile provides services defined in an ontology to be able to discover and execute such services. Each of the services needed by the KCCA Middleware can be mapped to actual service definitions either in OWL-S or WSMO (Web Services Modeling Ontology) or DOLCE Task Model with alignment to OWL-S.

The example below provides a mapping for the method getDataEncodings() as an OWL-S description. The method returns all supported data encodings provided by the Metokis system. It takes an input parameter the URI for the Metokis system. Within the OWL-S description 'GetDataEncodings()' refers to the method to be executed. It takes the 'MetokisSystemURI' as an input parameter. It returns supported data encodings as a collection. The collection of data encodings is referred to as 'DataEncodingBag'. The OWL-S description of a service (as described in section 3.10.2) consists of three parts: the service profile description, the process model description and the grounding information. The 'KCCAGetDataEncodingsProfile' describes the service profile and the 'KCCAGetDataEncodingsModel' describes the process model for the 'GetDataEncodings()' service.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:owl=              "http://www.w3.org/2002/07/owl#"
  xmlns:rdfs=            "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf=              "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:service=     "http://www.daml.org/services/owl-s/1.0/Service.owl#"
  xmlns:process=     "http://www.daml.org/services/owl-s/1.0/Process.owl#"
  xmlns:profile=     "http://www.daml.org/services/owl-s/1.0/Profile.owl#"
  xmlns:grounding=       "http://www.daml.org/services/owl-
s/1.0/Grounding.owl#"
  xml:base=            "http://metokis.salzburgresearch.at/kcca.owl">
<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://metokis.salzburgresearch.at/kcca.owl"/>
</owl:Ontology>
<!-- Service description -->
<service:Service rdf:ID="KCCAGetDataEncodingsService">
<service:presents rdf:resource="#KCCAGetDataEncodingsProfile"/>
<service:describedBy rdf:resource="#KCCAGetDataEncodingsModel"/>
<service:supports rdf:resource="#KCCAGetDataEncodingsGrounding"/>
</service:Service>
<!-- Profile description -->
<profile:Profile rdf:ID="KCCAGetDataEncodingsProfile">
<service:isPresentedBy rdf:resource="#KCCAGetDataEncodingsService"/>
<profile:serviceName xml:lang="en">KCCA Get Data Encodings System
Service</profile:serviceName>
<profile:textDescription xml:lang="en">
This service provides get data encodings for KCCA Metokis system URI.
</profile:textDescription>
<profile:hasInput rdf:resource="#MetokisSystemURI"/>
<profile:hasOutput rdf:resource="#DataEncodingBag"/>
</profile:Profile>
<!-- Process Model description -->
<process:ProcessModel rdf:ID="KCCAGetDataEncodingsModel">
<service:describes rdf:resource="#KCCAGetDataEncodingsService"/>
<process:hasProcess rdf:resource="#KCCAGetDataEncodingsProcess"/>
</process:ProcessModel>
<process:AtomicProcess rdf:ID="KCCAGetDataEncodingsProcess">
<process:hasInput rdf:resource="#MetokisSystemURI"/>
<process:hasOutput rdf:resource="#DataEncodingBag"/>
</process:AtomicProcess>
<process:Input rdf:ID="MetokisSystemURI">
<process:parameterType
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:label>Metokis System URI</rdfs:label>
</process:Input>
<process:Output rdf:ID="DataEncodingBag">
<process:parameterType rdf:resource="#DataEncodingBagType"/>
```

```
<rdfs:label>Collection of Data Encodings</rdfs:label>
</process:Output>
<!-- Grounding description -->
<grounding:WsdlGrounding rdf:ID="KCCAGetDataEncodingsGrounding">
<service:supportedBy rdf:resource="#KCCAGetDataEncodingsService"/>
<grounding:hasAtomicProcessGrounding
rdf:resource="#KCCAGetDataEncodingsProcessGrounding"/>
</grounding:WsdlGrounding>
<grounding:WsdlAtomicProcessGrounding
rdf:ID="KCCAGetDataEncodingsProcessGrounding">
<!-- grounding information to be provided -->
</grounding:WsdlAtomicProcessGrounding>
</rdf:RDF>
```

Figure 26: OWL-S description of Get Data Encodings service 'KCCAGetDataEncodingsService'

The 'KCCAGetDataEncodingsProcess' is modeled as an atomic process. The actual grounding information consists of a WSDL binding to the OWL-S description and has been left out from the current example.

8.6 KCTP- Session Profile

KCTP - Session Profile provides the necessary support primitives for maintaining the state of RDF Data Profile or Service Profile operations, if necessary. The Session Profile enables a particular Metokis system implementing the KCTP Protocol to represent and maintain the user session and make use of that state to execute stateful services.

The Session Profile Schema (as a part of the KCCA Architecture Schema) provides a limited vocabulary for modeling session contexts. It can be extended depending on specific functionality needed by a Metokis system.

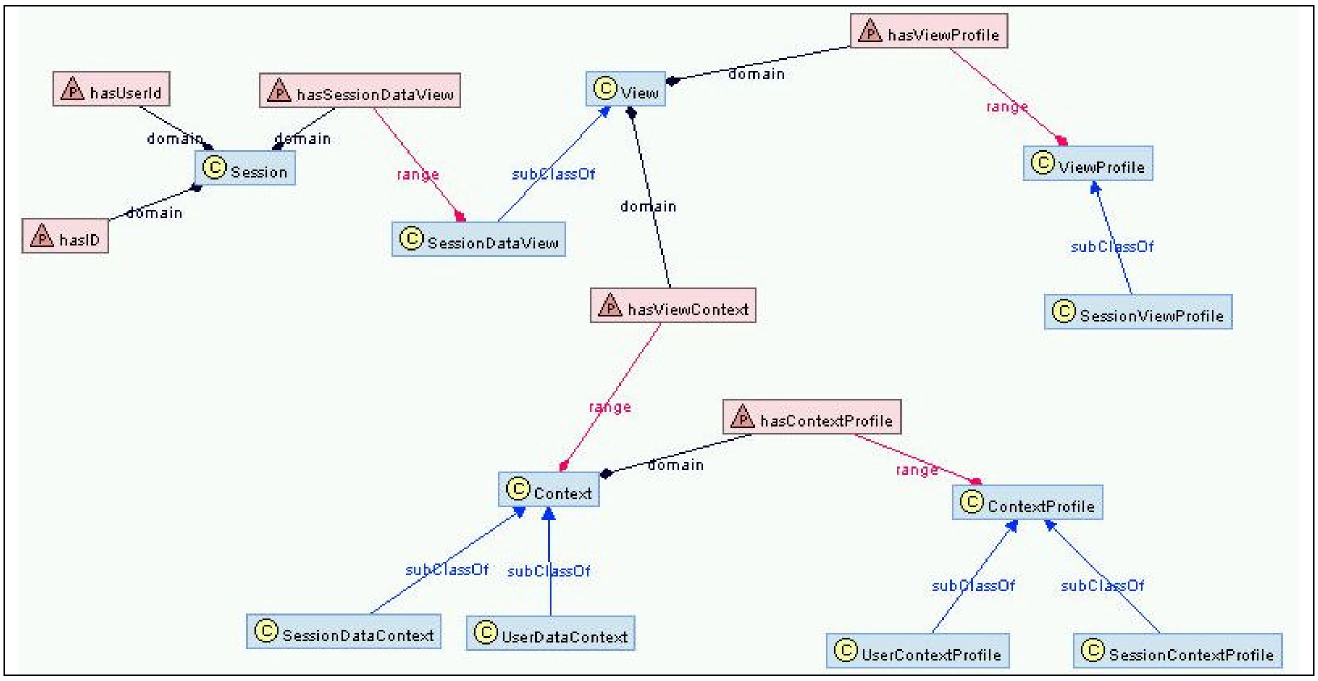


Figure 27: KCTP- Session Profile

The Session Profile consists of a concept "Session". Every Session has an associated ID called the Session ID with it. Session also includes a user id via the "hasUserID" property. The Session allows a user to store a temporary view called "SessionDataView" on the server for the session. This is done via "hasSessionDataView" property where a session can hold triples for the current user session. SessionDataView is a subclass of View.

A View has normally a ViewProfile by "hasViewProfile" property. SessionViewProfile is subclass of ViewProfile and holds a specific view profile for SessionDataView. SessionContextProfile and UserContextProfile are both subclasses of ContextProfile and hold contexts related to sessions. Every View has one more Contexts via "hasViewContext" property. SessionDataContext and UserDataContext are both subclasses of Context and provide specific contexts related to sessions.

SessionDataContext enables a system to store triples temporarily within a session. This is similar to storing cookies (key-value pairs) via Servlets. UserDataContext is a persistent form of SessionDataContext, which initializes the SessionDataContext specific to a user session whenever a new session is created for the user. UserDataContext adds the ability to store user specific Data across the duration of a single User Session. This is useful for the implementation of customized services for a registered User.

Based on the Session Profile vocabulary, a Metokis system will provide specific services for building state-oriented services. Session Profile enables a client application to store session state over multiple requests. A client application not only just needs to query or update data but at times it can store specific rules or logic statements within temporary models in a session and do reasoning on the same model over multiple requests. Session Profile also enables applications to merge multiple schemas temporarily, maintain them temporarily within a session and then do operations as needed.

State Maintenance is also useful in cases where particular result sets can be quite huge in number. Some query languages like OWL-QL enable getting partial results via maintaining partial state maintenance at the server.

8.7 KCTP - KCO Profile

The KCTP - KCO Profile provides a binding for the KCCA/KCTP infrastructure which enables systems to share KCOs. The KCO ontology along with its task definitions will provide the necessary primitives for the KCTP-KCO Profile. These will govern the interaction of KCOs within multiple KCCA Middleware systems.

8.8 KCTP Request/Response Protocol

The KCTP Request/Response Protocol defines a simple request-response protocol in which both the request and the response is serialized in RDF. The RDF Request/Response protocol enables systems to query a METOKIS System for specific information. The KCTP Profile data is serialized as RDF statements to be then interpreted by particular KCCA Middleware systems.

Below we describe a request-response protocol to query a particular system for information at the KCTP- RDF Data Profile layer. The system allows multiple query languages encoded as RDF in the request protocol to be performed. In the request format below, we mention a simple request format, which enables to query and update RDF triples kept from another system.

8.8.1 Request Format

A sample request query is shown below. The below request query is also available via a URI.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:vCard='http://www.w3.org/2001/vcard-rdf/3.0#'
      xmlns:info="http://somewhere/peopleInfo#"

xmlns:metokis='http://www.salzburgresearch.at/metokis/2004/01/metokis-
schema#'>
<rdf:Description about="http://www.salzburgresearch.at/metokis/query#q1">
<metokis:askMethod
rdf:resource="http://www.salzburgresearch.at/metokis/2004/01/metokis-
schema#ADD"/>
<metokis:modelURI>http://www.salzburgresearch.at/metokis/view1</metokis:model
URI>

<metokis:triples>
 <rdf:Description rdf:about="http://www.salzburgresearch.at/JakobSmith/">
    <vCard:FN>Jakob Smith</vCard:FN>
    <info:age>20</info:age>
    <vCard:N rdf:parseType="Resource">
      <vCard:Family>Smith</vCard:Family>
      <vCard:Given>Jakob </vCard:Given>
    </vCard:N>
  </rdf:Description>
</metokis:triples>
</rdf:Description>
</rdf:RDF>
```

Figure 28: This sample request format adds (metokis:askMethod is Add) particular triples (defined in metokis:triples) to a model with the uri "http://www.salzburgresearch.at/Metokis/view1". The Add operation is referred by the uri "http://www.salzburgresearch.at/metokis/2004/01/metokis-schema#ADD".

Request Methods

- **metokis:askMethod :**The 'askMethod' specifies the functionality that defines the necessary action, which needs to be taken by the system. The notion of 'ask' is used in the sense of an ask/tell protocol where 'ask' resembles a 'request' and 'tell' resembles a 'response' back from the system.

The fundamental operations are:
Add – This implies adding triples or RDF data to a model.
Query – This implies retrieving triples from a model.
Delete – This implies deleting set of triples from a model.
Update – This implies updating triples to a model.
Merge – Given two RDF graphs, the merge operator will add the two graphs as a non-redundant union (i.e. the semantic overlap is where graph 2 is added to graph 1). Obviously, the precise semantics of merge will have to be further defined depending on what type of object we merge. The main motivation for the merge operator is for fusing KCOs. An initial semantics for merging of KCOs has been given in Section 6 (KCO Model).

The following three tags specify a simple approach to query basic sets of triples within a model.

- **metokis:outgoingEdges :** The value is true or false depending on if outgoing edges are to be included in a query for retrieving triples associated with a resource.

- **metokis:incomingEdges :** The value is true or false depending on if incoming edges are to be included in a query for retrieving triples associated with a resource.

- **metokis:followLinks :** The value if set true implies one to follow incoming or outgoing edges recursively.

If both metokis:outgoingEdges and metokis:followLinks are set to true, then one obtains a concise bounded description of a resource.

- **metokis:modelURI :** A RDF resource can occur in more then one model within a system. A metokis:modelURI needs to be specified when querying for a resource to query the right model. The URI will be the URI of the View (as specified in Section 7.3) that a KCCA System makes available to the external systems.

- **metokis:triples :** This specifies a set of triples that needs to be added, removed or updated from within the model.

- **metokis:matches :** This value if set to true implies a resource to be matched against a case insensitive string. This allows one to search for triples based on string matching, rather than checking if an explicit resource exists or not.

8.8.2 Response Format

A sample response according to the above request is shown below. The below response format tells the system that the request was performed successfully.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:metokis='http://www.salzburgresearch.at/metokis/2004/01/metokis-
schema#'>
<rdf:Description
about="http://www.salzburgresearch.at/metokis/queryresponse#q1">
<metokis:requestURI
rdf:resource="http://www.salzburgresearch.at/metokis/query#q1" />
<metokis:response
rdf:resource="http://www.salzburgresearch.at/metokis/2004/01/metokis-
schema#Success"/>
</rdf:Description>
</rdf:RDF>
```

Figure 29: This sample response for the request query described in the previous figure.

- **metokis:response :** The response code specifies weather the request is successful or not. Some query languages don't distinguish between zero results and a particular query being not supported. If a particular query for getting all triples with dc:author is made and the system doesn't support dc:author, then a response has to be a different rather than no results. Metokis:response tag should be able to distinguish such response codes.
- **metokis:requestURI :** This contains reference to the query for which this response has been generated. Queries in a metokis system should be accessible via a URI.
- **metokis:triples :** This contains the resulting triples set returned after executing the query.

8.9 Conclusion

KCTP provides a basic protocol for sharing content and knowledge amongst multiple disparate systems. KCTP via multiple profiles enables varied degree of functionality to be implemented as demanded by the middleware systems. Any such functionality provided by the KCTP has to be defined by proper instantiation of the KCCA Architecture Schema and must be known to external systems via simple querying the Metokis System. KCTP provides a minimal protocol, which can be extended from simple basic data query and data manipulation operations to build stateful systems or to build an extensive semantic services based infrastructure.

9 KCCA/KCTP System

9.1 KCCA/KCTP System

The Metokis middleware architecture (KCCA + KCTP) will provide an infrastructure for semantic web information systems. A particular Metokis node does not need to provide or build the entire functionality but a subset of it. At the protocol layer (see Section 8.3), it can provide one of HTTP GET/Post, SOAP or may be both and similarly for data encodings.

Every Metokis system will need to implement a particular subset of the functionality depending on the particular requirement. For instance if a particular Metokis node, does not need a session service, then it does not need to provide an implementation for that.

We propose to have different levels of KCCA implementations depending on particular set of functionality the system requires.

KCCA Lite - A minimal version of the KCCA architecture and KCTP Protocol providing functionality for the RDF Data Profile layer (Section 8.4.1) with certain services from the Services Profile. This does not provide any support for sessions. It will provide a HTTP GET/POST protocol with RDF encodings for communication. KCCA Lite will be suitable for web based scenarios, where applications do not require much infrastructure support.

KCCA Standard - KCCA Standard provides more mature functionality with session support on top of KCCA Lite. It will work on multiple data encodings and will support protocols such as SOAP. It will provide some support for basic system and domain services. KCCA Standard will require some additional infrastructure support and will be usable by more mature web based systems and at the same time will provide some basic support for commercial organizations.

KCCA Full - KCCA Full will provide a much mature functionality with both system and domain services. It will have additional services for Registry, Ontology Management, Rights Management etc. KCCA Full is targeted towards mature semantic web platforms and systems within organizations.

The next section describes the KCCA - Lite system which provides the basic support for exchanging and working with KCOs over KCCA /KCTP infrastructure. The sections 9.3 to 9.6 describe the domain use cases and their interaction with the KCCA/KCTP infrastructure based on KCCA - Lite.

9.2 KCCA - Lite

KCCA Lite provides a light-weight implementation which enables web based systems to provide a middleware implementation of the METOKIS Services. KCCA Lite consists of the following key components:

1) KCCA Middleware

- **KCCA Repository:** Supports the Context and Views as defined in KCCA Architecture Schema. Mapping between Relational to RDF or from XML to RDF is not a mandatory part of KCCA-Lite.
- **KCCA Middleware Components:** Reasoning Engine for RDF/OWL.
- **KCCA Services:** KCTP Services and KCO Services (as mentioned below).

2) KCTP Protocol

- **Data Encoding:** uses RDF/XML as the data encoding.
- **Protocol:** uses HTTP as the binding protocol
- **RDF Data Layer:** uses RDQL & Concise Bounded Description as the Query Language. It supports "Query" as the Query Operations.
- **RDF Session Layer:** KCTP Lite does not provide implementations for session management.
- **RDF Service Layer:** It provides support for the basic services related to DataEncoding, Protocol and RDF Data Layer operations. It also includes services for the KCCA Repository with querying context profiles and view profiles.

For example:
getDataEncodings() : Provides description of supported Data Encodings
getProtocolBindings() : Provides description of supported Protocol Bindings
getQueryLanguages() : Provides description of supported Query Languages
getQueryOperations() : Provides description of supported Query Operations

3) KCO KCCA Binding: The KCO KCCA Binding consists of:

- **RDF Data Layer for KCO:** The RDF Data Layer for KCO consists of the KCO ontology as mentioned in Section 6
- **RDF Service Layer for KCO:** The RDF Service Layer for KCOs consists of the KCO operations. These operations will be implemented by the KCCA- Lite implementations to be able to exchange KCOs.

The above components provide a base implementation for the KCCA-Lite infrastructure which enables multiple systems to share Knowledge Content Objects. The above components are the mandatory components for any KCCA-Lite infrastructure. On top of KCCA-Lite other specific domain services etc. can be added. The table summarizes the specification of a minimal implementation of KCCA Lite.

System	Architecture Component	Architecture Sub-component	Description - Minimal Standard	Section Number
KCCA - Lite	KCCA Middleware	KCCA Repository	Supports Context and Views as defined in KCCA Architecture Schema. Mapping between Relational to RDF or from XML to RDF is not mandatory.	7.3
		KCCA Middleware Components	Reasoning Engine for RDF/OWL, Workflow Engine for Task Execution, Authentication	7.4
		KCCA Request Broker	A Request Broker including a system registry for the KCCA System.	7.5
	KCTP Protocol	KCTP - KCCA Profile	Supports the KCCA Architecture Schema for querying Metokis System information. Data Encoding - RDF/XML Protocol Binding - HTTP	8.3
		KCTP - RDF Data Profile	Query Language: RDQL and Concise Bounded Description Query Operations: Add, Delete, Query and Update.	8.4
		KCTP - Service Profile	Services are described in OWL-S. KCCA Services (*getDataEncodings(), getProtocolBindings() etc.*) KCCA Repository Services (*getContextProfiles(), getViewProfiles() etc.*) KCCA Domain Services (*Senior Executives, Education, Clinical Trials*)	8.5
		KCTP - KCO Profile	KCO Profile with KCO operations (tasks)	6.4, 6.5

Table 16: Overview of KCCA-Lite System

The following three sections outline how we envisage the three application cases to make use of the METOKIS Knowledge Content Carrier Infrastructure.

9.3 Senior Executive Information Systems

Templeton College hosts the Oxford Retail Futures Group (ORFG). This is a group of senior executives who meet several times a year to discuss the retail industry. The group is co-ordinated by a moderator who needs to set an agenda for the group's meetings over the forthcoming year. The use cases for setting this agenda are as follows:

- identify retail events
- choose retail topics
- choose the speakers
- publish the agenda

The Senior Executives Retail News System (as shown in Fig. 30) based on the KVIew RAPID Browser Server, acts as the content repository which aggregates news from various sources (e.g. RSS news feeds, NewsML, email etc.). A Rapid Browser client connected to the RAPID Browser server enables the moderator to define filters, create topics etc.

For the specific domain of Senior Executives, RAPID Browser Server will provide a means by which retail news feeds are chosen and classified. The eventual aim is to make use of a mini Retail Ontology (dealing with concepts in Retail sector) and a generic ontology about people, events, organization containing information specific to the Retail Sector. By making use of this ontology, the KView news management system will be able to annotate news according to the concepts defined in the Retail Ontology.

A moderator using Rapid Browser will be able to browse and classify news, personalize them by setting keywords and collect news related to the retail sector. This will enable the moderator to be updated about news from this sector (e.g. any new mergers and acquisitions in the retail sector or specific new technologies of relevance to the retail sector, etc.). The communication between RAPID Browser Server and RAPID Browser client includes KCTP, with data being shared in RDF.The Moderator posts agenda topics to a blog and other users send comments about the topics. Finally, the moderator publishes the agenda by exporting it from RAPID Browser Server to the blog or to a wiki.

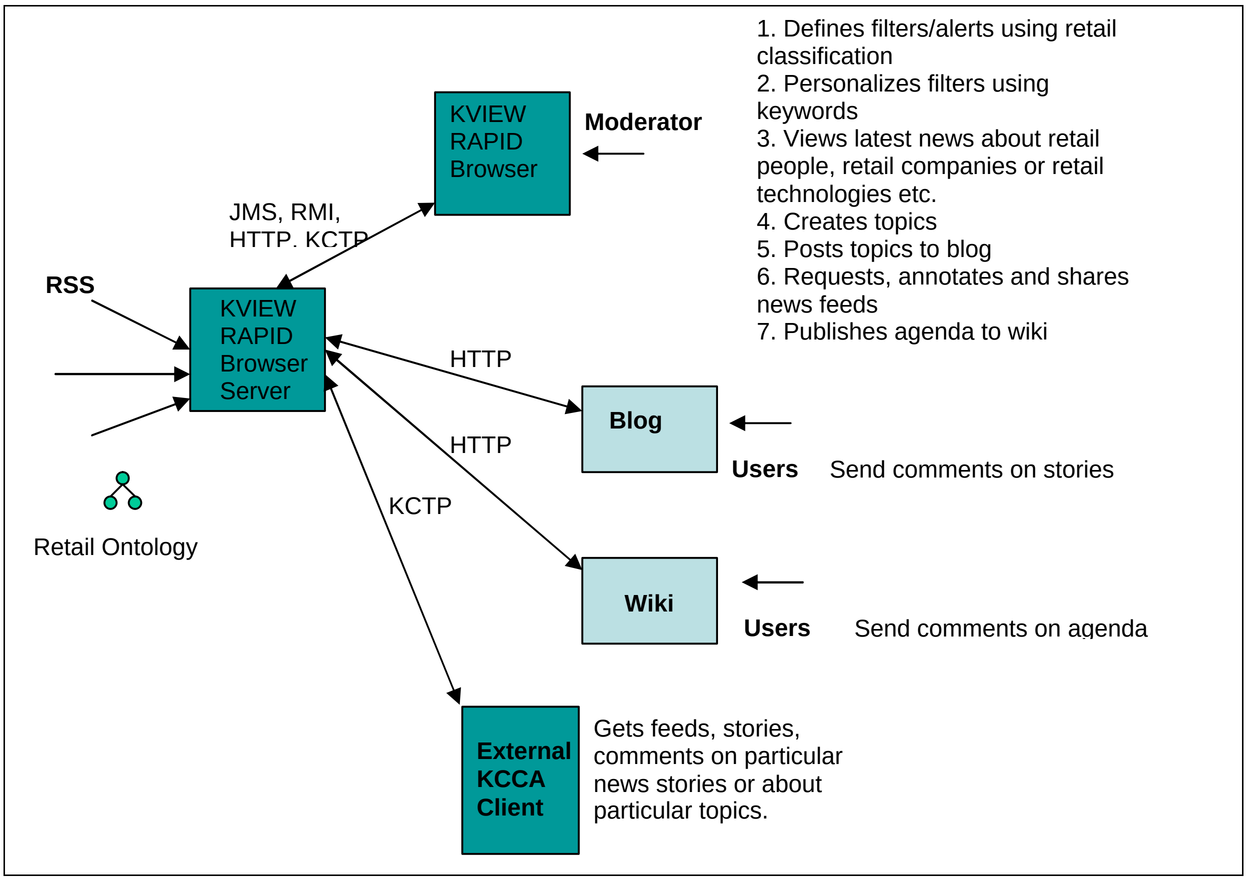

1. Defines filters/alerts using retail classification
2. Personalizes filters using keywords
3. Views latest news about retail people, retail companies or retail technologies etc.
4. Creates topics
5. Posts topics to blog
6. Requests, annotates and shares news feeds
7. Publishes agenda to wiki
KVIEW RAPID Browser
Moderator
JMS, RMI, HTTP, KCTP
RSS
KVIEW RAPID Browser Server
Retail Ontology
HTTP
Blog
Users
Send comments on stories
HTTP
KCTP
Wiki
Users
Send comments on agenda
External KCCA Client
Gets feeds, stories, comments on particular news stories or about particular topics.

Figure 30: Interaction of KnowledgeView system within the KCCA Infrastructure

Any external KCCA System should be able to query RAPID Browser Server and extract information about published services e.g. retail feeds and news stories. In order for KnowledgeView's RAPID Browser Server to participate in the KCCA/KCTP infrastructure, it needs to provide the necessary external interfaces. This can be done by providing a KCTP-RDF Data Profile layer for the data which RAPID Browser Server wants to share externally. Furthermore, it will also provide KCOs which will encapsulate domain specific information (e.g. Feeds, Stories, Comments etc.) which can be shared with other systems.

The figure below shows a simple schema consisting of concepts: Feeds, Stories, Comments, Topics, Persons and the relationships between them.

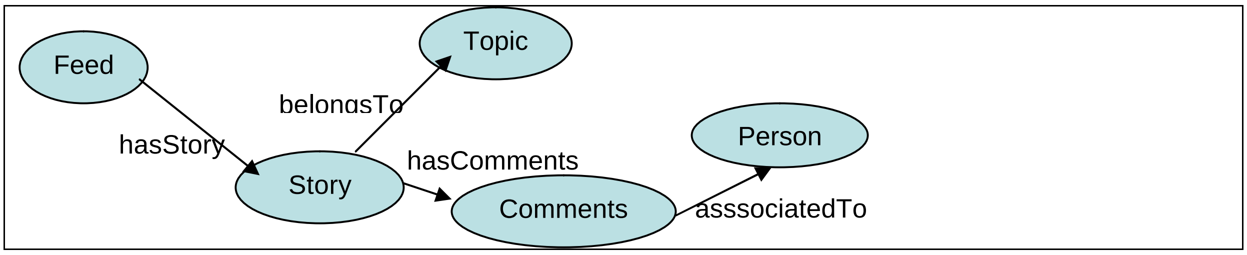


Figure 31: View Profile of RDF Data Layer within KnowledgeView system

By making the above schema (View Profile) available at the RDF Data Layer in RAPID Browser Server, any external system will be able to query RAPID Browser Server using query languages such as RDQL via KCTP to get information about news feeds etc.

The KnowledgeView Server will also be able to provide KCO's containing news stories which can then be shared with external systems.

The KView system will also publish certain services (described in OWL-S / WSMO) and an equivalent implementation via HTTP GET/POST. This will enable external systems to directly make use of such services to extract or change data in RAPID Browser Server. A description of the "KViewFeedsService" which extracts KCOs for a given feed id is represented in OWL-S. The grounding information for the service is not shown.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:owl=              "http://www.w3.org/2002/07/owl#"
  xmlns:rdfs=             "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf=              "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:service=    "http://www.daml.org/services/owl-s/1.0/Service.owl#"
  xmlns:process=    "http://www.daml.org/services/owl-s/1.0/Process.owl#"
  xmlns:profile=    "http://www.daml.org/services/owl-s/1.0/Profile.owl#"
  xmlns:grounding=      "http://www.daml.org/services/owl-
s/1.0/Grounding.owl#"
  xml:base=           "http://metokis.salzburgresearch.at/kcca.owl">
<owl:Ontology rdf:about="">
<owl:imports rdf:resource="http://metokis.salzburgresearch.at/kcca.owl"/>
</owl:Ontology>
<!-- Service description -->
<service:Service rdf:ID="KViewsGetFeedsService">
<service:presents rdf:resource="#KViewsGetFeedsProfile"/>
<service:describedBy rdf:resource="#KViewsGetFeedsModel"/>
<service:supports rdf:resource="#KViewsGetFeedsGrounding"/>
</service:Service>
<!-- Profile description -->
<profile:Profile rdf:ID="KViewsGetFeedsProfile">
<service:isPresentedBy rdf:resource="#KViewsGetFeedsService"/>
<profile:serviceName xml:lang="en">KCCA Get Feeds
Service</profile:serviceName>
<profile:textDescription xml:lang="en">
```

```
This service provides gets feeds from the KViews system.
</profile:textDescription>
<profile:hasInput rdf:resource="#FeedID"/>
<profile:hasOutput rdf:resource="#KCOBag"/>
</profile:Profile>
<!-- Process Model description -->
<process:ProcessModel rdf:ID="KViewsGetFeedsModel">
<service:describes rdf:resource="#KViewsGetFeedsService"/>
<process:hasProcess rdf:resource="#KViewsGetFeedsProcess"/>
</process:ProcessModel>
<process:AtomicProcess rdf:ID="KViewsGetFeedsProcess">
<process:hasInput rdf:resource="#FeedID"/>
<process:hasOutput rdf:resource="#KCOBag"/>
</process:AtomicProcess>
<process:Input rdf:ID="FeedID">
<process:parameterType
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
<rdfs:label>ID for the Feed</rdfs:label>
</process:Input>
<process:Output rdf:ID="KCOBag">
<process:parameterType rdf:resource="#KCOBagType"/>
<rdfs:label>Collection of KCOs</rdfs:label>
</process:Output>
<!-- Grounding description -->
<grounding:WsdlGrounding rdf:ID="KViewsGetFeedsGrounding">
<service:supportedBy rdf:resource="#KViewsGetFeedsService"/>
<grounding:hasAtomicProcessGrounding
rdf:resource="#KViewsGetFeedsProcessGrounding"/>
</grounding:WsdlGrounding>
<grounding:WsdlAtomicProcessGrounding
rdf:ID="KViewsGetFeedsProcessGrounding">
<!-- grounding information to be provided -->
</grounding:WsdlAtomicProcessGrounding>
</rdf:RDF>
```

Figure 32: getFeedKCO(feedid) expressed as an OWL-S Service. It returns collection of KCOs for a given 'feed id'. In the above figure, 'KCOBag' acts as a collection of KCOs.

9.4 E-Learning & Content Aggregation

The E-learning domain focuses on building an Educational Metokis Platform (EMPF) for the production of CBTs or WBTs. The objective is to create a platform that facilitates the software production following the rapid prototyping approach. The normal publishing workflow consists of the following key parts:

- Project Sketch
- Project Planning
- Realisation
- Production
- Archiving
- Sales/Distribution

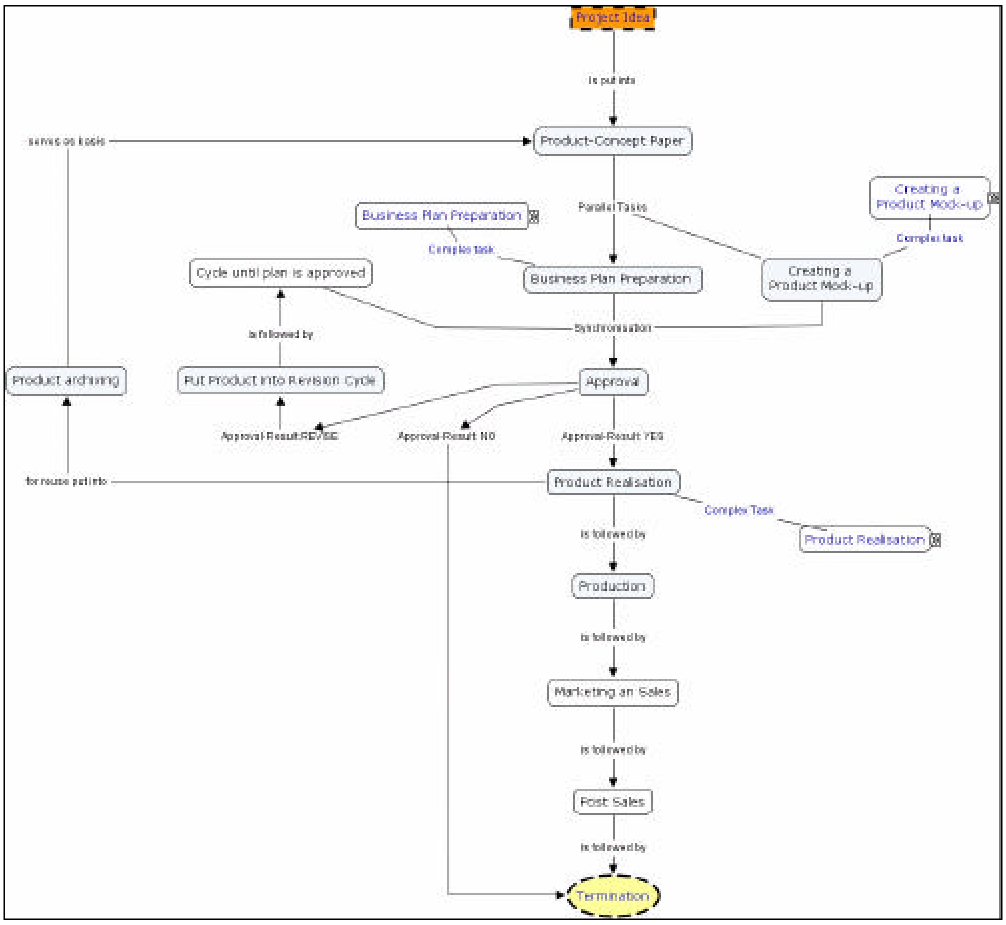


Figure 33: Workflow of the EMPF.

The EMPF will cover the first three phases, the project sketch, the project planning and the realisation and it has to facilitate the archiving, as the knowledge that is contained in the archived software shall be the starting point for following software productions. The knowledge is contained in the software at various levels and its reuse may decrease conceptualisation and realisation efforts remarkably.

The first important step in the EMPF workflow is the creation of a business plan. The business plan lays the cornerstones of the management of the following phases of the workflow. In addition to the business plan, the EMPF enables the PM to easily create a demonstrator of the “new software”. This demonstrator serves the PM to illustrate the conceptualisation of the software, and on the other hand it serves the decision makers to judge various aspects of the software on top of alphanumeric version of a business plan. The various actors (Authors, Editors, Controllers, Project Manager, Assistant, Advertising Consultant, Multimedia Editor etc.) play various roles in the workflow.

All elements of software that has been created using the EMPF are stored as KCOs. KCOs will include information about the production of an e-learning product: including information from the conceptual idea, management practices, market research, cost estimation, etc. It will also enable storage of metadata for the creation, edition and tagging of multimedia data with content and structure. KCOs will include information on the educational domain i.e. syllabuses etc. included as in the LOM standard. The figure below explains the access mechanism for the Klett Design Tool and other parts of the EMPF System interact with each other and the KCCA.

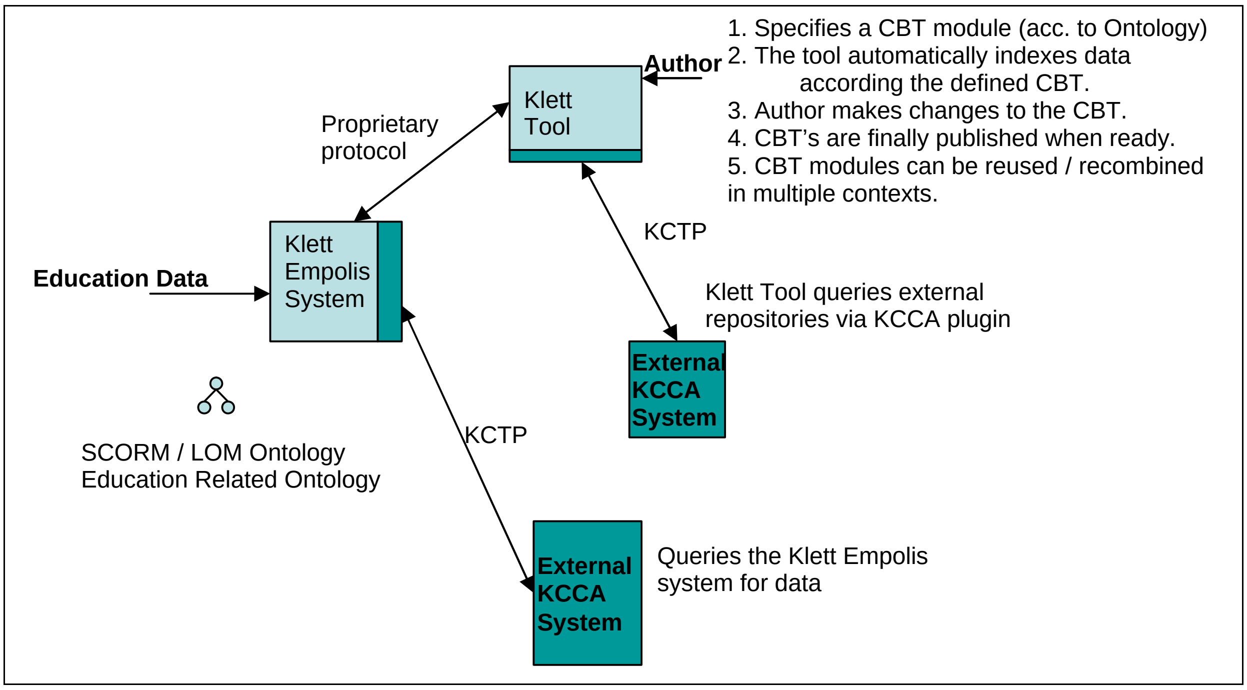

1. Specifies a CBT module (acc. to Ontology)
2. The tool automatically indexes data according the defined CBT.
3. Author makes changes to the CBT.
4. CBT's are finally published when ready.
5. CBT modules can be reused / recombined in multiple contexts.
Author
Klett Tool
Proprietary protocol
KCTP
Klett Tool queries external repositories via KCCA plugin
Education Data
Klett Empolis System
External KCCA System
SCORM / LOM Ontology
Education Related Ontology
KCTP
External KCCA System
Queries the Klett Empolis system for data

Figure 34: Interaction of Klett-Empolis system within KCCA Infrastructure.

The Authoring and Design Tools in EMPF work on Knowledge Content Objects (KCOs) kept within the system (locally) or in external repositories. The EMPF interacts with the KCCA middleware to access external collections of KCOs kept in external repositories. Authors, Designers and Managers participate as users within the EMPF at various stage to design a knowledge product (content) which is then published as a finished product.

The Klett system participates in the KCCA infrastructure by sharing the knowledge of CBT's that are designed with the Klett Design Tool with external systems. The Klett-Empolis system provides a RDF Data Layer with the data schema against which external systems can query for CBT modules. The Klett-Empolis system also provides CBT's as KCOs (CBT encapsulated as a KCO).

The Klett-Empolis System will also provide the services as mentioned in Section 6.6.2 to enable other systems to query its system for CBT. Along with this the Klett Design tool will be able to query for content both from its local repository as well as read KCO's from external repositories via the KCCA/KCTP infrastructure.

9.5 Visual modelling of Clinical Trials

The Clinical Trial Application in Metokis involves the building of a clinical trial protocol design tool. The tool will enable users or organizations dealing with clinical trials to design a clinical trial protocol, enter corresponding data relevant to the protocol; check the compliance of the data with the clinical trial procedures and to finally provide a visual interface for analysis of clinical trial data.

The figure below describes the process of designing and verifying a clinical trial design protocol.

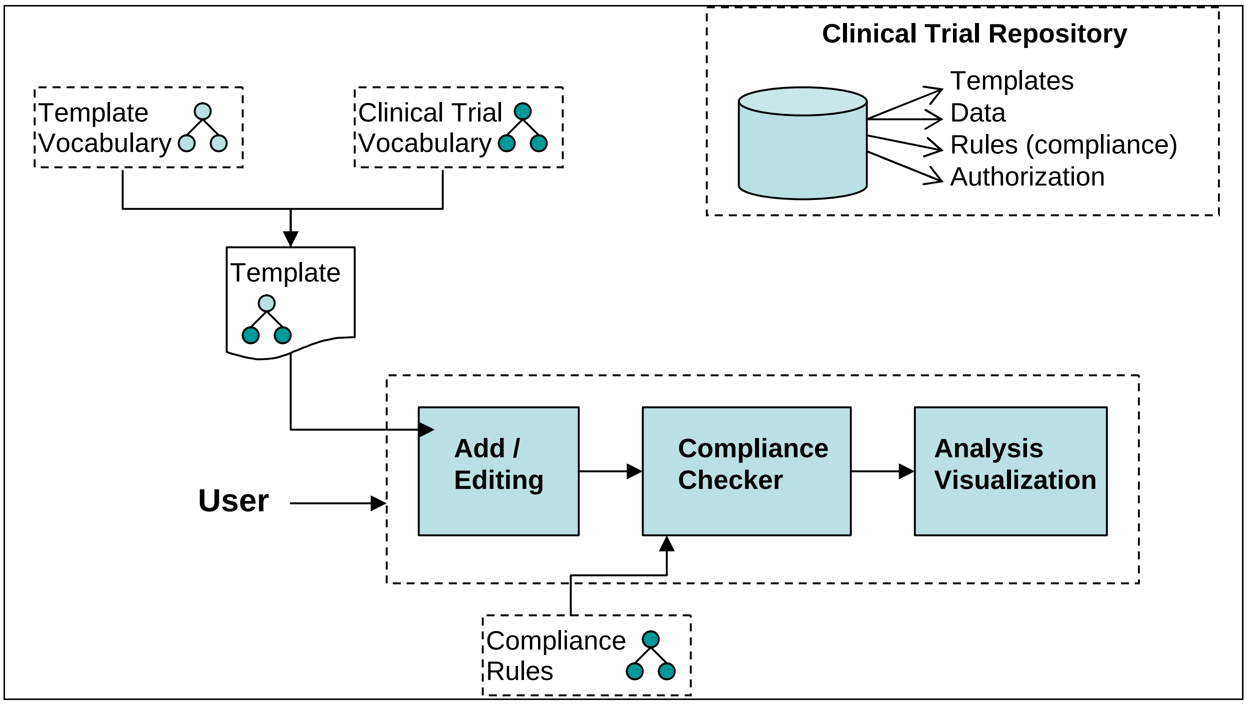


Figure 35: Overview Clinical Trial Protocol Design Application.

The specifications of the protocol are part of the Clinical Trial Vocabulary. The Template Vocabulary provides a generic ontology which enables the design tool to enforce rules for navigation, enabling filling of mandatory fields etc. The Template (Template Vocabulary + Clinical Trial Vocabulary) acts as a set of ontology rules which govern the presentation and data rules enabling user to add specific data for a particular clinical trial protocol.

The Clinical Trial Design Tool is operated by a user, where for a given template corresponding to a particular clinical trial protocol, the user can add or edit data specific to a clinical trial protocol. The second step is to do compliance checking where a user can check whether the clinical trial complies to standard procedures, or check for the completeness of the trial definition.

The third stage of the tool involves visualization and analysis of the clinical trial.

The Clinical Trial Application works as a client server application where the middleware holds information about the clinical trials. The Repository contains information about - the necessary templates, the data of the trials, compliance rules and authorization information. The middleware also provides particular services mentioned in Section 6.6.3 which enable external systems to query the middleware and perform operations. External Tools could use compliance service to check for compliance of their own specific clinical trial built via another tool. All elements of the Clinical Trial Protocol will exist as KCOs within the clinical trial repository.

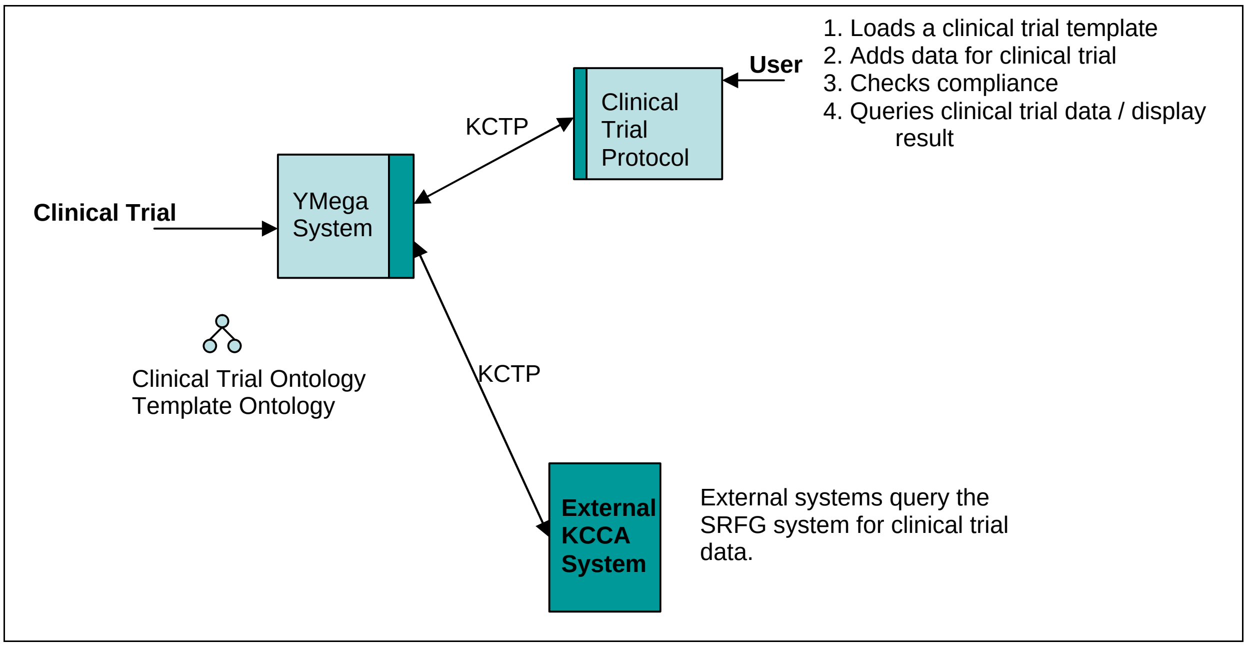

1. Loads a clinical trial template
2. Adds data for clinical trial
3. Checks compliance
4. Queries clinical trial data / display result
User
Clinical Trial Protocol
KCTP
Clinical Trial
YMega System
Clinical Trial Ontology
Template Ontology
KCTP
External KCCA System
External systems query the SRFG system for clinical trial data.

Figure 36: Interaction of Clinical Trial System in the KCCA Infrastructure

Any external KCCA System should be able to query the YMega System and extract information about published services e.g. clinical trial data. The YMega system can participate in the KCCA/KCTP infrastructure by providing the necessary external interfaces. This will be done by providing an RDF Data Layer for the data which the Ymega System wants to share externally. It will also provide KCO's which will encapsulate Clinical Trial data and will be shared across external systems.

The YMega System will also provide implementation of certain services, for instance "Check-Compliance" as mentioned in Section 6.6.3.

9.6 Cross Domain Use Cases

There are three application systems - one for each of the application domains: the KnowledgeView News Management System, the Empolis Educational Metokis Platform and the YMega Clinical Trial System. Each of these systems supports the base KCCA/KCTP infrastructure and in principle, can share KCOs between these systems.

For the validation of the generic value of KCCA, we are constructing a cross-domain use where the value of exchangeable knowledge content objects can be demonstrated.

For example, The Knowledge View system will be able to query the Empolis Educational Metokis Platform to query for CBT modules or multimedia items contained within the CBT KCOs. Similarly the Klett Design Tool via the KCTP will be able to query the Knowledge View system to get the latest news e.g. about research into history as input to some specific course on German history. In some of the cases like in Clinical Trials, the clinical trial system will be able to query the KViews system for news KCOs, e.g. from the pharmaceutical sector. The clinical trial system will be able to analyse the news KCO and interpret the basic properties such as copyright or track the source but since the retail news doesn't hold of importance to clinical trial application, the clinical trial system will not process further such KCOs.

We will attempt a three-way cross-domain use case, but we will start by developing bi-lateral cross-domain use cases as outlined above.

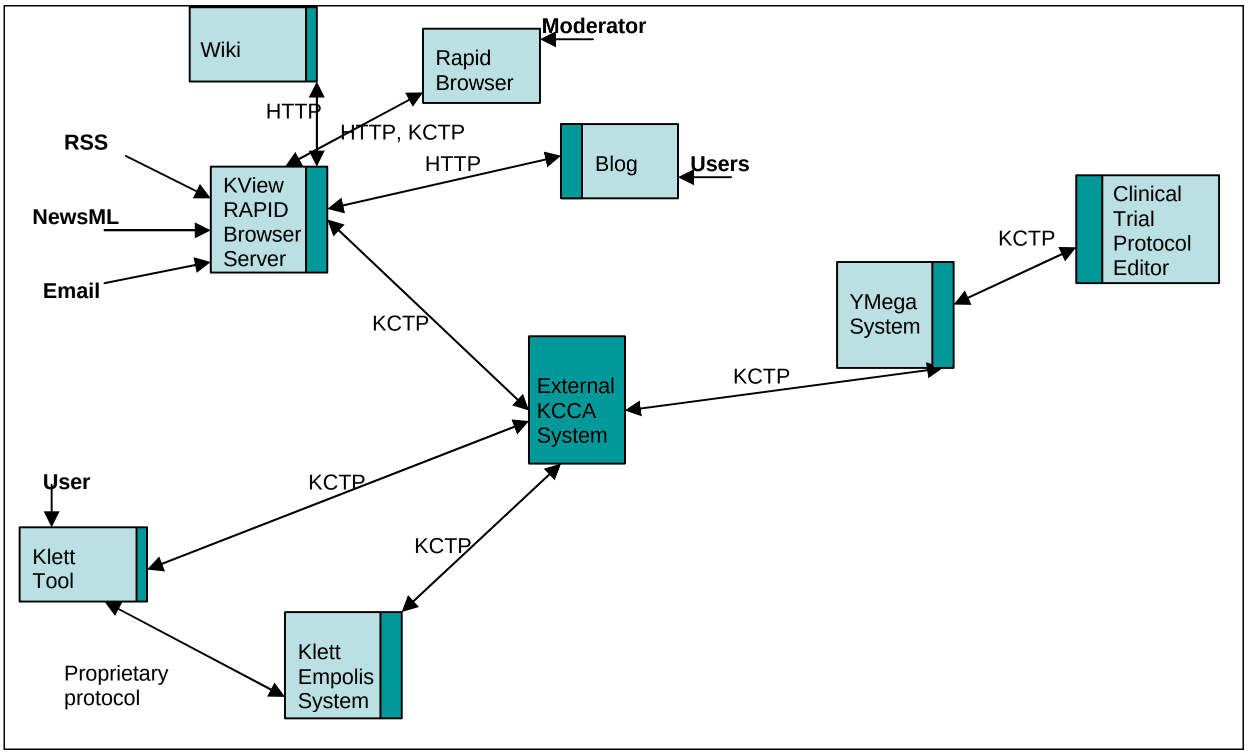


Figure 37: Interaction of the three application domain systems via KCCA/KCTP.

Below we present a cross domain use case for accessing content from multiple repositories via the KCCA Middleware Platform.

The Klett Application involves accessing educational content locally as well as from multiple external repositories. It needs to access educational content (digital items, metadata) from external sources such as Digital Library Deutsches Museum. By using the KCCA middleware the Klett Design Application, will be able to also access the latest news from the KViews Content Repository.

The below diagram explains the system interaction of the Klett KCCA Server with the KViews KCCA Server via using another Public KCCA Server.

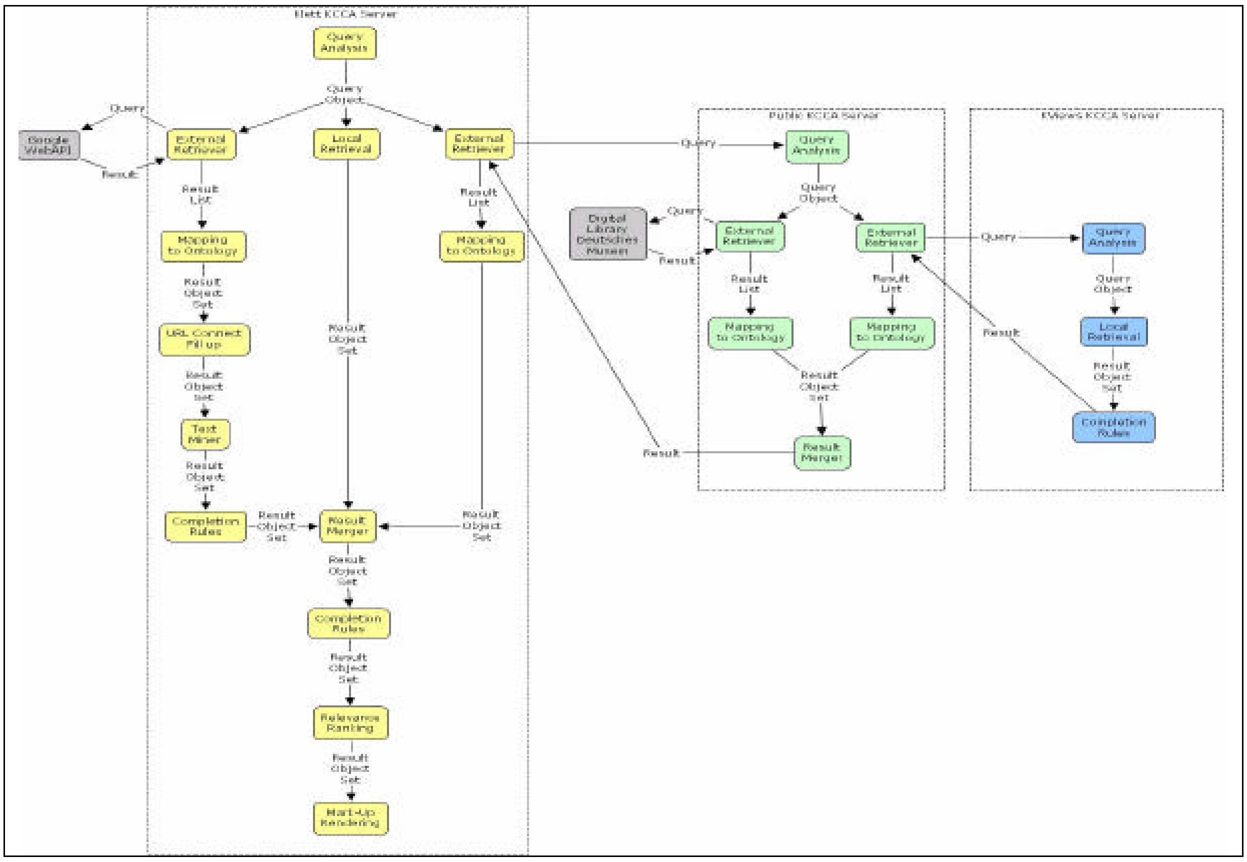


Figure 38: Klett Application accessing content from external middleware systems.

The "External Retrieval" process in a KCCA middleware is responsible for accessing content from external resources. The Klett KCCA Server makes a query to the public KCCA Middleware. The public KCCA Server holds connection to multiple content repositories such as the Deustches Museum and the KViews Middleware System. It queries the appropriate repositories, gets and combines the result set according to a pre-defined ontology and then returns back the answer to the Klett KCCA Server. The Klett Application incorporates these result sets into its local result sets, does a relevance ranking and displays the results back to the user.

10 Conclusion

This document provides a state of the art report on knowledge and content standards, semantic middleware systems and related technologies. It describes an initial architecture consisting of KCCA (Knowledge Content Carrier Architecture) and KCTP (Knowledge Content Transfer Protocol) for building semantic middleware systems. It also defines Knowledge Content Objects (KCO) that carry semantic description about multimedia content and will enable better sharing of knowledge amongst systems. The document also provides an overview of the three use cases (Senior Executives, Education Domain and Clinical Trials) with respect to the three application domains of the project.

Based on the document, a prototype of the KCCA and three application use cases will be developed. Existing systems within the application domain, which are not METOKIS compatible will be METOKIS enabled by using suitable adaptors.

11 References

[ABC01]	Lagoze, C. and J. Hunter (2001) "The ABC Ontology and Model". Journal of Digital Information, 2 (2) http://metadata.net/harmony/JODI_Final.pdf
[ABC02]	M.Doerr, J.Hunter, C. Lagoze "Towards a Core Ontology for Information Integration", submitted to Journal of Digital Information, October 2002 http://jodi.ecs.soton.ac.uk/Articles/v04/i01/Doerr/doerr-final.pdf
[ABC03]	Hunter, J. (2003) "Enhancing the Semantic Interoperability of Multimedia through a Core Ontology". IEEE Transactions on Circuits and Systems for Video Technology, January http://archive.dstc.edu.au/RDU/staff/jane-hunter/events/final_paper.pdf
[Abiteboul95]	Abiteboul, Hull, Vianu, 1995, Foundations of Databases, Addison Wesley.
[Beckett02]	Beckett, 2002, Connecting XML, RDF and Web Technologies for Representing Knowledge on the Semantic Web
[Biezunski01]	Biezunski M, Newcomb S R, 2001, Enhancing the Web for Knowledge Management, http://www.idealliance.org/papers/xml2001/papers/html/04-04-03.html
[BPEL01]	Business Process Execution Language for Web Services version 1.1 http://www-106.ibm.com/developerworks/library/ws-bpel/
[CBD]	Concise Bounded Description http://swdev.nokia.com/uriqa/CBD.html
[CDISC]	Standard Protocol Elements for Regulated Clinical Trials: Comprehensive Element List http://www.cdisc.org/standards/ProtocolElements_FullList1-0-0.pdf
[CDISC01]	Glossary Clinical Trials Terminology www.cdisc.org/glossary/CDISCGlossaryTerminologyV2.pdf
[CDISC02]	Glossary Acronyms, Abbreviations, and Initials http://www.cdisc.org/standards/ProtocolElements_FullList1-0-0.pdf
[CIDOC02]	Doerr, M. (2002) "The CIDOC CRM - an Ontological Approach to Semantic Interoperability of Metadata". *AI Magazine* - Special Issue on Ontologies, March
[COCOON]	Cocoon Web Development Framework http://cocoon.apache.org/
[Crubezy02]	Crubezy, M., Motta, E., Lu, W. and Musen, M. (2002) Configuring Online Problem-Solving Resources with the Internet Reasoning Service. IEEE Intelligent Systems 2002.
[DAMLS01]	An Experience Report on using DAML-S http://www.iids.org/publications/essw03.pdf
[DAML+OIL]	Ian Horrocks. DAML+OIL: A Description Logic for the Semantic Web. In IEEE Data Engineering Bulletin Vol. 25 No 1 Pages 4-9, 2002.
[Dawes04]	URI bloat in queries to smushed data http://www.phildawes.net/blog/2004/09/11/uri-bloat-in-queries-to-smushed-data/
[DC01]	Dublin Core Metadata Initiative http://dublincore.org
[Decker01]	Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al. (eds.): Semantic Issues in Multimedia Systems. Proceedings of DS-8. Kluwer Academic Publisher, Boston, 1999, 351-369

[DIP04]	Data, Information and Process Integration with Semantic Web Services http://dip.semanticweb.org
[DMAG]	DMAG (Distributed Multimedia Applications Group). http://dmag.upf.es
[DOLCE01]	Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, Luc Schneider Sweetening ontologies with DOLCE, In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 166 - 181, 2002. http://www.isib.cnr.it/infor/ontology/Papers/DOLCEEKAW.pdf
[EML1]	Adrian Rawlings, Peter van Rosmalen, Rob Koper (OUNL), Miguel Rodríguez-Artacho (UNED), Paul Lefrere (UKOU) Survey of Educational Modeling Languages (EMLs) Version 1 September 19st 2002 http://sensei.lsi.uned.es/palo/eml-version1.pdf
[EMMO02]	Sunil Goyal, Wernher Behrendt and Siegfried Reich: EMMOs --- Enhanced Multimedia Meta Objects for Re-purposing of Knowledge Assets in MIS '02 - Meta Informatics Symposium 2002, Esbjerg, August 2002.
[EMMO03]	Schellner Karin, Westermann Gerd Utz, Zillner Sonja and Klas Wolfgang CULTOS: Towards a World-wide Digital Collection of Exchangeable Units of Multimedia Content for Intertextual Studies. In Conference of Distributed Multimedia Systems (DMS2003), 2003, Miami Florida.
[Fensel99]	D. Fensel, V. R. Benjamins, E. Motta, and B. Wielinga. UPML: A framework for knowledge system reuse. In Proceedings of the 16th International Joint Conference on AI (IJCAI-99), page to appear, Sweden, 1999.
[Garshol02]	Garshol L M, 2002, What are Topic Maps, http://www.xml.com/pub/a/2002/09/11/topicmaps.html
[Garshol021]	Garshol L M, 2002, Topic Maps in Content Management, http://www.ontopia.net/topicmaps/materials/itms.html
[ICE01]	The Information & Content Exchange (ICE) Protocol Version 1.1 http://www.icestandard.org/servlet/RetrievePage?site=ice&page=current_specs
[ICH01]	ICH Efficacy Guidelines http://www.mcclurenet.com/ICHefficacy.html
[ISO99]	ISO/IEC 13250:2000 Topic Maps: Information Technology -- Document Description and Markup Languages, December 1999.
[ISO01]	ISO/JTC1/SC 32/WG2 N 000. Conceptual Graphs http://users.bestweb.net/~sowa/cg/cgstand.htm, April 2001
[IMPRIMAT]	The IMPRIMATUR Business Model, Version 2.1. In: IMPRIMATUR Project (Esprit 20676) http://www.imprimatur.net/IMP_FTP/BMv2.pdf
[Indecs00]	<Indecs>, 2000, Putting Metadata to Rights, Summary Final Report of the <Indecs> Project, http://www.indecs.org/pdf/SummaryReport.pdf
[Indecs02]	<Indecs>, 2002, Rights Data Dictionary, rdd White Paper of the <Indecs> Consortium, http://www.doi.org/topics/indecs-rdd-white-paper-may02.pdf
[Inkass01]	Andreas Abecker, Dimitris Apostolou, Jasmin Franz, Wolfgang Maass, Gregory Mentzas, Christian Reuschling, Sylvio Tabor - Towards an Information Ontology for Knowledge Asset Trading. http://imu.iccs.ntua.gr/Papers/C59-ICE_2003-INKASS_Final.pdf
[IPRONTO]	Jaime Delgado, Isabel Gallego, Silvia Llorente, Roberto García - IPROnto: an Ontology for Digital Rights Management http://hayek.upf.es/dmag/papers/jdigslrgjurix2003.pdf

[Jin01] Yuhui Jin, Stefan Decker, and Gio Wiederhold. OntoWebber: Model-Driven Ontology-Based Web Site Management. In Semantic Web Working Symposium (SWWS), Stanford University, California, USA, July 30 -- August 1, 2001. http://www-db.stanford.edu/pub/gio/2001/Ontowebber01.pdf

[LOM01] Draft Standard for Learning Object Metadata http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

[LOM02] Learning Technology Standards Committee: http://ltsc.ieee.org/index.html

[FACT98] I. Horrocks. The FaCT System. In Automated Reasoning with Analytic Tableaux and Related Methods: International Conference (Tableaux'98), volume 1397 of LNCS, pages 307--312. Springer-Verlag, May 1998.

[Franz02] Franz et al, 2002, Information Ontology - A Semantic Perspective on Knowledge Trading - Applying Case-Based Reasoning.

[Geurts03] Joost Geurts, Stefano Bocconi, Jacco van Ossenbruggen, Lynda Hardman: Towards Ontology-Driven Discourse: From Semantic Graphs to Multimedia Presentations. International Semantic Web Conference 2003: 597-612

[GINF99] S. Melnik: Generic Interoperability Framework, Working Paper, 1999 http://www-diglib.stanford.edu/diglib/ginf/

[JBOSS] JBoss - J2EE Application Server - http://www.jboss.org

[JENA] JENA Semantic Web Framework for Java - http://jena.sourceforge.net/

[JETSPEED] Jetspeed Enterprise Information Portal http://portals.apache.org/jetspeed-1/

[JMS] Java Message Service (JMS) http://java.sun.com/products/jms/

[JOSEKI] RDF Web API – JOSEKI http://www.joseki.org/

[J2EE] Java 2 Enterprise Edition J2EE - http://www.java.sun.com/j2ee/

[HERA01] Geert-Jan Houben, Peter Barna, Flavius Frasincar, Richard Vdovjak: Hera: Development of Semantic Web Information Systems. ICWE 2003: 529-538 http://wwwis.win.tue.nl/~hera/papers/ICWE2003/icwe.pdf

[HL7RIM] HL7 Reference Information Model http://www.hl7.org/library/data-model/RIM/modelpage_mem.htm

[KAON03] Raphael Volz, Daniel Oberle, Steffen Staab, Boris Motik KAON SERVER - A Semantic Web Management System In Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003. ACM, 2003.

[KAON04] Daniel Oberle, Andreas Eberhart, Steffen Staab, Raphael Volz Developing and Managing Software Components in an ontology-based Application Server In Middleware 2004, ACM/IFIP/USENIX 5th International Middleware Conference, Toronto, Ontario, Canada. Springer, 2004

[KIF] KIF Manual Version 3.0 http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps

[KOWARI] Kowari Metastore http://www.kowari.org/

[Maedche02] A. Maedche, B. Motik, N. Silva, R. Volz MAFRA - An Ontology Mapping Framework in the Semantic Web, In Proceedings of the ECAI Workshop on Knowledge Transformation, Lyon, France, 2002 2002/07/01.

[Mass01] Mass, 2002, INKASS Review Meeting, 3/10/2002, Brussels, (ppt slides)

[Maass02] Maass, 2002, Generic Structure of Information Objects, INKASS internal paper.

[MIKSI04] A. Wahler, B. Schreder, A. Balaban, J.M. Gomez, and K. Niederacher: MIKSI - A Semantic and Service Oriented Integration Platform. ESWS 2004:459-472

[Moira03] Moira C. Norrie, Beat Signer, 2003, Issues of Information Semantics and Granularity in Cross-Media Publishing. CAiSE 2003: 421-436

[Motta03] Motta, E., Domingue, J., Cabral, L. and Gaspari, M. (2003) IRS-II: A Framework and Infrastructure for Semantic Web Services. 2nd International Semantic Web Conference (ISWC2003) 20-23 October 2003, Sundial Resort, Sanibel Island, Florida, USA.

[MPEG-7] José María Martínez Sanchez, Rob Koenen, Fernando Pereira: MPEG-7: The Generic Multimedia Content Description Standard, Part 1. IEEE MultiMedia 9(2): 78-87 (2002)

[MPEG-21] Jan Bormans, Keith Hill MPEG-21 Overview v.5 http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm

[Newcomb02] Newcomb S R, 2002, The Economics of the Topic Maps Reference Model www.idealliance.org/papers/xml02/ dx_xml02/papers/05-03-05/05-03-05.pdf

[NEWSML] NewsML Version 1.2 Functional Specification, 10 Oct. 2003 http://www.newsml.org/IPTC/NewsML/1.2/specification/NewsML_1.2-spec-functionalspec_7.html

[NITF] News Industry Text Format, NITF Tutorial http://www.nitf.org/tutorial.php

[Ossen02] Jacco van Ossenbruggen, Lynda Hardman: Smart Style on the Semantic Web. Semantic Web Workshop 2002

[Ossen03] Jacco van Ossenbruggen, Lynda Hardman, Joost Geurts, Lloyd Rutledge: Towards a multimedia formatting vocabulary. WWW 2003: 384-393

[OWL] OWL Web Ontology Language 1.0 Reference*.* Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. W3C Working Draft 12 November 2002.

[OWLQL01] Richard Fikes, Pat Hayes, Ian Horrocks. OWL-QL: A Language for Deductive Query Answering on the Semantic Web. KSL Technical Report 03-14 http://classweb.gmu.edu/kersch/infs770/Topics/SemanticWeb/KSL-03-14(OWL-QL).pdf

[OWLS01] OWL-S Semantic Markup For Web Services http://www.daml.org/services/owl-s/1.0/owl-s.html

[OWLS02] A Broker for OWL-S Web Services http://www-2.cs.cmu.edu/~softagents/papers/aaai_ss04_broker.pdf

[PALO1] Miguel Rodríguez-Artacho PALO Language Overview Draft version Technical Report http://sensei.lsi.uned.es/palo/PALO-TR.pdf

[PALO2] PALO Language Homepage http://sensei.lsi.uned.es/palo/

[PIAZZA01] Piazza: Mediation and Integration Infrastructure for Semantic Web Data http://www.cis.upenn.edu/~zives/research/piazza-jws.pdf

[PIAZZA02] **Piazza: Data Management Infrastructure for Semantic Web Applications** http://www.cis.upenn.edu/~zives/research/piazza-www03.pdf

[Porter80] Porter, M.F., 1980, An algorithm for suffix stripping, Program, **14**(3):130-137 http://www.tartarus.org/~martin/PorterStemmer/index.html

[PRISM] PRISM: Publishing Requirements for Industry Standard Metadata Specification Version 1.2(h), September 23, 2003. http://www.prismstandard.org/PAM_1.0/PRISM_1.2h.pdf

[PSL01] The Process Specification Language (PSL) Overview and Version 1.0 Specification http://www.mel.nist.gov/psl/pubs/PSL1.0/paper.doc

[QEL] Mikael Nilsson, Wolf Siberski RDF Query Exchange Language (QEL) - concepts, semantics and RDF syntax -Working Draft. http://edutella.jxta.org/spec/qel.html

[RDFQ] RDFQ – RDF Query http://sw.nokia.com/rdfq/RDFQ.html

[RDF99] O. Lassila and R.R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium (W3C), http://www.w3.org/TR/1999/REC-rdf-syntax-19990222, February 1999.

[RDF02] D. Brickley and R.V. Guha. Resource Description Framework (RDF) Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, World Wide Web Consortium (W3C), http://www.w3c.org/TR/2002/WD-rdfschema-20020430, April 2002.

[RSS01] RSS 1.0 Specification http://purl.org/rss/1.0/spec

[RSS02] RSS 2.0 Specification http://blogs.law.harvard.edu/tech/rss

[RuleML] Rule Markup Language Initiative http://www.ruleml.org

[Rust00] Godfrey Rust and Mark Bide. 2000. The <in*d*ecs> Metadata Framework: Principles, Model and Data Dictionary. http://www.indecs.org/pdf/framework.pdf

[RVL03] Aimilia Magkanaraki, Val Tannen, Vassilis Christophides and Dimitris Plexousakis: Viewing the Semantic Web Through RVL Lenses, In Proceedings of the Second International Semantic Web Conference 2003 (ISWC 2003).

[SCAM01] SCAM Framework - http://scam.sourceforge.net/

[Schlosser02] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. In International Workshop on Agents and Peer-to-Peer Computing, Bologna, Italy, July 2002.

[SCORM04] Sharable Content Object Reference Model (SCORM) 2004 http://www.adlnet.org/index.cfm?fuseaction=DownFile&libid=648&bc=false

[SEAL01] Staab, Studer, Sure, Volz SEAL: a SEmantic portAL with content management. (2002)

[SEAL02] Alexander Maedche, Steffen Staab, Nenad Stojanovic, Rudi Studer and York Sure: Semantic Portal - The Seal Approach http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/2001_mit-book.pdf

[SERQL] SeRQL Sesame RDF Query Language http://www.openrdf.org/doc/users-1.1/ch05.html

[SERVLET] Java Servlet technology http://java.sun.com/products/servlet/

[SHAME] SHAME - http://kmr.nada.kth.se/shame/

[Smeul02] Arnold Smeulders, Lynda Hardman and Guus Schreiber. Integrated Access to Cultural Heritage E-Documents. Proceedings 4th Intl. Workshop on Multimedia Information Retrieval, Juan-les-Pins, 2002

[SMIL] Synchronized Multimedia Integration Language (SMIL 2.0), www.w3.org/TR/smil20/

[SMOM] Han Li and Guofei Jiang. Semantic Message Oriented Middleware for Publish/Subscribe Networks. http://www.ists.dartmouth.edu/ISTS/library/infrastructure-security/smo0404.pdf

[STRUTS] Struts Web Application Framework - http://struts.apache.org/

[SUMO01] Ian Niles, Adam Pease, Towards a Standard Upper Ontology http://home.earthlink.net/~adampease/professional/FOIS.pdf

[SWIM01] The ICS-FORTH SWIM: A Powerful Semantic Web Integration Middleware http://www.cs.uic.edu/~ifc/SWDB/papers/Christophides_etal.pdf

[SWRL04] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. By Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. W3C Member Submission. 21-May-2004.

[Tanenbaum02] Tanenbaum A, van Steen M, 2002, Distributed Systems - Principles and Paradigms,

[TAPESTRY] Tapestry Java Framework for Web Applications http://jakarta.apache.org/tapestry/

[TAP] TAP:Towards a Web of Data http://tap.stanford.edu/j1.html

[TAP01] TAP Project http://tap.stanford.edu

[TM00] ISO/IEC 13250:2000 Topic Maps: Information Technology – Document Description and Markup Languages, December 1999.

[TM01] Living with topic maps and RDF http://www.ontopia.net/topicmaps/materials/tmrdf.html

[TRIX01] Carroll, Jeremy J.; Stickler, Patrick: RDF Triples in XML, HPL-2003-268 20040211 http://www.hpl.hp.com/techreports/2003/HPL-2003-268.pdf

[URIQA] URIQA – URI Query Agent Model http://sw.nokia.com/uriqa/URIQA.html

[Vatant03] Vatant B, 2003, OWL and Topic Map Pudding, www.mondeca.com/owl/owltm.htm

[Villard01] Villard, Rosin, Layaida, 2001, An XML based multimedia document processing model for content adaptation

[WebDAV99] Jim Whitehead, Yaron Y. Goland: WebDAV : WebDAV: A network protocol for remote collaborative authoring on the Web. Submitted to European Computer Supported Cooperative Work (ECSCW'99).

[WebDAV01] WebDAV http://www.webdav.org

[WERKL96] Herzog M., Petta P., 1996, Knowledge Representation in WERKL, an Architecture for Intelligent Multimedia Information Systems, Austrian Research Institute for Artificial Intelligence, Vienna, TR-96-20, 1996.

[Wiele1] Jan Wielemaker, Guus Schreiber, Bob J. Wielinga: Prolog-Based Infrastructure for RDF: Scalability and Performance. International Semantic Web Conference 2003: 644-658

[Wiele03] Supporting Semantic Image Annotation and Search, Jan Wielemaker, August Th. Schreiber and Bob J. Wielinga in: S. Handschuh and S. Staab (eds), Annotation for the Semantic Web, Volume 96 Frontiers in Artificial Intelligence and Applications, 2003, 240 pp., hardcover

[WIPO] World Intellectual Property Organization. http://www.wipo.org

[WordNet01] WordNet: An Electronic Lexical Database, MIT Press, May 1998 ISBN 0-262-06197-X

[WordNet02] WordNet http://www.cogsci.princeton.edu/~wn/index.shtml

[WSDL01] Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/wsdl

[WSMO01] Web Service Modeling Ontology http://www.wsmo.org

[WSML] Web Service Modeling Language http://www.wsmo.org/wsml/index.html

[WSMX] Web Service Execution Environment http://www.wsmo.org/wsmx/

[W3CACL] W3C ACL System http://www.w3.org/2001/04/20-ACLs.html

[XML02] Extensible Markup Language (XML) (2002) W3C Architecture Domain http://www.w3.org/XML/

[XMP01] Adobe Extensible Metadata Platform (XMP) Specification, January 2004 http://partners.adobe.com/asn/tech/xmp/pdf/xmpspecification.pdf

[XRML01] XrML 2.0 Technical Overview, Version 1.0 March 8, 2002 http://www.xrml.org/reference/XrMLTechnicalOverviewV1.pdf

12 GLOSSARY

The glossary is by no means exhaustive, but focuses on those terms that we feel are particularly important for understanding this document and the rationale behind the METOKIS System. The glossary will be extended and maintained as a separate work item within the project.

TERM	EXPLANATION
METOKIS System	A set of http enabled Internet nodes consisting of METOKIS compliant servers and clients
KCCA	Knowledge Content Carrier Architecture - the software component structure needed to build a METOKIS compliant node
KCTP	Knowledge Content Transfer Protocol - the communication protocol needed for two or more KCCA nodes to exchange information
KCO	Knowledge Content Object - a defined data structure that holds semantic descriptions and references to actual multimedia resources, as well as meta data. KCOs are the main objects that are transmitted between KCCA nodes in a METOKIS system, using the KCT Protocol.
RDF	Resource Description Framework - A description language with an XML encoding that allows developers to specify where machine-interpretable information can be found on the Web. RDF is expressed in triples corresponding to the linguistic notion of subject - predicate - object.
Web Services	Web Services are Internet-level extensions of the ideas of remote procedure calls (RPC) or Object Request Brokers (ORB) in distributed object oriented systems. Web Services are seen as a key component of future global distributed systems as they should enable the semi-automatic chaining of business processes through high-level descriptions of services offered and through machine-enforced service level contracts. Several competing research prototypes and architectures exist at the time of writing (September 2004).
Interoperation	One of the key motivations for METOKIS is to develop an infrastructure that is well suited to interoperation between heterogeneous systems. The traditional approach has been one of translation between different systems that "spoke" different languages. The approach in METOKIS is to - one the one hand - support translation mechanisms where they are needed and - on the other hand - define the objects of interest (knowledge content objects) sufficiently well so that heterogeneous systems achieve homogeneous results by manipulating KCOs in their specific ways, but still producing valid KCOs as output. The idea is similar to relational algebra (the result of relational operators are always relational tables), but the objective is much more ambitious because we attempt interoperation at a conceptually higher level (e.g. merging contracts or propositional content or interactive multimedia presentations). METOKIS will only make a start, the aim is seamlessly interoperable content.
Metadata (in KCOs)	Any knowledge about data is usually referred to as meta data. Two conflicting interpretations arise from this: one school says "there is no meta data, because everything is *knowledge*". Recognising the pope on a video is knowledge and knowing that the frame rate of that video is 18 frames/second is also knowledge. The other school says "everything that is not rendered media content is *meta data*". We have decided to make a definitional distinction between meta data and knowledge: meta data is information that is ontologically more related to the medium than to what the medium portrays. knowledge is information that can be gleaned from consuming the

	medium. We therefore place the frame rate in the meta data container and the information about the pixel cloud that represents the pope, in the knowledge container of the KCO.
Ontologies	It is useful to distinguish up to four levels of ontologies, before embarking on discussion what is right or wrong with a specific ontology. The most generic ontologies are foundational ontologies which can be very abstract, but give us a formalism to e.g. distinguish between things that change over time and things that keep their properties over time. At the second level come sector-ontologies which cover the knowledge pertaining to a fairly large field, e.g. an ontology of geographical terms and relations, or an ontology of multimedia resources (e.g. the MPEG family of standards). At the third level come community ontologies that describe the knowledge (within one or more sectors) at a more specific level. An example would be a particular firm's knowledge about the legal requirements of doing transport of goods between the EU and Russia. At the fourth level come ontologies that include instantiations of the concepts defined at the higher levels. This would include knowledge about specific persons who need to sign certain documents in order for a specific transport job to be successfully accomplished between Russia and Poland.
Context Profile (KCCA term)	The Context Profile is a simple (often 1-to-1) mapping of the database schema terms into an equivalent RDF schema for the purpose of wrapping an external data source for METOKIS
View Profile (KCCA term)	The View Profile is a view that may be defined even across several context profiles and its purpose is to align the external data with the schema / ontology used by a METOKIS application
Context (KCCA term)	A set of instances (data) that conforms to a Context Profile
View (KCCA term)	A set of instances (data) that conforms to a View Profile
KCCA Request Broker	A light-weight software bus into which KCCA applications can be plugged
KCCA Repository	A (virtual) database which references the context and view profiles of the actual data sources participating in the METOKIS system
KCCA Middleware Components	Authentication, Workflow Engine, Session Management, Inference Engine, Domain Specific Rule Repository, System Registry
KCTP-KCCA Profile	declares to the external world which protocol bindings and data encodings apply to this system
KCTP-RDF Data Profile	declares to the external world the query mechanisms and operations that this system supports
KCTP-Service Profile	declares to the external world the services (system and domain specific application services) that this system supports
KCTP-KCO Profile	declares to the external world the KCO data structure and the operators and services (on the KCO) that this system supports
KCTP-Session Profile	declares to the external world (client and server) the schema by which session information is organised, kept and exchanged