

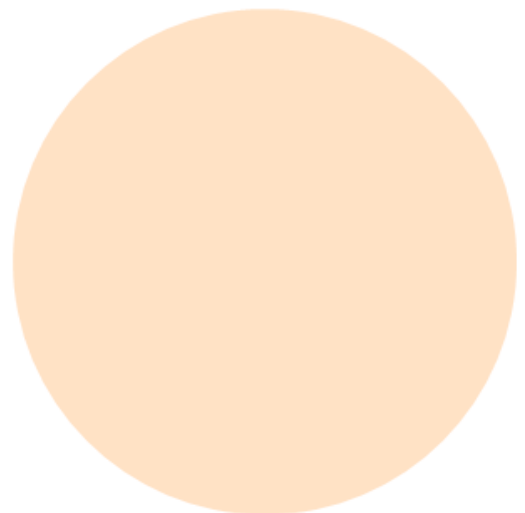


Suitability of Metadata Workbench for data modeling in Finnish social services

Test report

28.4.2011

Miika Alonen
Konstantin Hyppönen



Version	Date	Sections	Description of changes	Authors
0.1	10.4.2011		First draft	MA
0.2	29.4.2011		Text editing	KH
0.3	2.5.2011		More text editing	MA, KH
0.4	27.5.2011		Finnish abstract	MA
1.0	7.6.2011		Approved final version	

NATIONAL PROJECT FOR IT IN SOCIAL SERVICES
MINISTRY OF SOCIAL AFFAIRS AND HEALTH

Association of Finnish Local and Regional Authorities
National Institute for Health and Welfare
The East Finland Social and Welfare Centre Of Expertise

Content

1	Background	4
2	MDW Architecture.....	6
2.1	SPIN and SPARQLMotion.....	6
3	MDW Evaluation	8
3.1.1	Core Component Technical Specification.....	8
3.2	Metadata model evaluation	10
3.3	Functional evaluation.....	12
3.3.1	Core Components	13
3.3.2	Business Components.....	15
3.3.3	Business Documents	16
3.3.4	Analysis	17
3.3.5	Schema generation.....	18
4	Summary	19
4.1.1	Missing functionality.....	20
	Annex 1	21

Tiivistelmä

Metadata Workbench (MDW) on metatietojen muodostamiseen ja hallintaan tarkoitettu työkalu. MDW on kehitetty yhteistyössä Hollannin oikeusministeriön ja TopQuadrant-yhtiön kanssa. TopQuadrant-yhtiö kehittää ja markkinoi semanttisia teknologioita, kuten Topbraid Composer ja Topbraid Live. MDW perustuu semanttisiin teknologioihin ja käyttää ontologioita UN/CEFACT CCTS mukaiseen asiakirjamallinnukseen.

MDW-järjestelmää tutkittaessa alustavina vaatimuksina järjestelmälle oli: tietokomponenttien ja asiakirjojen versiointi, kollaboratiivinen ympäristö sekä muutosten- ja versionhallinta.

MDW on teknisesti hyvin kompleksinen ohjelmisto, jonka toiminta perustuu ontologoiden ilmaisuvoimaan ja päättelyyn. MDW:n käyttöliittymä on käyttäjäystävällinen, ja sisältää vain käyttäjälle tarpeellisia toiminnallisuuksia. Järjestelmä mahdollistaa skeemojen automaattisen generoinnin sisällön asiantuntijoiden muodostamista asiakirjarakenteista, sekä muodostettujen tietokomponenttien ja asiakirjojen analysoinnin.

MDW:n käyttämä CCTS-tietomalli on mutkikas, koska se pyrkii mallintamaan koko CCTS menetelmän. Tietomallia on kuitenkin mahdollista yksinkertaistaa ja kehittää eteenpäin sosiaalihuollon tarpeisiin paremmin sopivaksi. Järjestelmä sisältää kaiken tarvittavan toiminnallisuuden tietokomponenttien, ja asiakirjojen kollaboratiiviseen kehittämiseen. Kehitettävää on kuitenkin erityisesti versio- ja muutostenhallinnan osa-alueella.

Järjestelmän kehittäminen edellyttää semanttisten teknologioiden, kuten Topbraid ympäristön osaamista. MDW-käyttöliittymä ja CCTS-tietomalli perustuu avoimeen lähdekoodiin, ja mahdollistaa näin ollen järjestelmän avoimen kehityksen. MDW-järjestelmän kehittäminen edellyttää kuitenkin Topbraid Composer kehittäjälisenssiä. Lisäksi järjestelmän käyttö kollaboratiivisesti edellyttää toistaiseksi TopBraid Live -ohjelmistoa jonka kustannukset vaihtelee käyttötarpeen ja sopimuksen mukaan.

1 Background

Metadata workbench (MDW) is a tool for developing and maintaining metadata. The tool has been developed by the Dutch Ministry of Security and Justice (MoJ) with tools and support provided by TopQuadrant.

MDW is based on semantic web technologies and uses ontologies for the construction and management of business document structures. MDW uses OWL¹ for the representation of core components, business information entities and business documents modeled according to the UN/CEFACT CCTS² specification. OWL is a rich knowledge representation language which can be used for the representation of any knowledge, whereas CCTS is strict way of constructing business documents with re-usable building blocks.

This report is based on the requirements for harmonizing document structures used in Finnish social services. The requirements were identified in the Finnish National Project in IT for Social Services (Tikesos). Some preliminary requirements for managing information entities and business documents were:

- Versioning of the components and business documents
- Collaborative environment for content specialists
- Change management

¹ OWL 2 Web Ontology Language. Document Overview. W3C Recommendation 27 October 2009. <http://www.w3.org/TR/owl2-overview/>

² United Nations. Centre for Trade Facilitation and Electronic Business. Core Components Technical Specification. Version 3.0. 29 September 2009. <http://www.unece.org/cefact/codesfortrade/CCTS/CCTS-Version3.pdf>

2 MDW Architecture

MDW is built using the TopBraid Suite Architecture which is a semantic web application platform for developing and deploying semantic web services. TopBraid tools provide a large set of semantic modules that are easily combined, assembled and deployed as Web Services.

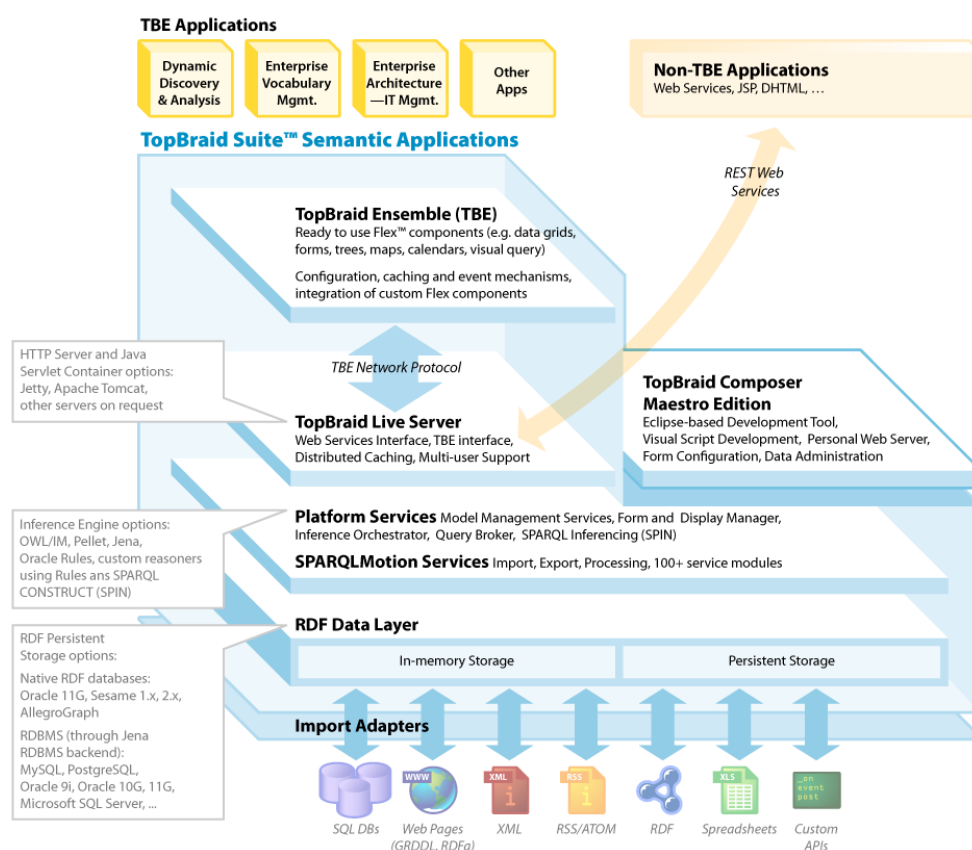


Figure 1: TopBraid Suite Architecture

TopQuadrant provides a flexible infrastructure for developing semantic web applications. The tools enable quick implementation for solutions that integrate data, content, application services and user interactions.

2.1 SPIN and SPARQLMotion

SPARQLMotion³ is a scripting language for semantic web data processing. Scripts are defined in RDF⁴ using ready-made modules that implement a wide range of data processing tasks. MDW uses SPARQLMotion scripts that are executed as REST⁵ Web services.

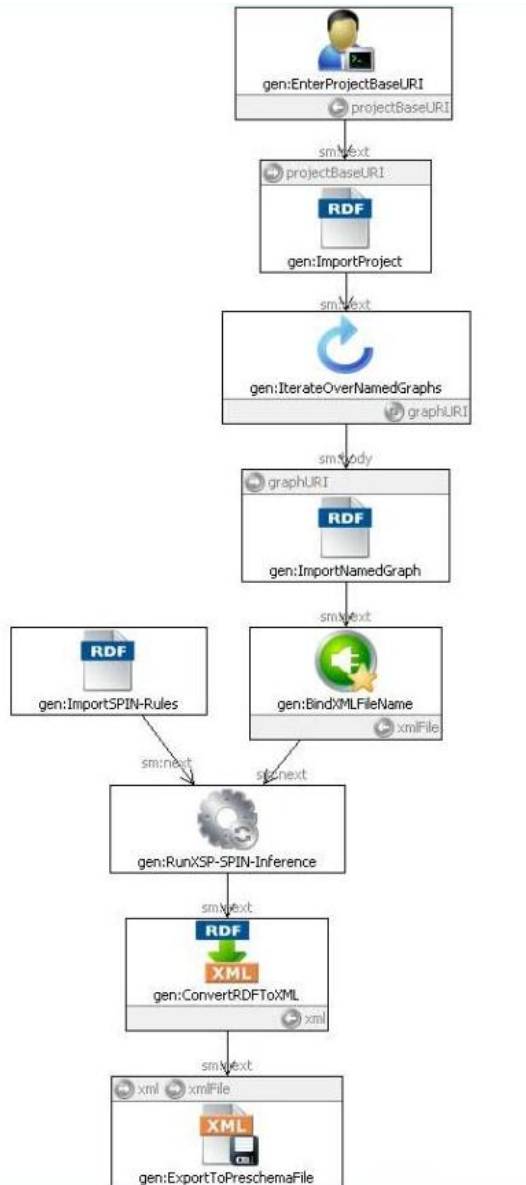


Figure 2: SPARQLMotion script that generates XML Schema Plus from RDF

Transformations from OWL representations of document structures into XML schemas^{6,7} is done with a set of SPARQLMotion scripts. The scripts generate the schema from the ontology (see Figure 2).

³ SPARQLMotion <http://sparqlmotion.org/>

⁴ RDF Primer. W3C Recommendation 10 February 2004 <http://www.w3.org/TR/rdf-primer/>

⁵ Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. Chapter 5. Representational State Transfer (REST). http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

⁶ XML Schema Part 1: Structures Second Edition. W3C Recommendation, 28 October 2004. <http://www.w3.org/TR/xmlschema-1>

⁷ XML Schema Part 2: Datatypes Second Edition. W3C Recommendation, 28 October 2004. <http://www.w3.org/TR/xmlschema-2>

3 MDW Evaluation

The evaluation report is divided into two parts: metadata model evaluation and functional evaluation. The metadata model used in MDW has been developed by MoJ in collaboration with TopQuadrant. Several evaluation approaches were examined, and some of these are also relevant for data modeling in the social services domain. Evaluation approach for the CCTS model was created considering:

- Performance
- Understandability
- Modularity
- Longevity
- Maintainability
- Ease of inferencing
- Ease of schema generation
- Integrateability / Interoperability
- Traceability

7 evaluation criteria were chosen for the evaluation of metadata model designed for CCTS:

1. CCTS Model MUST represent all of the constructs defined in the CCTS specification
2. CCTS Core Components ontology MUST be created from rich ontology
3. CCTS XML Schemas MUST be generated from ontology
4. Separate rich and CCTS ontologies
5. Ability to create instances from the model
6. CCTS Metamodel must be partitionable into separate named graphs
7. Ability to annotate in named graphs different to the home graph

The metadata model chosen for the MDW was the best match for these criteria. Metadata model is based on a 3-level approach. The 3-level model expresses the CCTS metamodel as subclasses of OWL structures. These subclasses are meta-classes for CCTS classes and properties. Meta-level model for CCTS structures defines the CCTS language described in the next chapter. The CCTS component level model contains the actual Core Components and Business Components. The instance level holds the actual business information in the components defined in the CCTS language.

3.1.1 Core Component Technical Specification

CCTS defines a method for designing common semantic building blocks called core components. The components form a language used in information exchange by business parties. A number of different core component types is defined (see Figure 3).

Aggregate Core Component (ACC) is an object class describing a concept with clearly defined semantics. The object class contains a number of properties related to this concept. An example of an ACC is Address.

Basic Core Component (BCC) is a property appearing in a certain object class (ACC). BCCs are based on simple data types (called Core Data Types) such as Text, Number, Amount. For example, the street name in an address can be represented as an BCC.

Associated Core Component (ASCC) is a property appearing in a certain object class (ACC). The property is based on another object class (another ACC). For example, the validity period of an address could be represented as an ASCC, based on the ACC Period.

Aggregate Business Information Entity (ABIE) is an object class which is a qualified version of an ACC. It may have a narrower meaning than the parent ACC. Moreover, any property of the ACC may be qualified in the ABIE. An example of an ABIE is Trade_ Address (an address representation used in trade-related business documents).

Basic Business Information Entity (BBIE) and Associated Business Information Entity (ASBIE) are properties of ABIE, constructed by qualifying corresponding BCC and ASCC properties of the parent ACC.

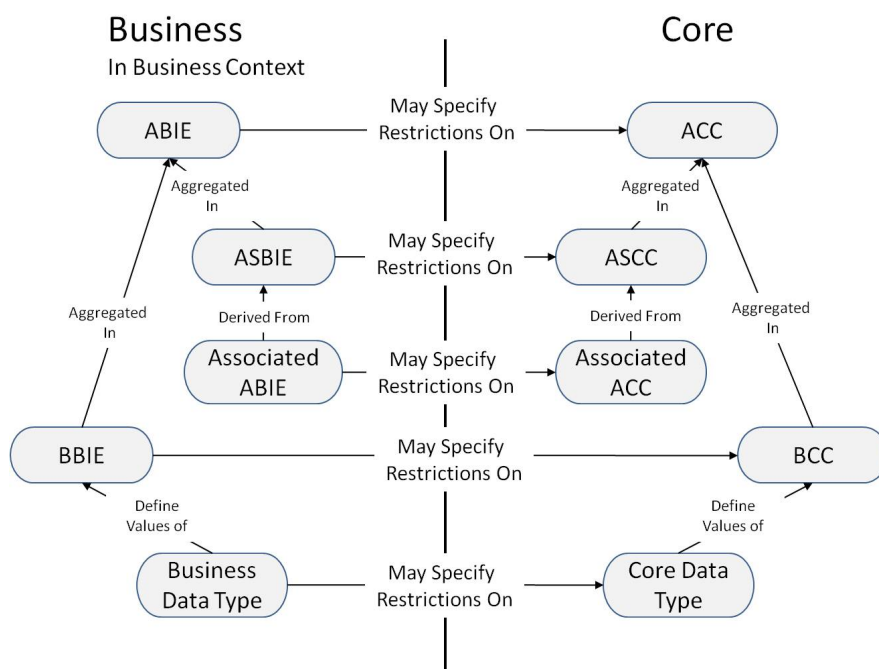


Figure 3: Relationships of different types of core components

3.2 Metadata model evaluation

Metaclasses are specified for every CCTS class type. Every CCTS class in the data model (ontology) is defined as a subclass of the corresponding CCTS meta-class (and not owl:Class). Metaclasses are composed according to the semantics of CCTS. The metaclasses specify which properties can be attached to which CCTS Classes (Figure 4).

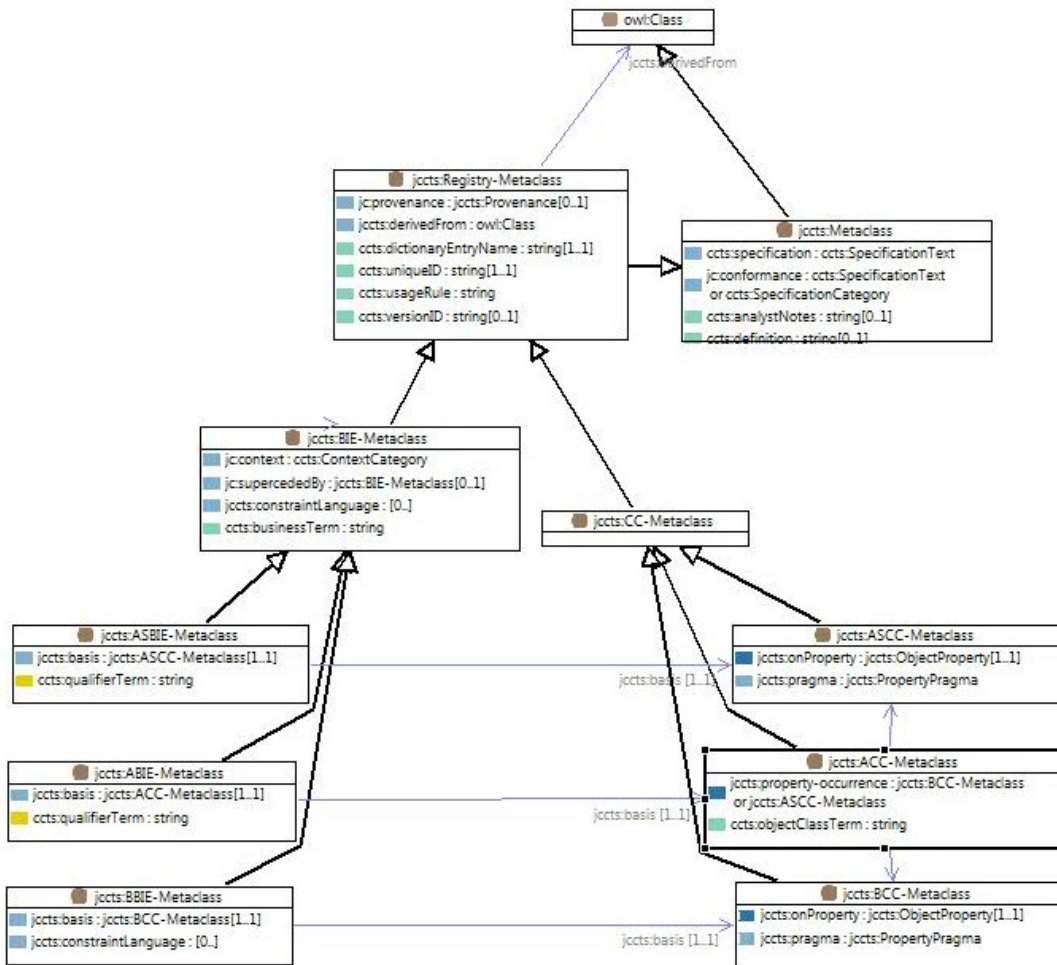


Figure 4: MDW metadata model for core components

jcts:Metaclass is the root for all CCTS Metaclasses defined in the MDW ontology. It contains information that is common for all subclasses. The jcts:Registry-Metaclass is a superclass for all CCTS Metaclasses and defines common and required properties for all CCTS Metaclasses, such as dictionary entry name, unique id, usage rules and version id.

jcts:Metaclass has the following 5 subclasses:

- CC-Metaclass for Core Component Classes
- BIE-Metaclass for Business Information Entities
- DT-Metaclass for Data Types

- BC-Metaclass for Business Circumstances
- BD-Metaclass for Business Document Classes

The `jccts:CC-Metaclass` is an abstract metaclass for BCC, ASCC and ACC. The ACC Metaclass contains the mandatory object class term, i.e. the name of the ACC, and properties (ACC contents) that are references to the child objects contained in the ACC. Both BCC and ASCC are represented as Classes and Object Properties. The `jccts:onProperty` (Figure 5) references the Object Property that is used for describing a restriction (e.g. a cardinality) in the parent ACC.

The `jccts:BIE-Metaclass` is a metaclass for business components. Every subclass of BIE-Metaclass includes a property `jccts:basis` that indicates the core component on which the BIE is based.

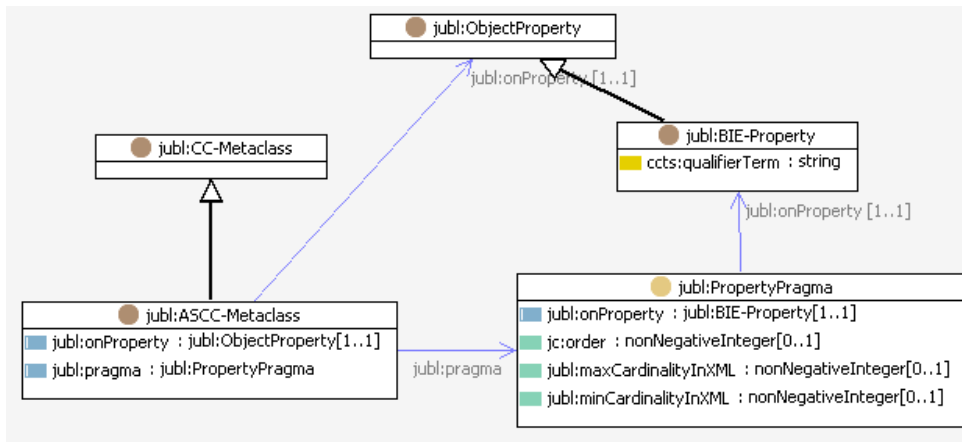


Figure 5: Pragmas for cardinality and order

Every ASCC and BCC has properties associated through `jccts:onProperty` and `jccts:pragma` properties. Property pragmas are associated with the same ASCC and BCC for representing the order of the property occurrences in the ACC Classes. (Figure 5).

DT-Metaclass is a metaclass for Core Data Types. CCTS data types are implemented as object classes and used with object properties.

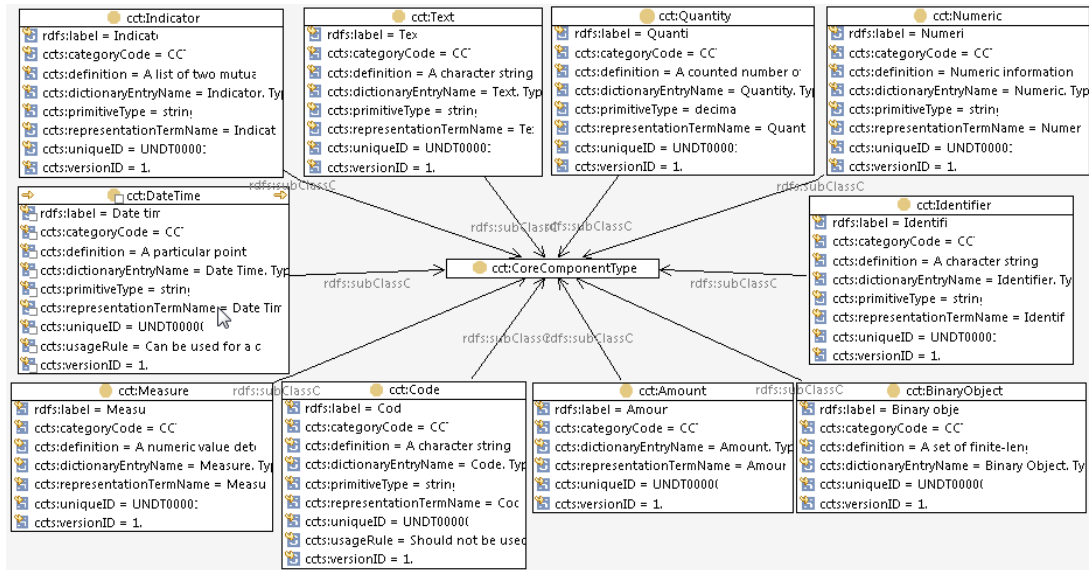


Figure 6: Core Data Types

3.3 Functional evaluation

Functional evaluation lists the functionalities that are implemented in the MDW environment and points out missing and potentially useful features.

Metadata Workbench functions are placed in the user interface on 6 tabs (Figure 7):

- Core Components
- Codelists
- Qualified Datatypes
- Business Information Entities
- Business Documents
- Analysis

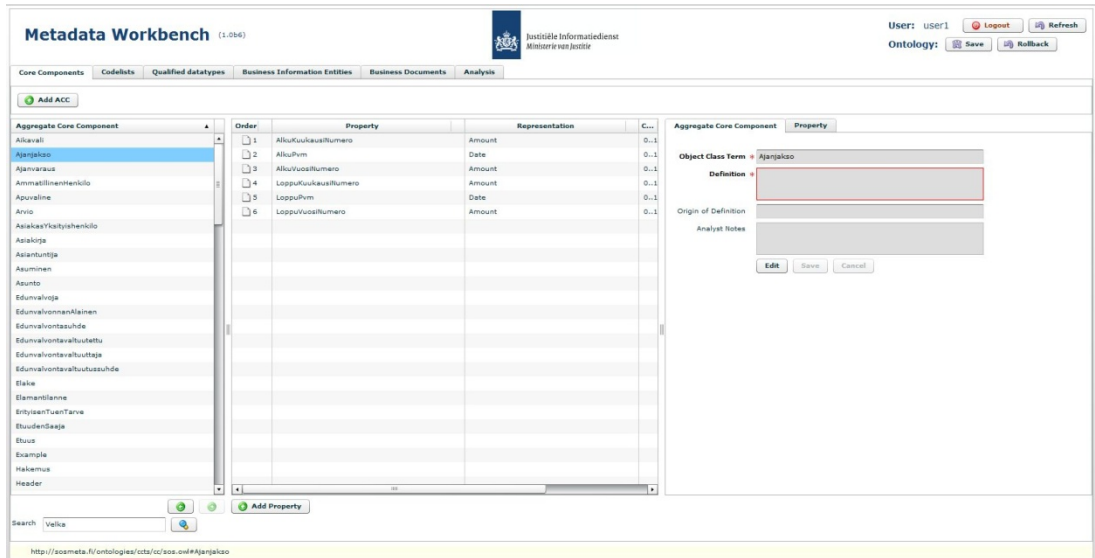


Figure 7: Overview of the MDW interface

Metadata Workbench interface provides functionality to create new Core Components, Business Components and Business Documents.

3.3.1 Core Components

The Core Component tab consists of the following features:

- **Add/Edit Core Component**

Creation of Core Components is requirement for new Business Components and Business Documents as defined in the CCTS Specification (Figure 8).

Figure 8: Add Core Component

- **Add/Edit Core Component Property**

Representation Term or Associated Object Class, Cardinality and Definition are mandatory according to the CCTS Specification. Association properties and basic properties are both expressed using the field named Representation Term (Figure 9).

The screenshot shows a dialog box titled "Add Core Component Property". It has several input fields:

- Property Term**: Example Property
- Representation Term**: Text (with a dropdown arrow)
- Cardinality**: 3 (with up/down arrows), Min: 1 (with up/down arrows), Max: unbounded
- Definition**: Example definition
- Origin of Definition**: (empty text box)
- Analyst Notes**: (empty text box)

 At the bottom, there are three buttons: "Edit", "Save", and "Cancel".

Figure 9: Add Core Component Property

An Aggregate Core Component or an Unqualified Datatype is chosen as a Representation term for the Core Component property (Figure 10).

The screenshot shows the "Add Core Component Property" dialog box with a sub-dialog box titled "Select Representation Term" open. The sub-dialog has two tabs: "Aggregate Core Component" and "Unqualified Datatype". Below the tabs is a list of items:

- Aikavali
- Ajanjakso
- Ajanvaraus
- AmmatillinenHenkilo
- Apuvaline
- Arvio
- AsiakasYksityishenkilo
- Asiakirja
- Asiantuntija
- Asuminen
- Asunto

 To the right of the list is a table with three columns: "Order", "Property", and "Representation". Below the list is a search box and a search icon. At the bottom of the sub-dialog are "Select" and "Cancel" buttons.

Figure 10: Select Representation Term

Editing a Core Component is restricted if there are depending Business Components which use it (Figure 11). The feature is useful as it allows tracing the dependencies between components and prevents the removal of properties which are still used elsewhere.



Figure 11: The editing of a property is restricted if the property is used elsewhere.

3.3.2 Business Components

Business Components are restricted versions of Core Components. MDW provides the following functionality for their management:

- **Add/Edit Business Component based on an ACC**
- **Add/Edit Business Component Properties based on ACC properties**

A Business Component is a restricted (qualified) version of a Core Component. Relevant properties for Business Components are added from the corresponding Core Components. Qualifier terms may be added to specify the semantic meaning of the component (Figure 12).

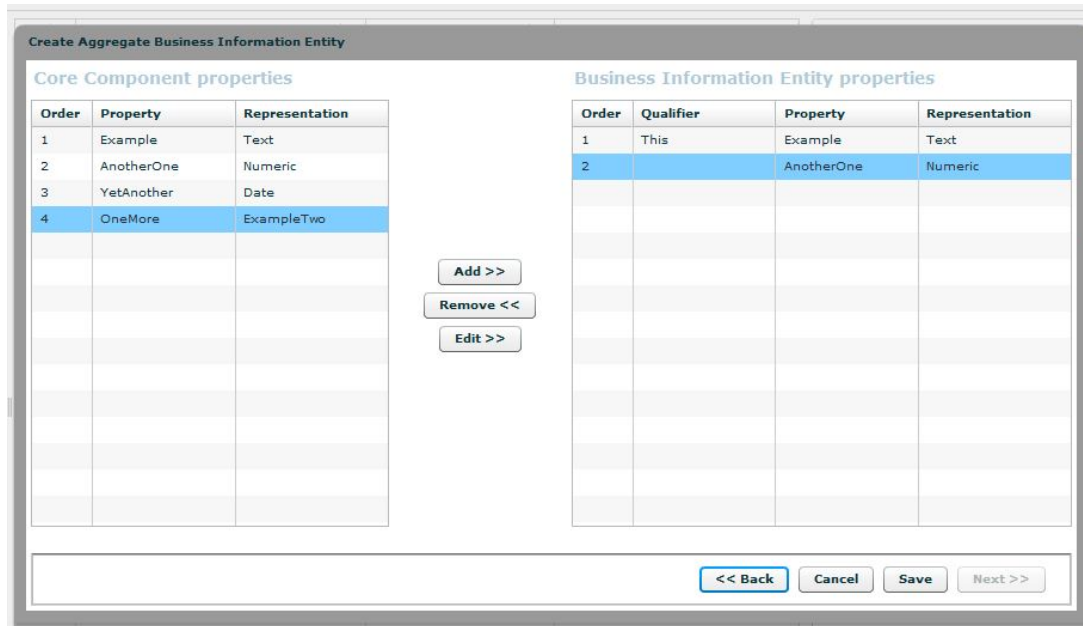


Figure 12: Create Aggregate Business Component

3.3.3 Business Documents

The Business Documents tab contains the following functionality:

- **Add Business Document**
- **Add Business Components to Business Document**

Business Documents may only contain Business Components. The Business Documents tab displays the ABIEs in a tabular form, that can be expanded (Figure 13).

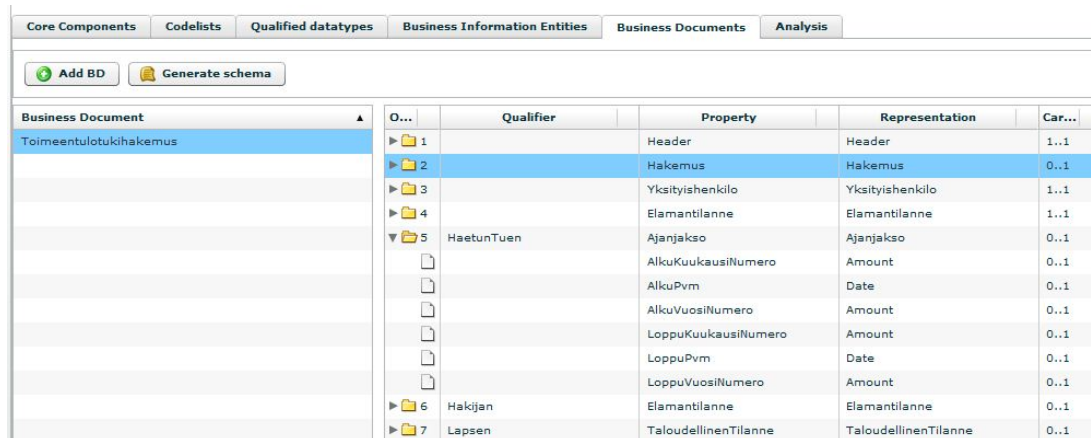


Figure 13: A Business Document opened for editing

Business document properties (ASBIEs) may have custom qualifiers and must have definitions. However, there is no possibility to add new BBIEs (i.e. document-specific properties) directly to the Business Document or qualify existing BBIEs.

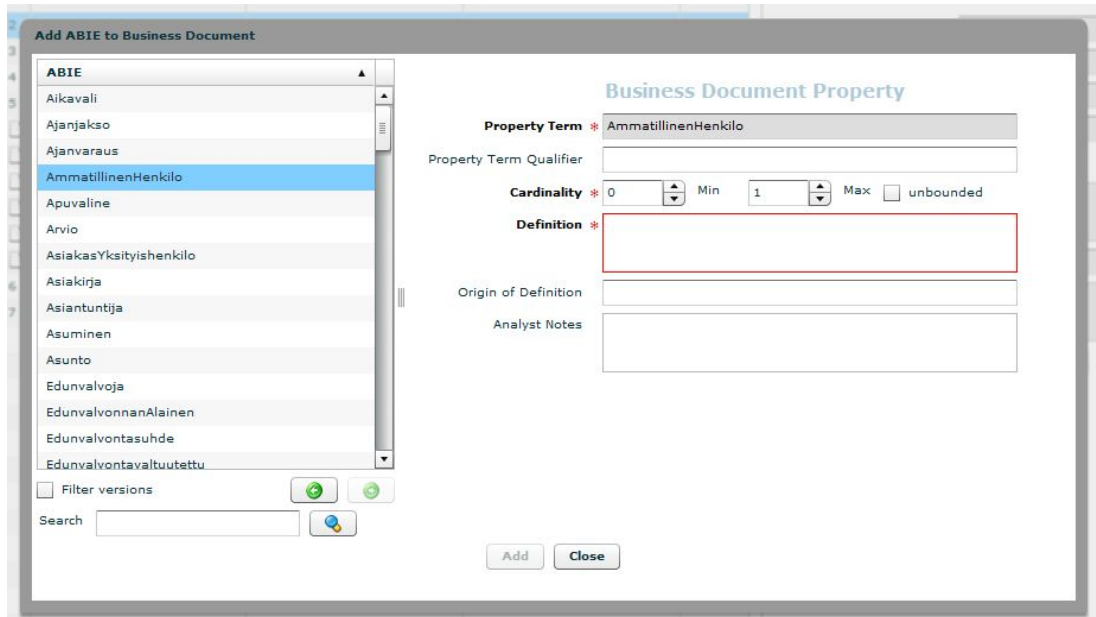


Figure 14: Add ABIE to Business Document

ABIEs are added to the Business Document through the “Add ABIE” form that displays all of the ABIEs available in the data model.

3.3.4 Analysis

The analysis tab contains analysis functionality for Core Components, Business Components and Business Documents.

Core Component analysis:

- Show difference between two Core Components
- Show Business Information Entities based on a certain ACC
- Show related Core Components
 - Ancestors
 - Descendants

Business Component Analysis:

- Show differences between two BIEs
- Show related Business Documents
 - Ancestors
 - Descendants
- Show Business Information Entities

Business Document Analysis:

- Show difference between two business documents

3.3.5 Schema generation

MDW generates XML schemas automatically from the ontology model (Figure 15). The schemas are generated from the model through a series of SPIN queries, SPARQL scripts and XSLT transformations. The procedure for generating XML schemas is rather complicated due to the multi-layered ontology model used in MDW.

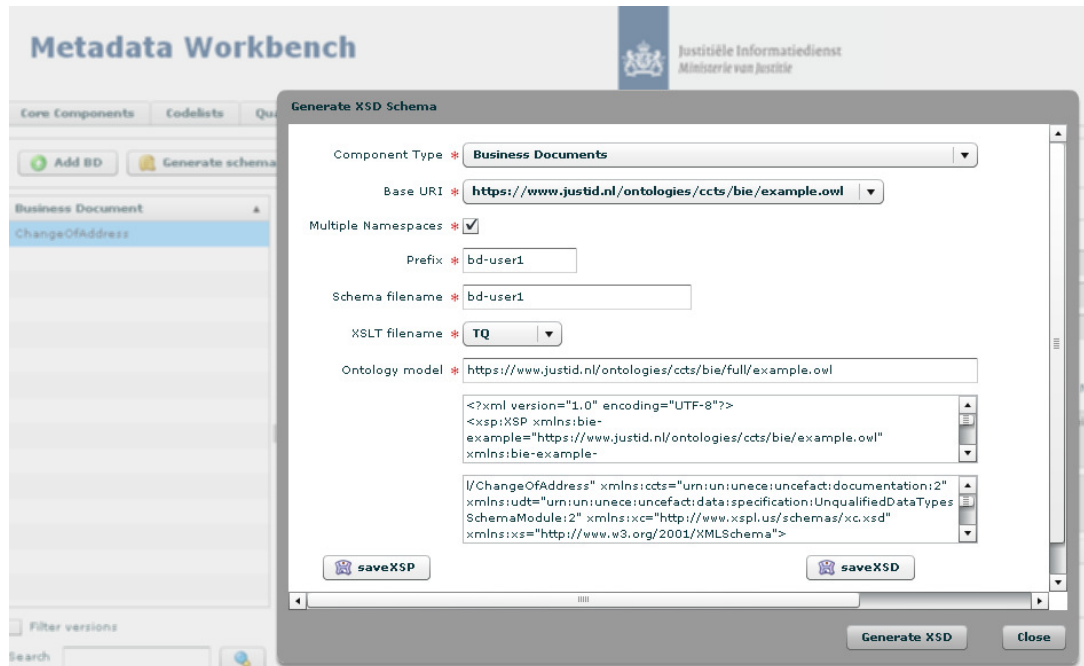


Figure 15: Schema generation

The complexity of schema generation is, however, abstracted through the user interface. XML Schemas can be generated from Datatypes, Core Components, Business Component and Business Documents. Schemas can be saved automatically to a specified location or copied directly from the user interface.

4 Summary

MDW was tested in the Tikesos project and its suitability for the management of data model developed for Finnish social services was analyzed. The tool implements much of the functionality required for the development and management of the data model artifacts. Among other benefits of MDW, the following are highlighted:

+ *Automatic generation of schemas.* In the Tikesos project the generation of XML schemas is semi-automatic, i.e. some of the steps required for producing schemas are manual. MDW makes it possible to generate CCTS-compliant XML schemas directly from the data model. Some modifications to the generation process would be required in order to comply with Finnish public administration recommendation JHS XML.

+ *Analysis of components.* Analysis functionality lists all of the ancestors and descendants of the component. This provides a means to analyze the usage of a component and specifies how it is related to other components.

+ *Abstraction over complex models.* The user interface of the tool facilitates the development of data components and business document structures without having to constantly control and remember the dependencies between different components. In the data model for social services there are some 200 data components with over a 1000 properties, which makes it difficult to keep track of them for a data analyst.

+ *Very adaptable semantic models.* The underlying data model can be easily extended, for example for adding more metadata to the components or implementing functionality missing from the tool.

The flexibility of the tool has also several implications which could be considered as drawbacks:

– *Complex models.* The metamodel built using the 3-layer approach is rather complex because it has to encompass all features of the CCTS data model.

– *High learning curve.* Due to the complexity of the underlying model it could be challenging for a developer without prior knowledge of the tool to understand its architecture in order to add missing functionality.

– *Partially closed source.* The source code of MDW is not fully open, as it is based on the proprietary TopBraid suite.

4.1.1 *Missing functionality*

MDW contains already much of the functionality required in the management of the components developed in the Tikesos project. However, there are several additional needs that arise from the differences in the context and modeling process of the business documents. The metadata model for MDW and the process for creating core components were developed so that the CCTS Core Component model can be generated from the rich ontology that is created by the content specialists. MDW has been designed with different evaluation approaches than those that are needed for managing core components in the context of Finnish social services.

Requirements for a tool that satisfies the need of change/version management in the context of social services will be analyzed in a more detailed report. Some of the missing functionalities that have been found during the testing of the MDW are specified and divided in two groups considering the effort that is needed to implement the functionality:

Major

- **Registration status**
Every component must be assigned with registration status for monitoring the quality and relevance of the component. Progression from candidate component to preferred standard is not clear. (Annex 1)
- **Versioning of individual components**
Possibility to compare superseded and current versions of the components. Minor changes as well as the registration status of the components could be implemented and tracked with an `rdf:Statement`.
- **Add Business Document specific BBIEs**
Possibility to add business document specific BBIEs that are only present in the context of a certain business document.
- **Add Business Document specific cardinalities and definitions to ABIEs and containing BBIEs**
Possibility to add additional definitions and redefine cardinalities in the components.

Minor

- **Dependency/Analysis reports**
Create Analysis reports listing all of the affected components from different scenarios:
 - adding a new field
 - making a field deprecated
- **Revision reports**
Create a revision report about minor and major changes, showing new and deprecated fields and the affected components.

Annex 1

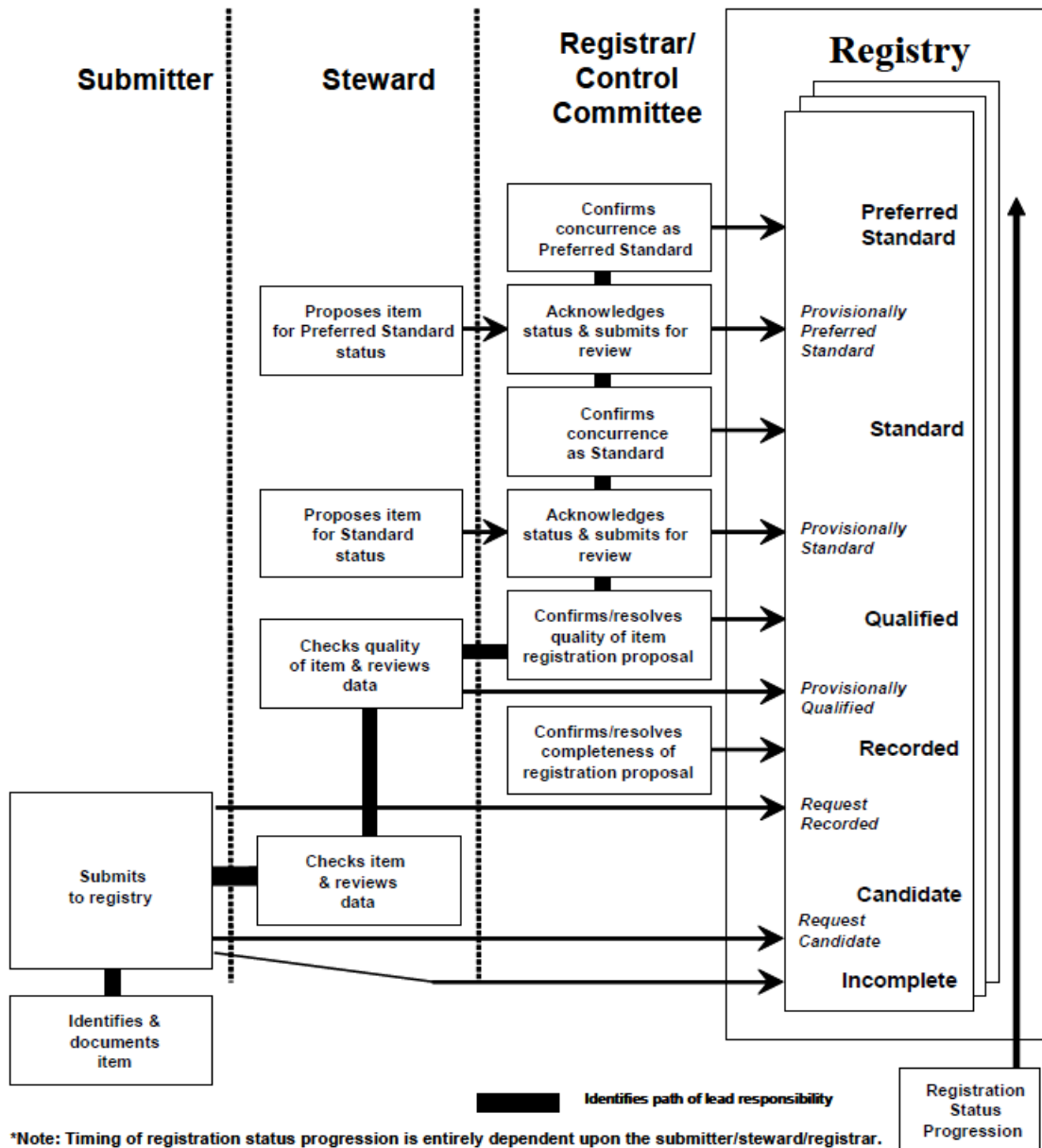


Figure 16: Registry process defined in ISO 11179