Submitted by

**Brigitte Andorfer-Plainer, BSc**

Submitted at

**Department of Business Informatics – Data & Knowledge Engineering**

Supervisor

**o.Univ.-Prof. DI Dr.**

**Michael Schrefl**

Co-Supervisor

**Ilko Kovacic, MSc**

Month Year

(For Information: Month and year of submission to Examination and Recognition Services)



Master Thesis

to obtain the academic degree of

Master of Science

in the Master’s Program

Business Informatics

SemNOTAM Container Management and its Integration with a Briefing Application

Eidesstattliche Erklärung

Ich, Brigitte Andorfer-Plainer, erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, Date

Signature

Acknowledgements

Abstract

Zusammenfassung

Table of Contents

1 Introduction 1

1.1 Preface 1

1.2 Problem statement 1

1.3 Outline 1

2 Fundamentals 2

2.1 SemNOTAM 2

2.2 BEST 2

2.3 Knowledge-Based Systems 2

2.4 Existing applications 2

3 Requirements Analysis 3

3.1 Task Description 3

3.2 Requirements 5

3.3 Challenges and Problems 7

4 Design 8

4.1 Approach 8

4.2 Data Model 8

4.3 Architecture 8

5 Implementation 12

5.1 Technologies 12

5.2 Container Ontology 12

5.3 Container Management Application 13

5.4 Container Description Service 14

6 Application Usage/Demonstration 16

6.1 Container Ontology 16

6.2 Container Management Application 16

7 Conclusion 17

Appendix 23

# Introduction

## Preface

## Problem statement

## Outline

The following thesis is structured as follows: First, an overview of the fundamentals is given, followed by the definition of the task and the requirements analysis. Afterwards, the approach for fulfilling the defined requirements and the architecture are introduced. Then the implementation is presented, and the usage of this application is demonstrated. Finally, an outlook and a conclusion are given.

Section 2 gives an overview of the fundamentals of this thesis. Therefore, the basic concepts of the projects SemNOTAM (Section 2.1) and BEST (Section 2.2) are described. Furthermore, the main concepts of knowledge-based systems are described (Section 2.3). Finally, the existing applications are introduced, and their relationship with the container management application is shortly described (Section 2.4).

In Section 2.4.1 first, an overview about the task is given (Section 3.1), and then it is divided into the three main parts container ontology (Section 3.1.1), container management application (Section 3.1.2), and container description service (Section 3.1.3). Based on this task description, requirements are derived and categorized (Section 3.2). Furthermore, challenges and problems that might occur during the implementation are described (Section 3.3).

Section 4 describes the design process. The approach that is used for the different parts of the application (Section 4.1), and the resulting data model are described (Section 4.2). Moreover, the architecture of the application is described for the client application as well as the server application (Section 4.3).

The implementation is covered in Section 5. Therefore, the used technologies for the ontology, the management application, and the services are described (Section 5.1). First, the implementation of the ontology is presented (Section 5.2) followed by the implementation of the container management application (Section 5.3). The last part describes the container description service, which is used by the container management application as well as the briefing application (Section 5.4).

Section 6 covers the different usages of the application. First, the administration of the ontology is demonstrated (Section 6.1), followed by the description of the use of the container management application (Section 6.2).

Finally, this thesis is summarized in conclusion and gives an outlook on future work (Section 7).

# Fundamentals

In this section, the fundamentals for understanding the purpose and goals of this thesis are described. Therefore, the main concepts of the SemNOTAM project and its architecture are introduced. This also includes the description of the different kind of inputs, e.g., Notices to Airmen (NOTAMs) and interest specifications. Furthermore, the purpose of the project BEST is described, to understand the concept of data containers. Afterwards, an overview of knowledge-based systems is given, which is the foundation for the definition of data containers. The last section introduces the existing applications, which are used by or will use the implementation of this thesis.

## SemNOTAM

Method: Rules, Filtering (Relevance, Importance)

Architecture

Notams, Interest Specifications, ObjectLogic

## BEST

SWIM Environment

Data containers

## Knowledge-Based Systems

Ontologies, Reasoning, Subsumption Hierarchy

## Existing applications

This section gives an overview of the applications that are used by or use the Container Management Application and the Container Description Service. As described in Section 2.1 the aim of the SemNOTAM project is to enable the intelligent filtering of NOTAMs. To provide this functionality a webservice was developed, which is also used by the Container Management Application for secondary containers. As this thesis also covers the integration of a briefing application, the SemNOTAM Briefing Application is also described.

### SemNotamWebService

different evaluate methods 🡪 IS + Notams, only IS (call Container Description Service)

### SemNOTAM Briefing Application

evaluate specific flight 🡪 SemNotamWebService evaluate only IS

# Requirements Analysis

This section contains a detailed task description, which defines the parts of this thesis. This task description serves as a base for the definition of requirements. Furthermore, this section shows possible problems that might occur. Parts in the other sections will then refer to the identified requirements when describing the architecture (Section 4.3) and the implementation (Section 5).

## Task Description

A management application for data containers should be developed. A data container can either represent a primary container which does not have another data container as a data source or a secondary container which uses other primary or secondary containers as data source. Regardless of the type, a data container consists of the data set including the data items and a semantic label representing metadata. The metadata can be either used to describe the membership conditions fulfilled by the data items in the data set, i.e., descriptive metadata, or to capture other technical, quality, and provenance aspects of the data container, i.e., administrative metadata.

The semantic label includes the descriptive metadata, such as the time period or the spatial area covered by the dataset. The semantic label, as well as the descriptive metadata, shall be represented by concepts within an ontology. The concepts of the descriptive metadata are assigned to semantic labels by so-called facets. The basic idea of facets is to describe features of the content. To do so, facets are used to assign concepts to the semantic label. Each data item in a data set has to fulfill the assigned concepts, and all existing data items which fulfill the concepts must be included in the dataset. For example, assigning the concept Austria over the spatial facet to a semantic label indicates that the data set contains all existing data items which can be assigned to Austria, or even more.

Since the concepts shall be defined in an ontology, it will be possible to define specialization respectively generalization hierarchies between two concepts explicitly or to derive them by employing subsumption reasoning. Following this approach, it will be possible to derive specialization hierarchies between semantic labels and since semantic labels describe the content of the data container it also allows to draw subsumption hierarchies between the data containers. The descriptive metadata concepts can also be used for semantic labels which represent information needs. Applying subsumption reasoning allows finding the most specific superset respectively semantic label of a data container for a given semantic label concept representing an information need. Since multiple data containers can represent the most specific superset, the administrative metadata can be considered to decide which superset shall be selected, e.g., select the superset which contains most-recent data items, has the smallest size, or the one with the right encoding. This additional information represented by the administrative metadata shall not be used for reasoning purposes, which is why it shall not be represented by concepts in the ontology instead it shall be annotated in some way.

The communication for the creation of a container shall be implemented through web services to enable the integration with the SemNOTAM Briefing Application. The management application and the briefing application both need to be able to select concepts for facets. Furthermore, both systems need to build interest specifications based on the selected concepts and use this interest specification as input for the SemNotamWebService. The difference between the two systems is that in the management application the user shall decide which data container shall be used as data source. The briefing application is not interested in the used NOTAM data. Therefore, one service shall find the most specific superset in the ontology, based on the subsumption hierarchy between containers and their administrative metadata.

To support these and other tasks an ontology, a management application for data containers, and a web service project shall be developed. These three subsystems shall fulfill the following tasks:

### Ontology Management

The semantic label of a data container includes the descriptive metadata which shall be represented by ontology concepts. The facets are also represented as concepts, and the descriptive metadata concepts are defined as their sub-concepts. Moreover, each descriptive metadata concept has an interest representation annotated. The administrative metadata is assigned to the punned instance of a data container. The ontology management shall support create, modify, and delete operations on concepts and their annotations.

### Container Management Application

An application for managing data containers shall be implemented. This application should enable the user to view, modify, delete, create, and refresh data containers respectively their metadata.

View: The user shall be able to view the existing containers which are available in the corresponding ontology. The list of containers shall display the descriptive metadata, and a detailed view of a single container shall display descriptive and administrative metadata.

Modify: Furthermore, the user shall be able to modify some of the administrative metadata, e.g., the refresh interval and the refresh until timestamp. If other metadata shall be modified, e.g., descriptive metadata, then this shall be prevented, and it shall be indicated that a new container shall be created for this purpose.

Delete: The user shall also be able to delete a container, but in this case, possible existing dependencies to another container must be considered, e.g., the user should be informed that the data container might act as a data source for another container.

Create: For the creation of new containers there shall be an interface where the user can select a concept for each facet. The user can choose between creating a primary container or a secondary container.

For primary containers, the corresponding dataset has to be declared explicitly, since they do not have a data source. Therefore, no other primary or secondary data container shall be selected. The user shall ensure that the selected dataset includes all the data items specified by the container facets respectively the administrative metadata.

For the creation of a new secondary container, first, its most specific superset shall be found. The user shall be able to decide which superset should be used. Furthermore, a service that is used for filtering the data set has to be selected by the user. The dataset of the subsuming container is used as input for the selected service which filters it based on the selected concepts and if existing on the specific interest. The output of the service represents the data set of the created container. The service and the parameters that are used for calling it are saved in the ontology to enable the refreshing of the container.

A data container shall be created within a repository. A repository includes the ontology and the data sets of the existing data containers. Repositories shall allow creating “public” containers and “private” containers, e.g., for specific flights. These repositories shall be considered during the data container search. The implementation only includes one repository, which acts as public repository. However, the possibility to select another repository shall be given.

Refresh: The refreshing of containers shall only be implemented as a manual task. The automatic refreshing, however, shall be considered as well. Furthermore, different variants of refreshing shall be analyzed such as the decentral/central or the push/pull refresh process. The basic steps of the refresh process can be summarized as follows: When a refresh is initiated the qualitative data of the container and its data source container must be checked, whether there has been a refresh of the data source and whether the data source contains new data. If a refresh is needed, then the new data items must be filtered with the service and the parameters of the container and the data container with its metadata shall be updated.

### Container Description Service

To create a container description, the user shall be able to select concepts for the different facets. Therefore, a service to select a concept for one specific facet, which only provides concepts for this facet, must be provided. Furthermore, it shall be possible to select other ontologies which include other concepts for this facet.

The SemNOTAM system requires an interest specification as input to filter a set of DNOTAMs. This means that the container description needs to be translated to its corresponding interest specification representation. Therefore, a service is needed which realizes the mapping between a concept and its interest representation. Therefore, the annotated interest representations of the concepts in the ontology are used.

Another service shall enable the user to find the most specific superset of his/her information need respectively specified interest specification. Therefore, a service for retrieving the most specific superset shall be implemented. The service needs to map the interests back to its corresponding concepts since interests itself do not store the concept information. The service shall find all possible most specific supersets and decide which one to use based on the administrative metadata.

## Requirements

The task description of the previous section is used to specify the requirements. These requirements are first grouped according to which of the three subsystems (Container Ontology, Container Management Application, and Container Description Service) they belong. A second grouping is done according to the part or functionality of the subsystem they describe.

1. Container Ontology
   1. Container
      1. A data container must be represented by a class and an individual, which have the same IRI.
      2. It must be possible to assign descriptive and administrative metadata to a data container.
      3. It must be possible to create a subsumption hierarchy of the existing data containers.
   2. Descriptive Metadata
      1. A single characteristic is represented by a concept in the corresponding facet.
      2. The different types of descriptive metadata must be assigned to facets, which are represented as classes.
      3. A data containers’ descriptive metadata must be described by the combination of one concept of each facet.
      4. The concepts of the facets must be organized hierarchically, to enable subsumption reasoning.
   3. Administrative Metadata
      1. The administrative metadata of the data container must be assigned to the individual of the data container.
      2. For secondary data containers, the data source container must be referenced through an object property.
      3. The other administrative metadata is assigned through data properties.
      4. Each data container needs to have different timestamps assigned to enable the comparison of their quality.
2. Container Management
   1. View container
      1. The application must enable the user to see a list of all containers in an ontology.
      2. The list of containers must show the descriptive metadata.
      3. The application must enable to view the details of a specific container, which includes its descriptive and administrative metadata.
   2. Create container
      1. The application must enable the user to create new containers.
      2. The user must be able to specify the descriptive metadata using concepts for the facets.
      3. The user must be able to define parts of the administrative metadata, i.e., data format, data encoding, refresh interval and refresh until timestamp.
      4. For the creation, the user must be able to decide if a primary or secondary container should be created.
      5. For primary containers, the user must be able to upload the data items for the new container.
      6. For secondary containers, the user must be able to select the superset data container to be used.
      7. The descriptive metadata and parts of the administrative metadata must be shown for the possible superset data containers.
      8. The user must be able to decide which service shall be used for filtering the data items of the superset data container.
   3. Modify container
      1. The user must be able to modify the refresh interval and the refresh until timestamp.
   4. Delete container
      1. The user must be able to delete containers of an ontology.
      2. The application should inform the user that containers may serve as data source for other containers.
   5. Refresh container
      1. The application must enable the user to initiate a refresh for a container.
      2. For secondary data containers, the refresh shall only be started when it is needed. This means that the refresh until timestamp is not reached yet and the data source container contains new data items.
      3. When refreshing a primary data container, the user must be able to upload the file with the current data items for this data container.
3. Container Description Service
   1. selectConceptForFacet
      1. The service has a facet IRI as input parameter and returns a list of concept IRIs.
      2. The service must return all possible concepts for the given facet.
   2. getInterestForConcept
      1. This service has the ontology IRI and a concept IRI as input parameter.
      2. It must return the interest representation for the given concept in the given ontology.
   3. findMostSpecificSuperset
      1. This service has an interest specification as input parameter and returns a feature collection.
      2. The interests in the interest specification must be mapped to its corresponding concepts.
      3. For the given interest specification, a data container must be created, described by concepts in the facets.
      4. The service must find the most specific superset in the main ontology and return its feature collection.
      5. For the decision which data container to use the administrative metadata shall be considered.

In the further sections, especially the architecture and the implementation, these requirements will be referenced.

## Challenges and Problems

Container in different repositories 🡪 copying the description from the private ontology to the public ontology

The realization of different repositories, e.g., private and public repositories, leads to a challenge concerning the data discovery. When a user wants to find the superset data container, first it shall be checked if the private repository contains a relevant container. Then this container shall be used and otherwise the superset container has to be found in the public repository. Therefore, the container description of the information need has to be defined in both ontologies. As the public repository shall not contain data containers for a specific information need, e.g., a specific flight, the data container of the information need is only kept in the ontology of the user’s private repository. As this data container saves a reference to its superset data container there are two possibilities:

1. The data container has to be copied to the private repository to enable the link of the user’s data container and its superset data container.
2. The reference to the superset data container must contain the information which data container is referenced and in which ontology this data container can be found, respectively its full IRI.

The first option only moves the problem from the new data container to the superset data container, because then the refresh of this data container is not possible anymore. To keep the superset data container up-to-date either a central or a decentral approach can be followed. For the central approach, the data container in the public repository has to save all its representations in private repositories and forward the new data items to them. This would mean that the public repository had to know all the existing private repositories which is not the purpose of public and private repositories.

The decentral approach would mean that the data container in the private repository has to check if it’s representation in the public repository has changed. Therefore, the data container only has to access the public repository and check the container with its corresponding IRI.

The saving of the reference to the superset data containers in the ontology cannot be done with object properties, but a data property or annotation can be defined. With this solution the number of refreshment checks is not increasing, and the public data containers stay in the public repository.

As the implementation only considers one repository and has no distinction between public and private repositories, this problem is only relevant for further implementations.

Another challenge, which also regards this implementation, is the access to the ontology. The Container Management Application as well as the Container Description Service have to have access to the ontology. In this implementation the Container Management Application and the Container Description Service are located on the same device. Therefore, both applications can use absolute paths to access the ontology and the files of the data containers. To consistently have only one system that accesses the ontology, the whole communication with the ontology would have to be implemented in the webservice. This would mean that each time the Container Management Application needs information about a container a webservice call has to be executed.

# Design

This section describes the design process of this thesis. First, the approach for the realization of the whole system is introduced. This system consists of the Container Ontology, the Container Management Application, and the Container Description Service. Afterwards, the data model gives an overview of the relationship between the data of the different subsystems. Moreover, this section shows the architecture of the client and the server application.

## Approach

### Container Ontology

match between task description and definition of an ontology

#### Data Container

Ontology classes 🡪 hierarchy

descriptive metadata 🡪 describe content 🡪 hierarchy

administrative metadata 🡪 describe quality 🡪 comparison of provenance

#### Descriptive Metadata

facets as classes

concepts as subclasses of facets

object properties 🡪 assigning descriptive metadata to data container

concepts described by data properties 🡪 e.g. time period, wingspan, weight, …

#### Administrative Metadata

data properties 🡪 dates, data volume, data location, …

object property 🡪 data source

### Container Management Application

Web Application, Design pattern Model-View-Controller

Briefing Appllication uses Vaadin 🡪 also use Vaadin – they mention Model-View-Presenter

overview Page; Details Page (Modify, refresh, delete); Create Form (Primary, secondary)

### Container Description Service

## Data Model

## Architecture

### Client-Application

#### Technology Layer

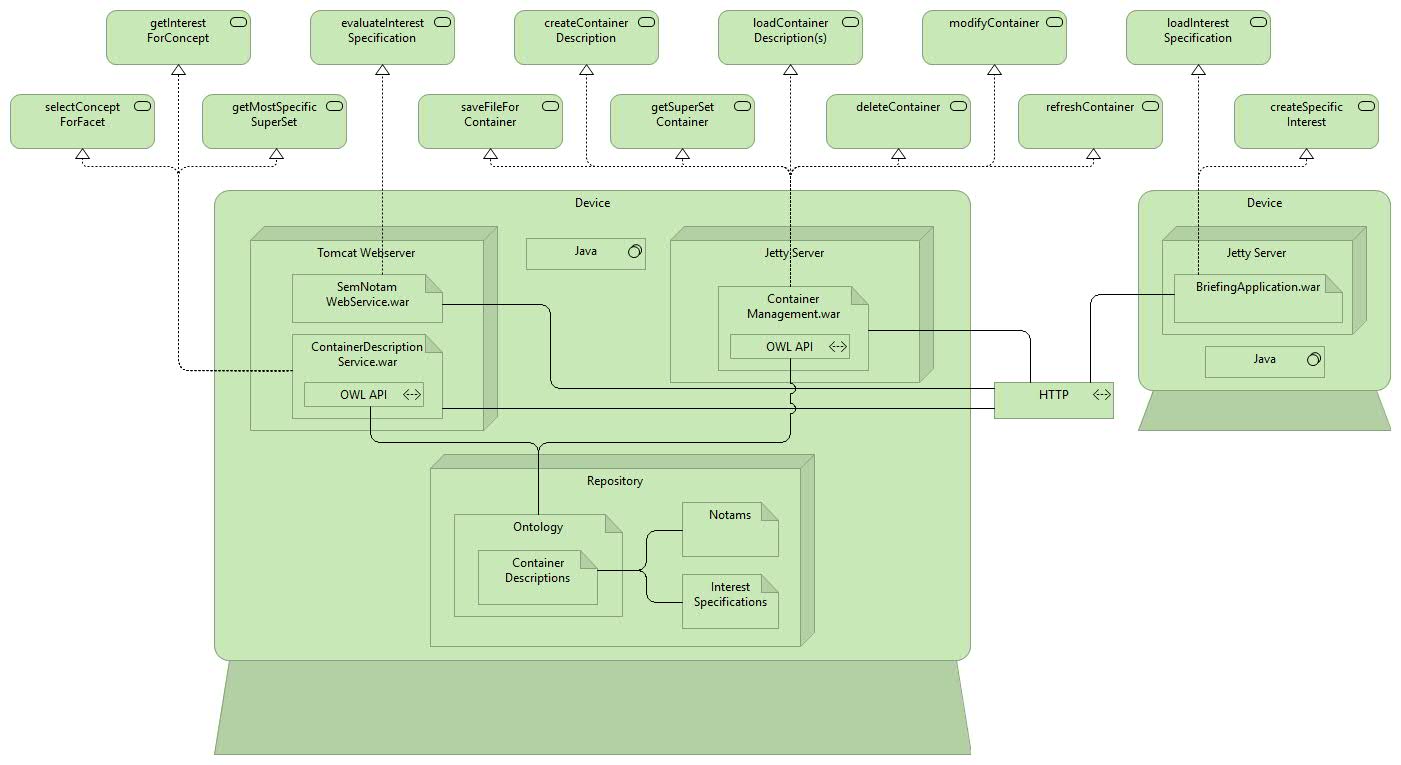


Figure 4.1: Client Application – Technology Layer

#### Application Layer

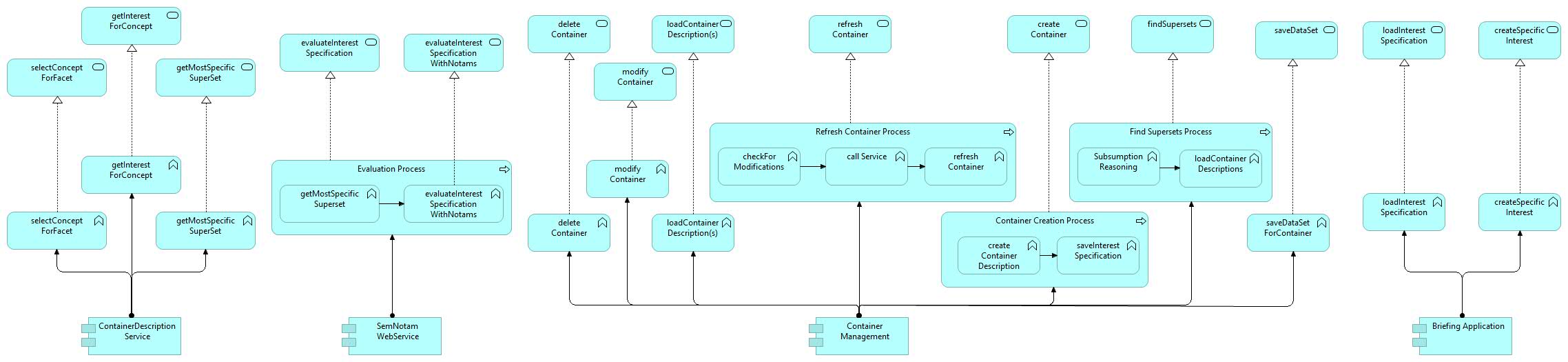


Figure 4.2: Client Application – Application Layer

#### Business Layer

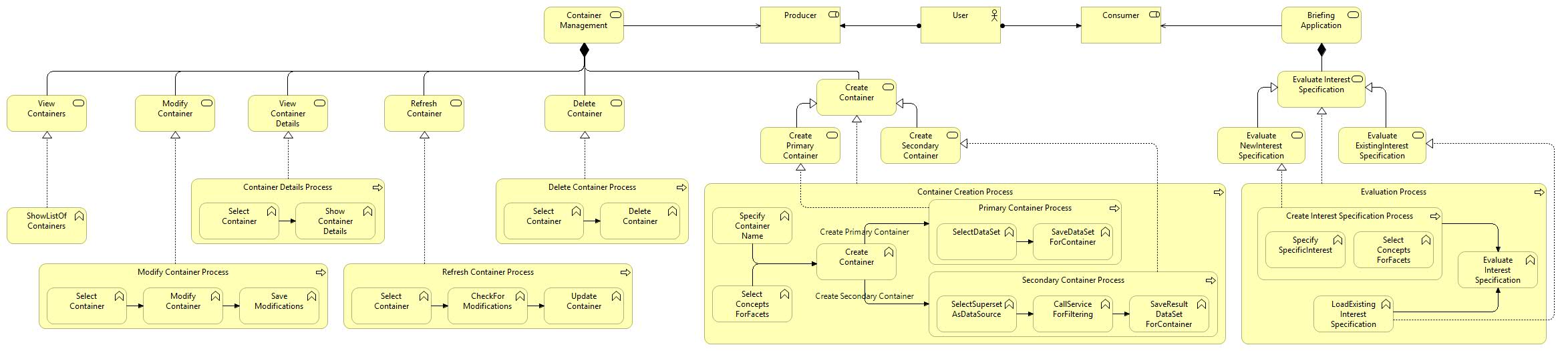


Figure 4.4: Client Application – Business Layer

### Server-Application

#### Technology Layer

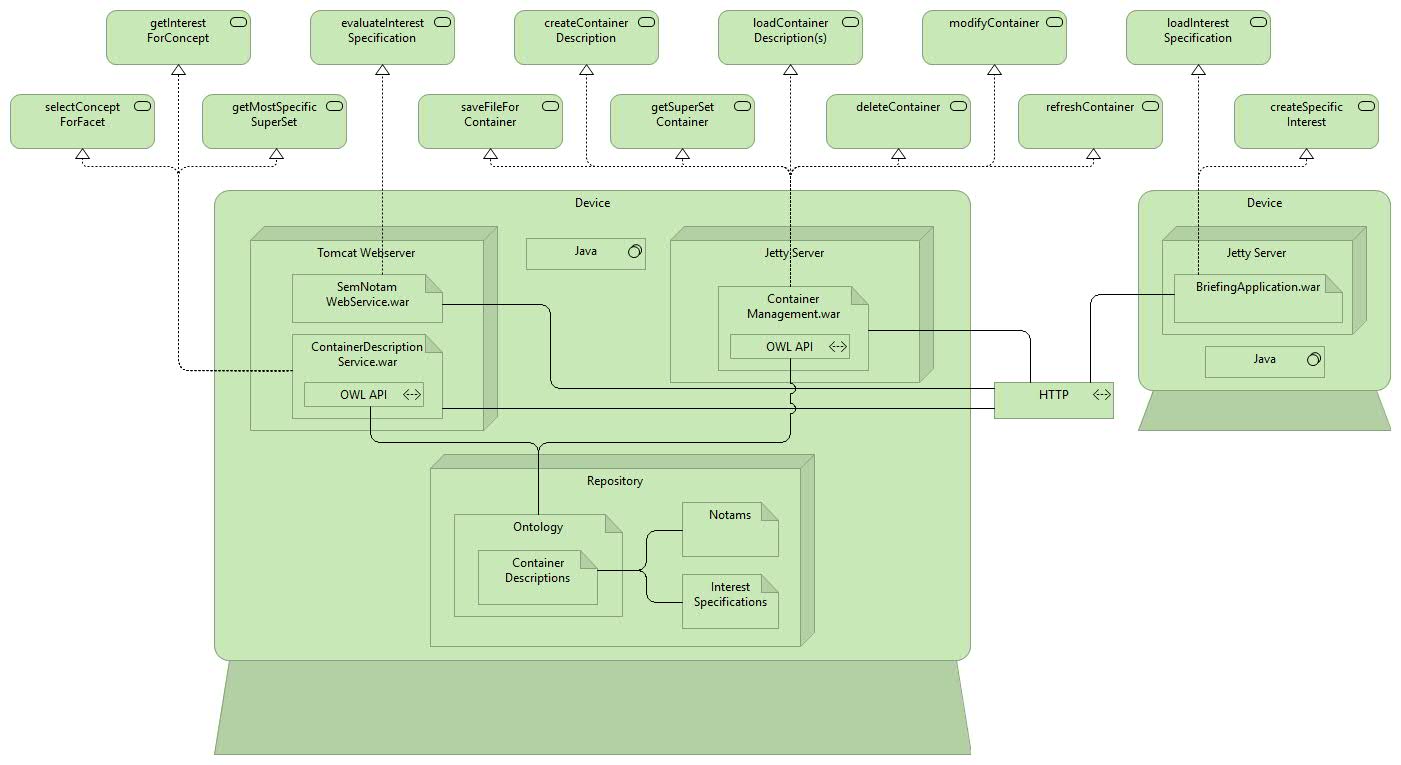


Figure 4.5: Server Application – Technology Layer

#### Application Layer

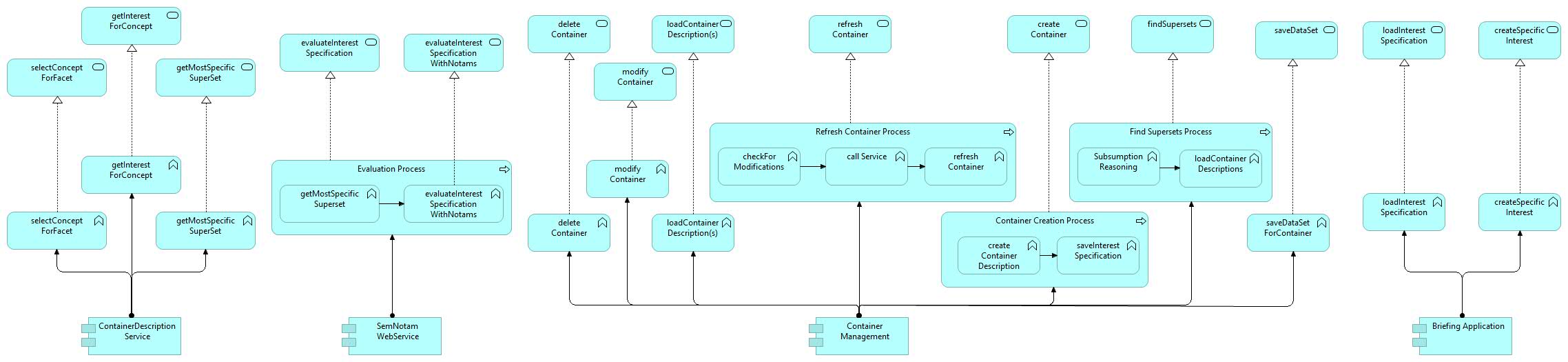


Figure 4.6: Server Application – Application Layer

#### Business Layer

There is no business layer for server applications.

# Implementation

The implementation of the defined requirements (Section 3.2) and the resulting architecture (Section 4.3) is described in this section. First, an overview of the technologies, which are used for the ontology, the client application, and the server application, is given. Section 5.2 covers the implementation of the container ontology. The container management application, which is the client application, is described in Section 5.3. The last section, Section 5.4, documents the implementation of the server application, respectively the container description service.

## Technologies

In this section, an overview of the technologies that are used for the implementation is given. The Web Ontology Language (Section 5.1.1) is used for the realization of the container ontology and Protégé (Section 5.1.2) is the tool that is used for the creation and administration of the container ontology. For the container management application and the container description service the programming language Java is used (Section 5.1.3). Furthermore, the container management application uses the framework Vaadin (Section 5.1.4), whereas, the container description service uses the web service framework Apache CXF (Section 5.1.5).

### OWL

[1]

### Protégé

[2]

### Java

[3]

### Vaadin

[4]

### CXF

[5]

## Container Ontology

As described in Section 4.1.2 data containers will be represented by an ontology modeled in the Web Ontology Language (OWL) by using the tool Protégé. The implementation of this ontology is divided into three parts. The first part (Section 5.2.1) describes how the facets are represented in the ontology. This also includes the modeling of the hierarchy between the concepts of the different facets. Afterwards, Section 5.2.2 covers the realization of the data containers, which consists of their representation as class and individuals, and the definition of the descriptive and administrative metadata. Finally, the data discovery is explained in Section 5.2.3, which enables the creation of a hierarchy between data containers.

### Facets

Concepts (Requirements 1.2.1, 1.2.2, 1.2.4)

Data Properties – has… (Requirements1.2.4)

### Container

Data Container (Requirements 1.1.1, 1.1.2)

Object Properties – Facets (Requirement 1.2.3)

Object Properties – dataSource (Requirements 1.3.2, 1.1.3)

Data Properties (Requirements 1.3.1, 1.3.3)

Data Properties – timestamps (Requirement 1.3.4)

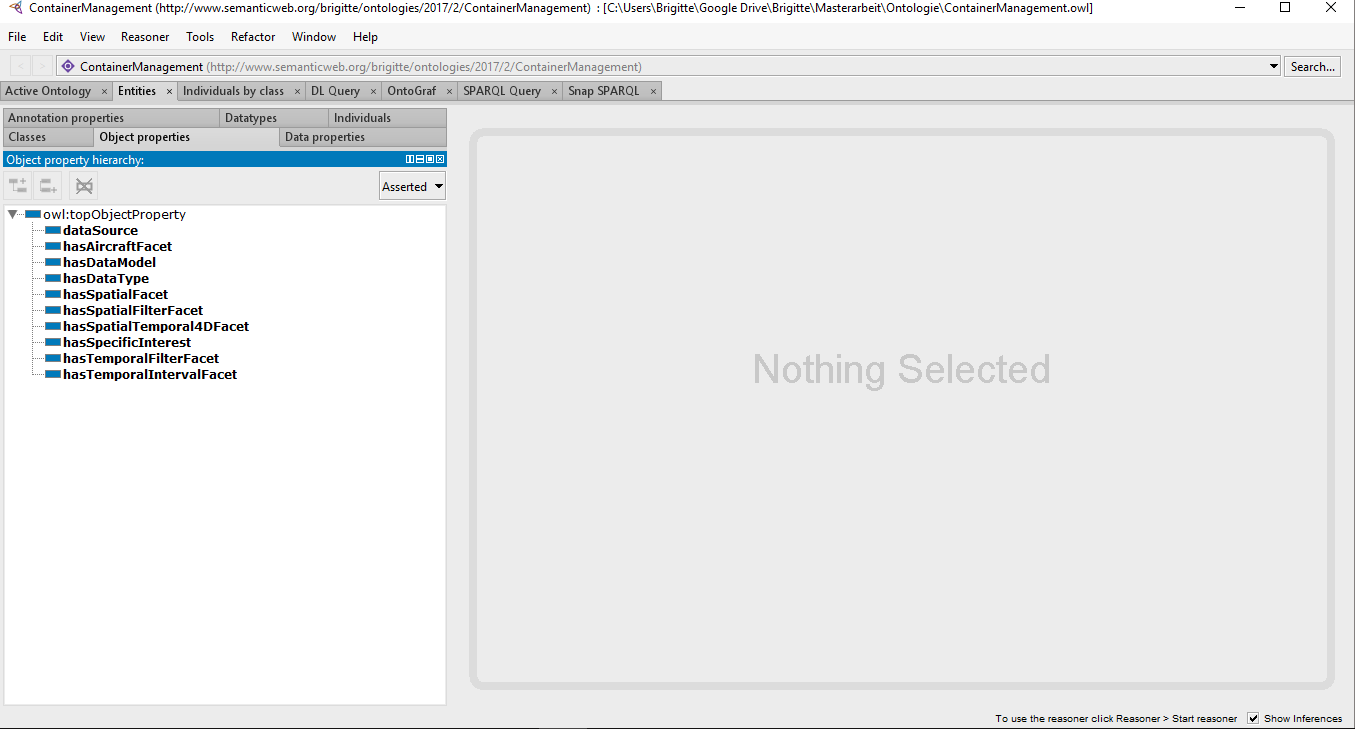


Figure 5.1: Container Ontology – Object Properties

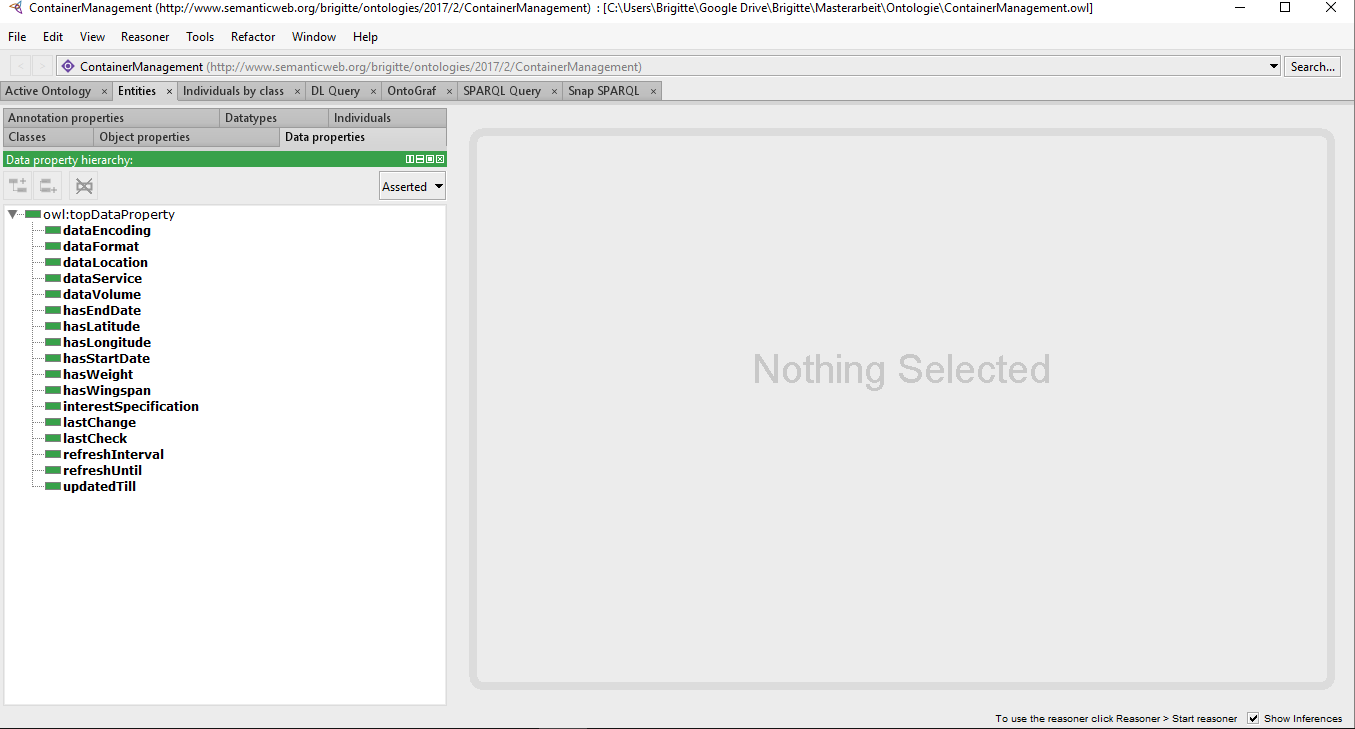


Figure 5.2: Container Ontology – Data Properties

### Discovery

Requirements (1.1.3)

## Container Management Application

### List of available Containers

Requirements 2.1.1, 2.1.2

### Details of a Container

Requirements 2.1.3

### Deletion of a Container

Requirements 2.4.1

Info Message – Requirement 2.4.2

### Editing a Container

Requirement 2.3.1

### Refreshing a Container

Requirement 2.5.1

Secondary Container – Requirement 2.5.2

Primary Container – Requirement 2.5.3

### Creation of a Container

Requirements 2.2.1, 2.2.2, 2.2.3, 2.2.4

#### Primary Container

Requirement 2.2.5

#### Secondary Container

Requirements 2.2.6, 2.2.7, 2.2.8

## Container Description Service

### selectConceptForFacet

Requirements 3.1.1, 3.1.2

### getInterestFromConcept

Requirements 3.2.1, 3.2.2

### getMostSpecificSuperset

Requirements 3.3.1

Interest spec to data container (Requirements 3.3.2, 3.3.3)

find superset (Requirements 3.3.4, 3.3.5)

# Application Usage/Demonstration

In this section, the usage of the implemented system is demonstrated. The first part describes the usage of the container ontology, which consists of the management of the concepts for the different facets. The second part shows the usage of the container management application, which can be used by providers as well as consumers of data containers.

## Container Ontology

## Container Management Application

# Conclusion

Bibliography

[1] “OWL - Semantic Web Standards.” [Online]. Available: https://www.w3.org/OWL/. [Accessed: 27-Apr-2017].

[2] “protégé.” [Online]. Available: http://protege.stanford.edu/. [Accessed: 27-Apr-2017].

[3] “Was ist Java?” [Online]. Available: https://java.com/de/about/whatis\_java.jsp. [Accessed: 17-Nov-2017].

[4] “Vaadin - Framework,” *Vaadin*. [Online]. Available: https://vaadin.com/framework. [Accessed: 17-Nov-2017].

[5] “Apache CXF -- Index.” [Online]. Available: http://cxf.apache.org/. [Accessed: 17-Nov-2017].

List of Figures

Figure 4.1: Client Application – Technology Layer 9

Figure 4.2: Client Application – Application Layer 9

Figure 4.3: Client Application – Business Layer 10

Figure 4.4: Server Application – Technology Layer 10

Figure 4.5: Server Application – Application Layer 11

List of Tables

List of Listings

Acronyms

# Appendix