Semantic Annotations for BPMN models:
Extending SeMFIS for supporting ontology reasoning and query functionalities

Aikaterini Dimitraki

Thesis submitted in partial fulfillment of the requirements for the

Master of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
University Campus, Voutes, Heraklion, GR-70013, Greece

Thesis Supervisor: Prof. Dimitris Plexousakis

Heraklion, November 2016

This work has been supported by the Institute of Computer Science of the Foundation for Research and Technology-Hellas (FORTH)
Semantic Annotations for BPMN models: 
Extending SeMFIS for supporting ontology reasoning and query functionalities

Thesis submitted by
Aikaterini Dimitraki
in partial fulfillment of the requirements for the 
Master’s of Science degree in Computer Science

THESIS APPROVAL

Author: ____________________________
Aikaterini Dimitraki

Committee approvals: ____________________________
Dimitris Plexousakis
Professor, Thesis Supervisor

______________________________
Irini Fundulaki
Principal Researcher, Committee member

______________________________
Antonios Savides
Professor, Committee member

Departmental approval: ____________________________
Antonis Argyros,
Professor, Director of Graduate Studies

Heraklion, November 2016
Abstract

Business process models are a sequential representation of all functions associated with a specific business activity. In implementing business process modelling, there are many techniques that have been tried and tested throughout the years. Business Process Modelling Notation (BPMN) is the global standard for process modeling and provides a standard notation that is readily understandable by management personnel, analysts and developers. The original intent of BPMN was to help bridge communication gaps that often exist between the various departments within an organization or enterprise, by providing a notation that is intuitive to business users, yet able to represent complex process semantics.

Here we will discuss the approach of enriching the existing standards, with semantic information provided by OWL ontologies. Business Process annotation with semantic tags taken from an ontology is becoming a crucial activity for business designers. In fact, semantic annotations help business process comprehension, documentation, analysis and evolution. For this purpose there is an ongoing research and several software tools have been developed to support this approach.

SeMFIS tool is an open and extensible platform for engineering semantic annotations of conceptual models. We decided to extend SeMFIS to support consistency check on OWL ontology models, created for annotate BPMN process models, with the use of HermiT reasoner and a feature to allow user uploading his ontology with its instances and properties on Openlink Virtuoso engine and query them with SPARQL language. Creating a meat supply chain process model and a meat supply chain OWL ontology that enriches the process model with information necessary for traceability purposes, helps us understand the need for annotations and for features which process them.
Σημαιολογικές υποσημειώσεις σε BPMN μοντέλα: Επεκτείνοντας το SeMFIS για την υποστήριξη εξαγωγής συμπερασμάτων και επερωτήσεων σε οντολογίες

Περίληψη

Τα επιχειρησιακά μοντέλα διαδικασιών είναι μια διαδοχική αναπαράσταση όλων των λειτουργιών που σχετίζονται με μια συγκεκριμένη επιχειρηματική δραστηριότητα. Για την υλοποίησή τους υπάρχουν πολλές τεχνικές που έχουν δοκιμαστεί και ελεγχθεί όλα αυτά τα χρόνια. Business Process Modelling Notation (BPMN) είναι το παγκόσμιο πρότυπο για τη μοντελοποίηση διαδικασιών το οποίο παρέχει μια πρότυπη μορφή που είναι εύκολα κατανοητή από διευθυντικά στελέχη, αναλυτές και προγραμματιστές. Ο πρωταρχικός σκοπός της BPMN ήταν να συμβάλει στη γεφύρωση των κενών επικοινωνιών που συχνά υπάρχουν μεταξύ των διάφορων τμημάτων στα πλαίσια ενός οργανισμού ή μίας επιχείρησης, παρέχοντας έτσι ένα συμβολισμό που είναι διασημιτικά κατανοητός για όλους τους εμπλεκόμενους στον επιχειρηματικό τομέα, και ακόμη είναι σε θέση να αναπαραστήσει πολύπλοκη σημαιολογία.

Εδώ θα συζητήσουμε την προσέγγιση του εμπλουτισμού των υφιστάμενων προτύπων, με σημαιολογική πληροφορία που παρέχεται από οντολογίες εκφρασμένες σε γλώσσα OWL. Οι επισημειώσεις επιχειρηματικών διαδικασιών με σημαιολογικές ετικέτες που λαμβάνονται από μια οντολογία, είναι ολόκληρα και πιο κρίσιμα θέμα για τους σχεδιαστές διαδικασιών. Στην πραγματικότητα, οι σημαιολογικές επισημειώσεις βοηθούν στην κατανόηση, την τεκμηρίωση, την ανάλυση και την εξέλιξη των επιχειρηματικών διαδικασιών. Για το σκοπό αυτό, υπάρχει εν εξελίξει έρευνα και πολλά εργαλεία λογισμικού έχουν αναπτυχθεί για να υποστηρίζουν αυτή την προσέγγιση.

Το εργαλείο SeMFIS είναι μια ανοιχτή και επεκτάσιμη πλατφόρμα για την κατασκευή σημαιολογικών επισημειώσεων πάνω σε εννοιολογικά μοντέλα. Αποφασίσαμε να επεκτείνουμε το SeMFIS ώστε να υποστηρίζει ελέγχο συνέπειας σε μοντέλα OWL οντολογιών, που έχουν δημιουργηθεί για υποσημείωση BPMN επιχειρησιακών διαδικασιών, με τη χρήση του εξαγωγείου συμπερασμάτων HermiT και ακόμη και την προσθήκη μιας επέκτασης που επιτρέπει την υποστήριξη υποσημειώσεων σε εννοιολογικά μοντέλα. Αποφασίσαμε να επεκτείνουμε το SeMFIS ώστε να υποστηρίζει ελέγχο συνέπειας σε μοντέλα OWL οντολογιών, που έχουν δημιουργηθεί για υποσημείωση BPMN επιχειρησιακών διαδικασιών, με τη χρήση του εξαγωγείου συμπερασμάτων HermiT και ακόμα και την προσθήκη μιας επέκτασης που επιτρέπει την υποστήριξη των οντολογιών της ελέγχος συνέπειας σε εννοιολογικά μοντέλα. Η δημιουργία ενός μοντέλου διαδικασιών της εφοδιαστικής αλυσίδας κρέατος και μίας οντολογίας σε OWL που εμπλουτίζει το μοντέλο με πληροφορίες αναγκαίες για την εργαλεία λογισμικού έχουν αναπτυχθεί για να υποστηρίζουν αυτή την προσέγγιση.

Το εργαλείο SeMFIS είναι μια ανοιχτή και επεκτάσιμη πλατφόρμα για την κατασκευή σημαιολογικών επισημειώσεων πάνω σε εννοιολογικά μοντέλα. Αποφασίσαμε να επεκτείνουμε το SeMFIS ώστε να υποστηρίζει ελέγχο συνέπειας σε μοντέλα OWL οντολογιών, που έχουν δημιουργηθεί για υποσημείωση BPMN επιχειρησιακών διαδικασιών, με τη χρήση του εξαγωγείου συμπερασμάτων HermiT και ακόμα και την προσθήκη μιας επέκτασης που επιτρέπει την υποστήριξη των οντολογιών της ελέγχος συνέπειας σε εννοιολογικά μοντέλα. Η δημιουργία ενός μοντέλου διαδικασιών της εφοδιαστικής αλυσίδας κρέατος και μίας οντολογίας σε OWL που εμπλουτίζει το μοντέλο με πληροφορίες αναγκαίες για την εκκεντρικότητα του κρέατος, μας βοηθά να κατανοήσουμε την ανάγκη για υποσημειώσεις και για εργαλεία που τις επεξεργάζονται.
Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω πολύ τον επιβλέποντα καθηγητή μου κ. Δημήτρη Πλεξουσάκη για την βοήθειά και την καθοδήγησή που μου προσέφερε στην εκπόνηση αυτής της εργασίας. Επίσης, ήθελα να εκφράσω τις ευχαριστίες μου στην κ. Ειρήνη Φουντουλάκη για την ιδέα της εργασίας και την πολύ ουσιαστική βοήθεια που μου προσέφερε καθ’ ολή τη διάρκεια καθώς και στον κ. Hans-Georg Fill για τις άμεσες και αναλυτικές απαντήσεις σε κάθε μου απορία. Ακόμη θα ήθελα να ευχαριστήσω τον κ. Αντώνη Σαββίδη για την προθυμία του να συμμετάσχει στην τριμελή επιτροπή. Να ευχαριστήσω επίσης το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την υποτροφία που μου προσέφερε, καθώς και για την πολύτιμη υποστήριξή σε υλικοτεχνική υποδομή και τεχνογνωσία.

Τέλος, θα ήθελα να ευχαριστήσω ιδιαίτερα τους γονείς μου και τους φίλους μου, που χάρη σε αυτούς έζησα μια 2η φοιτητική ζωή.
# Contents

1) Introduction
1.1) Motivation .................................................. 19
1.2) Goals and approach ........................................ 19
1.3) Thesis Outline ............................................. 20

2) Background and related work
2.1) Background .................................................. 22
2.1.1) Business process models ................................. 22
2.1.2) Business Process Model and Notation (BPMN) .......... 23
2.1.3) Why Semantic annotated BPMN models? ............. 26
2.1.4) Why Web Ontology Language (OWL)? ............... 26
2.1.5) Reasoning & semantic queries .......................... 27
2.2) Business process modelling and semantic annotation tools .... 29
2.2.1) Maestro ................................................... 29
2.2.2) WSMO Studio ............................................ 29
2.2.3) Pro-SEAT .................................................. 30
2.2.4) SeMFIS ....................................................... 30

3) Consistency Check on Domain Ontology Concepts
3.1) SeMFIS Tool ................................................. 31
3.1.1) SeMFIS Features ......................................... 31
3.1.2) Model Editors ............................................. 31
3.1.3) Scripting and Analysis functionalities .................. 34
3.2) Extending SeMFIS modeling constructs for supporting new useful concepts of OWL 2.0 ................................................. 34
3.2.1) Restrictions on Data Properties ....................... 35
3.2.2) Sequence restrictions ................................... 36
3.3) Consistency Check feature ................................ 37
3.3.1) Functionality ............................................. 37
3.3.2) Reasoner .................................................. 41
3.3.3) Explaining Inconsistencies in OWL Ontologies .... 41

4) Queries
4.1) AQL and limitations ........................................ 43
4.2) Need for SPARQL ........................................... 45
4.3) Semantically enriched queries .............................. 46
4.3.1) Virtuoso .................................................... 46
4.3.2) Upload to Virtuoso ....................................... 46
4.3.3) Query Ontology Models ................................. 49

5) Use Case
5.1) Traceability ..................................................... 55
5.2) Meat Value Chain ........................................... 56
5.2.1) Entity-Relationship Model .............................. 57
5.2.2) The Open Provenance Model Core Specification (v1.1) .... 58
5.2.2.1) Introduction ........................................... 58
5.2.2.2) Basics .................................................. 58
5.2.2.3) Dependencies ......................................... 59
5.2.2.4) Correlation of OPM with Meat Value Chain..........................60
5.2.3) BPMN Process model..............................................................61
5.3) Meat value chain ontology model..............................................72
5.4) Use case examples.......................................................................82
  5.4.1) Consistency check feature examples........................................82
  5.4.2) Semantically enriched queries feature examples.........................82
6) Conclusion
  6.1) Summary...................................................................................84
  6.2) Limitations and Future work......................................................84

References
List of Figures

2.1 Example of business process modeling of a process with a normal flow with the Business Process Modeling Notation ................................................................. 22
3.1 Business process model (BPMN 2.0) ............................................................... 32
3.2 Ontology model ......................................................................................... 33
3.3 Semantic Annotation model ....................................................................... 34
3.4 User can define the value of a datatype property ....................................... 35
3.5 User can set datatype property restrictions ............................................... 35
3.6 User can set datatype property restrictions ............................................... 36
3.7 Reasoning feature on Menu ..................................................................... 37
3.8 Prompts user to choose his ontology models .......................................... 38
3.9 The user chose the ontology model “myMeatOntology_1” ......................... 38
3.10 Message in case of inconsistency ............................................................. 39
3.11 The ontology is consistent ...................................................................... 39
3.12 User should choose the preferred BPMN and Semantic Annotation models 40
3.13 Annotated BPMN Activity .......................................................... 38
3.14 The feature could not find any annotated activities .................................. 41
4.1 Single Entity Patterns ............................................................................. 44
4.2 Composite Patterns ............................................................................... 44
4.3 Annotations Patterns ............................................................................ 45
4.4 Semantically enriched queries ................................................................. 47
4.5 User selects ontology models for uploading on Virtuoso ......................... 47
4.6 Choose the ontology models for export .................................................. 48
4.7 Successful uploading ............................................................................. 48
4.8 Query Ontology Models ......................................................................... 49
4.9 List of predefined queries ....................................................................... 49
4.10 Enter the name of your class .................................................................. 50
4.11 Results of predefined query 1 ................................................................. 50
4.12 Enter the name of your instance ............................................................. 50
4.13 Results of predefined query 2 ................................................................. 51
4.14 Enter the name of your instance ............................................................. 51
4.15 Enter the name of your property ............................................................ 51
4.16 Results of predefined query 3 ................................................................. 52
4.17 Results of predefined query 4 ................................................................. 52
4.18 Results of predefined query 5 ................................................................. 52
4.19 Results of predefined query 6 ................................................................. 53
4.20 Enter PREFIX ......................................................................................... 53
4.21 Enter your query ................................................................................... 53
4.22 Results of query ................................................................................... 54
5.1 Entity Relationship model .......................................................... 57
5.2 Edges in the Open Provenance Model: sources are effects, and destinations causes 59
5.3 The general process model ................................................................. 61
5.4 Zooming in general process model ....................................................... 61
5.5 Zooming in general process model ....................................................... 62
5.6 Transportation of livestock process model ............................................ 63
5.7 Zooming in transportation of livestock process model ......................... 63
List of Tables

5.1 Relations correlation.................................................................60
5.2 Entities Correlation.................................................................60
Chapter 1

Introduction

1.1) Motivation

Business process models are the analytical representation or illustration of an organization’s business processes. They are widely used from business industry and several techniques have been developed to support them, because they are viewed as a critical component in successful business process management.

Nevertheless, available modelling notations for business processes, such as BPMN (Business Process Modelling Notation), lack the ability to specify semantic properties of the processes, including those related to their business domain. However, semantic information is important for tasks that involve reasoning over the process and for which automated support is desirable. For example, documenting or querying a process, enforcing a policy, or verifying constraints on the business logics involve semantic reasoning that cannot be carried out on process models expressed in BPMN or similar languages.

For this purpose there is an ongoing research and several software tools have been developed that permit to enrich business process models with semantic annotations. The field is open for new approaches and further development of existing ones.

1.2) Goals and approach

My goal is to contribute on improving existing semantic annotation tools for offering better user experience after the addition of new useful features. After searching for related tools, I found out that there is not an open source tool with integrated owl ontology reasoning and querying functionalities. In that direction, I decided to add an ontology consistency check feature and a feature for performing semantically enriched queries over owl ontology schemas and their instances, that are annotated to business process models expressed in BPMN notation. I chose to implement these features on SeMFIS tool [15] because it allows semantic enrichment of several modelling languages and it is an open and extendable platform. Though, I am going to focus on BPMN language for modelling business processes.
SeMFIS allows the user to annotate some or every activity in BPMN models with ontology classes, in that way it can represent possible additional information that BPMN specification does not support into OWL representational language. In that way it achieves better understanding, documenting, querying and reasoning about properties, constraints and design choices of BPMN models that cannot be expressed in a purely syntactic way.

Assuming that there are activities of process models that require the record of more information than BPMN could express, record them in owl language allows the user to benefit from the above mentioned properties. Meanwhile a non expert user can conduct a consistency check on business process models instances expressed in OWL and see directly on BPMN process model editor in which activity perhaps an inconsistency occurs.

Also, we should mention the advantage of easily adaptability. Business process models, owl ontology schemas and the connection between them are all in different model editors. With this distinction a user can benefit from the fact that without changing his process models, he can just adapt his annotated ontology to different documentation needs e.g different legal or business requirements. For example, state could define new laws for food traceability recording requirements or businesses could add their own for better auditing. Somebody should have business process models and their ontology annotations in separate models, so he can use them as distinct models, import them in different tools for exploiting their functionality or process them in different ways.

Then with a use case of a meat supply chain I will show the added value in the tool and how the new functionality benefits expert and non-expert users.

1.3) Thesis Outline

My thesis is organized in six chapters and here follows a brief outline of each chapter. In Chapter 2, I am going to present the basic terms that I am using and their definitions such as Business Process Modelling, BPMN, Semantic Annotations and OWL ontology language and also I will give a short description of some business process modelling and annotation tools and what is their contribution on the domain. In Chapter 3, I will describe with more details the features and functionalities of SeMFIS tool, I will refer to my additions so as the tool can support the most basic concepts of OWL 2.0 and subsequently the functionality of a Consistency check feature that I added to it. In Chapter 4, I will present AQL, SeMFIS query language, I will talk about its limitations and thus the need for SPARQL language and I will describe my semantically enriched query feature. In Chapter 5, I will present my use case which is based on the importance of food traceability. I am going to show a process model of meat supply chain and then I will describe the need for documenting more information that a BPMN process model
can not support. Then, after I record these extra information, necessary for traceability, in OWL language, I will show the added value of my features to the tool and how a user can benefit. Chapter 6, draws some conclusions on my work, limitations and some proposals for improvement.
Chapter 2

Background and related work

2.1) Background

As i already mentioned in thesis outline, in this chapter i will describe with more details the terms that i studied and worked on, additionally i will present the related work.

2.1.1) Business process models

**Business process modeling (BPM)** [21] in systems engineering is the activity of representing processes of an enterprise, so that the current process may be analyzed or improved. The business objective is often to increase process speed or reduce cycle time; to increase quality; or to reduce costs, such as labor, materials, scrap, or capital costs. In practice, a management decision to invest in business process modeling is often motivated by the need to document requirements for an information technology project.

Change management programs are typically involved to put any improved business processes into practice. With advances in software design, the vision of BPM models becoming fully executable (and capable of simulations and round-trip engineering) is coming closer to reality.

![Figure 2.1: Example of business process modeling of a process with a normal flow with the Business Process Modeling Notation](image)
A business model is a framework for creating economic, social, and/or other forms of value. The term 'business model' is thus used for a broad range of informal and formal descriptions to represent core aspects of a business, including purpose, offerings, strategies, infrastructure, organizational structures, trading practices, and operational processes and policies.

A business process is a collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer or customers. There are three main types of business processes:

- **Management processes**, that govern the operation of a system. Typical management processes include corporate governance and strategic management.

- **Operational processes**, that constitute the core business and create the primary value stream. Typical operational processes are purchasing, manufacturing, marketing, and sales.

- **Supporting processes**, that support the core processes. Examples include accounting, recruitment, and technical support.

A business process can be decomposed into several sub-processes, which have their own attributes, but also contribute to achieving the goal of the super-process. The analysis of business processes typically includes the mapping of processes and sub-processes down to activity level. A business process model is a model of one or more business processes, and defines the ways in which operations are carried out to accomplish the intended objectives of an organization. Such a model remains an abstraction and depends on the intended use of the model. It can describe the workflow or the integration between business processes. It can be constructed in multiple levels.

### 2.1.2) Business Process Model and Notation (BPMN)

In Implementing Business Process Modeling, there are many techniques that have been tried and tested throughout the years. Some may have few drawbacks and some proven successful.

Some business process modeling techniques are:

- Business Process Model and Notation (BPMN)
- Cognition enhanced Natural language Information Analysis Method (CogNIAM)
- Extended Business Modeling Language (xBML)
- Event-driven process chain (EPC)
- ICAM DEFinition (IDEF0)
• Unified Modeling Language (UML), extensions for business process such as Eriksson-Penker's
• Formalized Administrative Notation (FAN)

We will further describe **Business Process Model and Notation (BPMN)** that is the modelling technique that is the most widely used.

Business Process Model and Notation (BPMN) [22] is a standard for business process modeling that provides a graphical notation for specifying business processes in a Business Process Diagram (BPD). The objective of BPMN is to support business process management, for both technical users and business users, by providing a notation that is intuitive to business users, yet able to represent complex process semantics. The BPMN specification also provides a mapping between the graphics of the notation and the underlying constructs of execution languages, particularly Business Process Execution Language (BPEL).

It is widely accepted by both commercial and open source BPMS tooling vendors. It is highly adaptable and can be used to capture everything from abstract process outlines to detailed process flows to implementation ready processes. One of the main value propositions of BPMN besides being a diagram standard is the precise semantics behind the diagram. The shape, the symbols, the borders, the placement of the BPMN diagram elements, as well as their properties have well defined meanings and have to be interpreted in the same manner by all tools.

*The [23] five basic categories of elements:*

1. Flow Objects
2. Data Objects
3. Connecting Objects
4. Swimlanes
5. Artifacts

**Flow Objects**

Flow Objects are the main graphical elements to define the behavior of a Business Process

1. Events
2. □ Activities
3. ◊ Gateways

**Data Objects**

1. □ Data Objects
2. □ Data Inputs
3. □ Data Outputs
4. □ Data Stores

**Connecting Objects**

There are four ways of connecting the Flow Objects to each other or other information

1. ▸ Sequence Flows
2. ◄ Message Flows
3. ▶ Associations
4. ▼ Data Associations

**Swimlanes**

1. □ Pools
2. □ Lanes
Artifacts

Artifacts are used to provide additional information about the Process

1. Group

2. Text Annotation

2.1.3) Why Semantic annotated BPMN models?

Business Process management has attracted the attention of companies and organizations, which have heavily invested in the research and development of tools and standards for process representation and business requirement formalization. Several tools have been produced for the representation and management of business processes, and many companies have designed their enterprise models using business process standards. Here we will discuss the approach of semantic annotations, to enrich the existing standards, in particular the Business Process Model Notation (BPMN), with the semantic information provided by OWL ontologies.

Business Process annotation with semantic tags taken from an ontology is becoming a crucial activity for business designers. In fact, semantic annotations help business process comprehension, documentation, analysis and evolution.

Although the semantic scope [14] of a modeling language could be adapted through a redesign and re-implementation of the language, this is often unfavorable. Especially in industry scenarios where sometimes thousands of models have been created over long time periods and with considerable effort, changes in the underlying modeling language may lead to unexpected side effects. Even more so in the case of runtime models, where execution systems interact with the models controlling their behavior. As a solution to this problem, semantic annotations have been introduced for conceptual models. By linking elements of conceptual models to elements in ontologies, additional semantic information can be added without having to modify the underlying modeling language. Thereby, ontologies are chosen because of their nature as an "explicit specification of a conceptualization" that can be shared and that permits to process the annotations by machines.

2.1.4) Why Web Ontology Language (OWL)?

The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies. OWL [14] is designed for use by applications that need to process the content of information instead of just presenting information to humans. It represents rich and complex knowledge about things, groups of things, and relations
between things. Building upon RDF and RDFS, OWL provides more machine-interpretable semantics by defining additional vocabulary along with formal semantics. OWL builds on Description Logics which is a restriction of First Order Logic and provides three increasingly expressive sublanguages: OWL Lite, OWL DL (Description Logics), and OWL Full.

An OWL ontology usually consists of classes, properties, instances of classes, and relationships between these instances. Instances of classes in OWL are called individuals. OWL classes are described through "class descriptions", which can be combined into "class axioms". With class axioms, OWL Lite can represent generalization (rdfs:subClassOf), equality (owl:equivalentClass). Besides, OWL DL can specify classes as logical combinations of other classes (owl:intersectionOf, owl:unionOf, owl:complementOf), or as enumerations of specified objects (owl:oneOf) or as distinction of two classes (owl:disjointWith).

OWL distinguishes between two main categories of properties — object properties (owl:ObjectProperty) to link individuals to individuals and datatype properties (owl:DatatypeProperty) to link individuals to data values. Properties can be specified through domains (rdfs:domain) and ranges (rdfs:range). More property axioms are supported by OWL are sub-property (rdfs:subPropertyOf), equivalent property (owl:equivalentProperty), inverse property (owl:inverseOf), functional property (owl:FunctionalProperty), transitive property (owl:TransitiveProperty), symmetric property (owl:SymmetricProperty) and etc. An arbitrary number (zero or more) of values for a property is represented by cardinality constraints (owl:maxCardinality, owl: minCardinality, and owl: cardinality). Value constraints (owl:allValuesFrom and owl: someValueFrom and owl: hasValue) specify the quantifier restriction of a property. OWL individuals are specified through the class axiom rdfs:subClassOf. The identity of individuals can be stated by referring to the same individual (owl:sameAs), or referring to different individuals (owl:differentFrom), or listing all different individuals (owl: AllDifferent).

2.1.5) Reasoning & Semantic Queries

A formal ontology consists of a set of specifications of representational vocabularies (concepts) for a shared universe of discourse, which may include definitions of classes, relations, functions, and other objects. Hence, ontology contains agreed-upon definitions in the form of human readable text and machine-enforceable, declarative constraints on their well-formed use. Besides, ontologies consist of a set of inference rules from which machines can make logical conclusions.

Reasoning in ontologies and knowledge bases is one of the reasons why a specification needs to be formal one. By reasoning we mean deriving facts that are not expressed in ontology or in knowledge base explicitly. For that purpose we need a reasoner. A reasoner
is a piece of software able to infer logical consequences from a set of asserted facts or axioms.

A few examples of tasks required from a reasoner are as follows [25]:

- **Satisfiability of a concept** - determine whether a description of the concept is not contradictory, i.e., whether an individual can exist that would be instance of the concept.

- **Subsumption of concepts** - determine whether concept C subsumes concept D, i.e., whether description of C is more general than the description of D.

- **Consistency of ABox with respect to TBox** - determine whether individuals in ABox do not violate descriptions and axioms described by TBox.

- **Check an individual** - check whether the individual is an instance of a concept

- **Retrieval of individuals** - find all individuals that are instances of a concept

- **Realization of an individual** - find all concepts which the individual belongs to, especially the most specific ones

Ontologies expressed in OWL language allow along with the use of SPARQL query language, the expression of more complex queries than extracting information from relational databases.

From Wikipedia we get the following definition [26]: Semantic queries allow for queries and analytics of associative and contextual nature. They enable the retrieval of both explicitly and implicitly derived information based on syntactic, semantic and structural information contained in data. Semantic queries work on named graphs, linked-data or triples. This enables the query to process the actual relationships between information and infer the answers from the network of data.

From a technical point of view semantic queries are precise relational-type operations much like a database query. They work on structured data and therefore have the possibility to utilize comprehensive features like operators (e.g. >, < and =), namespaces, pattern matching, subclassing, transitive relations, semantic rules and contextual full text search. The semantic web technology stack of the W3C is offering SPARQL to formulate semantic queries in a syntax similar to SQL. Semantic queries are used in triplestores, graph databases, semantic wikis, natural language and artificial intelligence systems.
2.2) Business process modelling and semantic annotation tools

From [14] we see that regarding approaches that particularly address the semantic annotation of conceptual models, several tools can be found in the area of semantic business process management. The original motivation for semantic business process management was to combine business process management with semantic web concepts. For this purpose software tools have been developed that permit to enrich business process models with semantic annotations.

2.2.1) Maestro

Maestro [29] is a modeling tool from SAP Research. Maestro for BPMN allows to annotate and automatically compose activities within business processes. Business processes are often modeled as a set of activities (or tasks) together with their control flow. In order to make a process executable, e.g., in a workflow execution engine, all tasks in the process have to be carried out manually or automatically by Web services. The tool’s aim is to semantically annotate such tasks and to automatically discover or compose (if needed) the services which collectively implement the required functionality.

From the graphical point of view, Maestro for BPMN follows BPMN. However, it makes use of the sBPMN ontology, by creating on-the-fly a set of instances for sBPMN classes. The sBPMN ontology serves as a meta model for BPMN process models, featuring the concepts, relations and attributes for standard BPMN. This enables supportive reasoning over the working ontology.

2.2.2) WSMO Studio

The Web Services Modelling Framework (WSMF) and the Web Services Modelling Ontology (WSMO) provide a unique, highly innovative perspective onto the Semantic Web and Web Services technologies. WSMF provides the appropriate conceptual model for developing and describing web services and their composition and WSMO is a conceptual model that provides an ontology based framework, which supports the deployment and interoperability of Semantic Web Services. WSMO Studio presents a prototype that supports and elaborates that perspective, making the technology easy to use and transparent for the end user.

According to [30] we see that WSMO Studio functionality consists of:

- Design of a framework for an integrated service environment compatible with the WSMO approach of describing SWS. The architecture will be based on the Eclipse architecture
- Development of a set of components (plug-ins) that add specific functionality to the service environment
- Integration of already existing components into the service environment (for
example existing components for WSDL)
  • Contributions of plug-ins to the Studio by 3rd parties

In addition the below non-functional requirements should be taken into consideration when designing and developing WSMO Studio
  • Role-oriented development
  • Extensibility
  • Open standards
  • Flexible licensing
  • Usability

2.2.3) Pro-SEAT

The Standalone Process Semantic Annotation Tool (Pro-SEAT) [28] has been developed using the semantic annotation framework. The tool needs to import already existing business models, so it must be able to read models created from certain tools. Pro-SEAT works on models generated by the Metis platform, which permits to define arbitrary modeling languages, such as UML, EEML and BPMN. OWL-DL is chosen for modeling the ontology. Therefore, the annotation tool provides an OWL ontology browser and an ontology selection tool to support the ontology-based annotation.

2.2.4) SeMFIS

The aim [27] of the Semantic-based Modeling Framework for Information Systems (SemFIS) is to allow for the semantic enrichment of conceptual modeling languages that represent a particular knowledge area. For this purpose it provides configurable meta models, mechanisms and algorithms that extend the modeling language of the knowledge area and enable the user to apply advanced semantic processing.

Due to the flexible approach of SeMFIS, it can be easily adapted to arbitrary knowledge areas. The current implementation of the SeMFIS framework provides an exchange mechanism with the Protégé toolkit developed by Stanford University to integrate data from OWL ontologies in the modeling framework and make it available for the annotation of model elements.
Chapter 3

Consistency check on domain ontology concepts

My contribution’s goal was to find a tool that allows semantic annotations and add new functionality which will add extra value to that tool and will assist users taking advantage of theirs enriched BPMN models in a more complete way. The first part of my contribution is the addition of an ontology consistency check feature. It allows the user to perform a consistency check on the obtained ontology information related to one or several BPMN models. Consistency checking operations check whether instances of data are consistent with a given schema. The tool I chose to use and extend is SeMFIS.

Below I will describe with more details the properties and capabilities of SeMFIS tool.

3.1) SeMFIS Tool

3.1.1) SeMFIS Features

SeMFIS is an open and extensible platform for engineering semantic annotations of conceptual models. SeMFIS is based on the ADOxx. The ADOxx platform is a metamodeling-based development and configuration environment to create domain-specific modelling tools and it is developed from The Open Models (OMi) Laboratory. SeMFIS can be extended and adapted to fit your personal needs.

The intention [14] behind SeMFIS is to provide a link in the form of a software platform between the field of conceptual modeling and the field of ontologies. Thereby, profits can be gained from both sides: on the one hand, ontologies provide formal information structures and reasoning mechanisms that can be used to enrich conceptual models. On the other hand, the semi-formal style and visual editors traditionally used for conceptual models facilitate the interaction, in particular for non-technical users.

Due to the decoupling of the semantic annotations in separate annotation models, SeMFIS can be directly added to existing modeling methods without affecting their structure nor behavior. It also provides a highly performant persistence layer in the form of a relational database as well as interfaces, scripting and analysis functionalities.
3.1.2) Model Editors

For representing semantic annotations and ontologies, SeMFIS uses a meta model-based approach. SeMFIS meta model comprises four model types: the semantic annotation model type, the frames ontology model type, the OWL ontology model type, and the term model type.

I will use the below three model types that fit my needs, i.e. the annotation of BPMN models with ontologies:

a) **Business process diagram (BPMN 2.0)**

The Business process diagram (BPMN 2.0) contains the most important modelling objects and relations of BPMN 2.0

![Business process model (BPMN 2.0)](image)

Figure 3.1: Business process model (BPMN 2.0)

b) **Ontology model**

The OWL ontology model is used to represent ontologies based on the OWL specification in the form of visual models. The ontology model consists of 7 modeling objects: Package, Namespace, Class, Property, Instance, predicate and AllDifferent.
c) Semantic annotation model

In order to [14] decouple the annotation information from the underlying conceptual models and ontologies; SeMFIS adds a distinct information structure for describing annotations. In this information structure references to elements in conceptual models and to ontologies are established. These references are then linked using annotator elements, which define the type of annotation. In this way, neither the modeling language for conceptual models nor the specification of ontologies has to be changed.

The Semantic Annotation Model consists of three objects for reference, the Model reference, the Ontology reference and the Connector reference. Also it has an Annotator object and three relations (is input for, Refers to, linked to).
3.1.3) Scripting and Analysis functionalities

As SeMFIS [14] has been implemented on top of the ADOxx meta modeling platform, it can revert to a number of functionalities that are provided by the platform’s components. This also includes scripting and analysis functionalities that have been made available for SeMFIS. Via the scripting functionality, statements in the domain-specific ADOscript language can be executed. This language permits to access almost any platform functionality in a programmatic way. It can thus be used for tasks such as constraint checking, model manipulation and analysis, report generation, interaction with third-party tools and APIs, user interaction via a set of pre-defined UI components and many more. In addition to analyses with ADOscript, SeMFIS also integrates the analysis component of the ADOxx platform. By using this component queries expressed in AQL (ADOxx Query Language) can be composed and executed. Although this language is currently not as powerful as SQL - e.g. joins are not yet available - it can be used for easily gathering information from the models.

3.2) Extending SeMFIS modelling constructs for supporting some concepts of OWL 2.0

I added the below extensions so as the Ontology model of SeMFIS could support the needs of my use case ontology “Meat Value Chain”. So, now SeMFIS can support schema constraints, value constraints and cardinality constraints.
3.2.1) Restrictions on Data Properties

A) Define data range restrictions on Datatype properties

I added a record that allows the user to define inequality and data range restrictions on Property class.

![Figure 3.4: User can define the value of a datatype property](image)

Example:

P6_hasDuration
Domain: TransportationOfLivestock
Range: xsd:int[ >5, <10 ] (hours)

B) Define data range restrictions on Datatype properties as subclass of a class

I added a record that allows the user to define inequality and data range property restrictions as subclasses of object Class.

![Figure 3.5: User can set datatype property restrictions](image)
Example:

Class: E45_TransferFrozenMeat

\[
\text{P12\_hasTemperature \ owl: allValuesFrom } \text{xsd: double[<=-18 \ ] } ^\circ \text{C}
\]

This statement restricts the instances of type E45_TransferFrozenMeat when they are related to the property \text{P12\_hasTemperature}, they are allowed to have values of type \text{double} and range <=-18 otherwise a consistency check will find our ontology instances inconsistent.

3.2.2) Sequence restrictions

A) I extended the record “Restrictions” on notebook of object Class so as to be able to support concepts such as [\text{Class1 subclassOf (Property1 Class2 (or Class3) ..)}]

This axiom says that if something is an instance of \text{Class1} and it's related to something by the property \text{Property1}, then that something is an instance of \text{Class2} (or an instance of \text{Class3}) etc.

This axiom is useful because I can restrict a process to be followed by specific process (or processes).

![Figure 3.6: User can set datatype property restrictions](image)

We can see that if an instance of class \text{E3\_Slaughtery} is related to an instance by the property \text{P3\_leadsTo}, that instance could belong only to the classes \text{E4\_TransportationForProcessing, E5\_Processing or E6\_Distribution}. 
3.3) Consistency Check feature

3.3.1) Functionality

The Consistency Check feature checks if an ontology is consistent and then if it is not, checks if a BPMN activity is annotated to the class type of that inconsistent instance and then places an error mark around that activity. So, a non-expert user can conduct reasoning to an ontology that he knows that is annotated to BPMN processes of his interest and view the results in the BPMN model without get involved with ontology details.

As far as SeMFIS exports his models in an XML format we should transform the exported file(s) into RDF/XML syntax for processing them. The Java Architecture for XML binding (JAXB) simplifies access to an XML document from a Java program by presenting the XML document to the program in a Java format. So JAXB helped me to transform the OWL ontology concepts represented in XML to RDF/XML syntax. The transformation and processed results were then processed and displayed with the use of the script language of ADOxx AdoScript.

Describing my feature with more details:

1) We go to the Analysis interface and we see on the Menu bar the word “Reasoning”, when we click on it, it is pointed out to click “Consistency check..”.

![Figure 3.7: Reasoning feature on Menu](image)

2) After clicking on “Consistency Check..” a message is displayed that calls the user to choose the preferred ontology for reasoning.
3) The user chooses the ontology model for reasoning, then he chooses a file for exporting it and at last he clicks “Export. We suppose that the reasoning will be conducted over one ontology schema though ontology instances can be into multiple ontology models.
4) Then in case that the chosen ontology is inconsistent, we get the following message:

![Figure 3.10: Message in case of inconsistency](image)

It prompts us to choose one or more BPMN and Semantic Annotation models that we know (or suppose) that we would find an annotation with our inconsistent ontology. The feature finds just one inconsistent instance for the sake of simplicity. If we want to find out more inconsistencies we have to correct the ones that has been found previously.

Otherwise we get the message that our ontology is consistent:

![Figure 3.11: The ontology is consistent](image)
5) The user chooses the BPMN and Semantic Annotation models.

Figure 3.12: User should choose the preferred BPMN and Semantic Annotation models

6) If there is a semantic annotated BPMN activity with an inconsistent class, the BPMN model that contains it, opens automatically and we can see an error mark around that activity.

Figure 3.13: Annotated BPMN Activity
If we cannot address an annotated activity we get the following message:

![Image](image.png)

Figure 3.14: The feature could not find any annotated activities

### 3.3.2) Reasoner

For [8] conducting reasoning on my ontology I use the HermiT reasoner. HermiT is reasoner for ontologies written using the Web Ontology Language (OWL). Given an OWL file, HermiT can determine whether or not the ontology is consistent, identify subsumption relationships between classes, and much more.

HermiT is the first publicly-available OWL reasoner based on a novel “hypertableau” calculus which provides much more efficient reasoning than any previously-known algorithm. Ontologies which previously required minutes or hours to classify can often be classified in seconds by HermiT, and HermiT is the first reasoner able to classify a number of ontologies which had previously proven too complex for any available system to handle.

HermiT uses direct semantics and passes all OWL 2 conformance tests for direct semantics reasoners. HermiT is open-source, released under GNU Lesser General Public License (LGPL) and uses the OWL API 3.4.3.

### 3.3.3) Explaining Inconsistencies in OWL Ontologies

HermiT reasoner will indicate us if our ontology is consistent or not. If the result is inconsistent we would like to know why. What type of entailments drove to that result? For finding these explanations I used the code of Matthew Horridge “owlexplanation” [11] from GitHub. It is an API and reference implementation for generating justifications for entailments in OWL ontologies.

One of the key aspects of OWL [12] is that it is built upon the foundations of a Description Logic. A consequence of the fact that OWL corresponds to a highly
expressive description logic is that unexpected and undesirable inferences (entailments),
can arise during the construction of an ontology. The reasons as to why an entailment
holds in an ontology can range from simple localized reasons through to highly non-
obvious reasons.
Without automated explanation support it can be very difficult to track down the axioms
in ontology that give rise to entailments. It is for this reason that automated explanation is
an important topic in this area.
In the world of OWL there has been a significant amount of research devoted to the area
of explanation and ontology debugging. In particular, research has focused on a specific
type of explanation called justifications. A justification for an entailment in an ontology is
a minimal subset of the ontology that is sufficient for the entailment to hold. The set of
axioms corresponding to the justification is minimal in the sense that if an axiom is
removed from the set, the remaining axioms no longer support the entailment.
Justifications have turned out to be a very attractive form of explanation: They are
conceptually simple, they have a clear relationship with the ontology from which they are
derived, there are off-the-shelf algorithms for computing them, and there are simple
presentation strategies which work well most of the time.
Yet, it is complex enough to extract from the derived explanations which class’s instance
has an inconsistency, I restricted my code to find out only datatype property
inconsistencies, restrictions on object properties and functional property inconsistencies.
Chapter 4
Queries

SeMFIS tool supports also query functionalities on its ontology and BPMN models. The disadvantage is that it uses AQL (ADOxx Query Language) that it is not that powerful language for querying ontology models. So, there is a need for a more flexible and expressive language.

4.1) AQL and limitations

Extended Backus Naur Form (EBNF) notation is used for describing the AQL syntax. For understanding the expressive capabilities of AQL is necessary to comprehend the modelling constructs of SeMFIS.

The modelling language implementation on SeMFIS consists of (concrete) classes, relation classes, modeltypes or attributes.

A more in depth description of the modelling constructs:

1. **Classes and Relations:** as the base modelling constructs for every modelling toolkit

2. **Attributes on Instance and Class Level:** to provide means to specify instance objects or define classes accordingly.

3. **Special Attributes on Instance and Class Level:** to enable ADOxx specific functionality such as attribute-dependent graphical notation.

4. **Attribute facets:** configuration of attribute appearance and behavior

5. **Modeltypes and model type attributes:** similar as for classes and relations on modeltype level.

The evaluation patterns [13] that SeMFIS is capable to support have been divided into three groups: single entity patterns, composite patterns, and annotation patterns. Thereby the single entity patterns comprise the most basic forms of semantic evaluations that contain only either classes, relation classes, or attributes of a process modeling language. For achieving more complex types of semantic evaluations, the composite patterns combine one or more of the basic entities. Finally, the annotation patterns make use of
semantic annotations of elements of a process modeling language to realize even more comprehensive types of evaluations.

Table 1: Single Entity Patterns

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Contained Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS-EVAL</td>
<td>Class Instances</td>
<td>This pattern focuses only on the class instances of a process modeling language. It can be used to evaluate which classes have been used in a process model.</td>
</tr>
<tr>
<td>REL-EVAL</td>
<td>Relationclass Instances</td>
<td>This pattern focuses only on the instances of relationclasses of a process modeling language. It can be used to evaluate which relationclasses have been used in a process model and which instances of the classes the relationclasses connect.</td>
</tr>
<tr>
<td>ATTR-EVAL</td>
<td>Attribute Values</td>
<td>This pattern regards the attribute values attached to class, relationclass and model instances in process models. Thereby the types of the attributes and the attribute values can be evaluated.</td>
</tr>
</tbody>
</table>

Figure 4.1: Single Entity Patterns

Table 2: Composite Patterns

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Contained Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS-ATTR</td>
<td>Class Instances &amp; Attributes</td>
<td>This pattern focuses on class instances together with their attributes. It can be used to evaluate which attribute values are assigned to specific class instances.</td>
</tr>
<tr>
<td>REL-ATTR</td>
<td>Relationclass Instances &amp; Attributes</td>
<td>This pattern focuses on relationclass instances together with their attributes. It can be used to evaluate which attribute values are assigned to specific instances of relationclasses that connect class instances.</td>
</tr>
<tr>
<td>CLASS-REL</td>
<td>Class Instances &amp; Relations</td>
<td>This pattern focuses on instances of classes and relationclasses. It can be used to evaluate which instances of relationclasses connect which instances of classes.</td>
</tr>
<tr>
<td>CLASS-REL-ATTR</td>
<td>Class Instances, Relationclass Instances, Attributes</td>
<td>This pattern focuses on instances of classes and relationclasses, and attributes. It can be used to evaluate complex combinations of class and relationclass instances together with their attribute values.</td>
</tr>
</tbody>
</table>

Figure 4.2: Composite Patterns
We can see that even the most complex patterns that AQL supports they are not that expressive.

When it comes to query Ontology models the most complex pattern that we can evaluate is “CLASS-ATTR” i.e. we can search for instances of type CLASS that have certain attribute values. Thus, the idea for converting the Ontology models which in SeMFIS could be retrieved only in an XML format, into OWL format, and then be able to evaluate them with a more powerful query language, came up as a need for more flexible and complex queries.

**4.2) Need for SPARQL**

SPARQL [20] is an RDF query language, it is a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. It was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web.

Many [19] successful query languages exist, including standards such as SQL and XQuery. These were primarily designed for queries limited to a single product, format, type of information, or local data store. Traditionally, it has been necessary to formulate the same high-level query differently depending on application or the specific arrangement chosen for the relational database. And when querying multiple data sources it has been necessary to write logic to merge the results. These limitations have imposed higher developer costs and created barriers to incorporating new data sources.
The goal of the Semantic Web is to enable people to share, merge, and reuse data globally. SPARQL is designed for use at the scale of the Web, and thus enables queries over distributed data sources, independent of format. Creating a single query across diverse data stores is easier than having to create multiple queries; it also costs less and provides richer results. SPARQL provides a full set of analytic query operations such as JOIN, SORT and aggregate functions.

Because SPARQL has no tie to a specific database format, it can be used to take advantage of the tidal wave of Web 2.0 data and mash it up with other Semantic Web resources. Furthermore, because disparate data sources may not have the same 'shape' or share the same properties, SPARQL is designed to query non-uniform data.

4.3) Semantically enriched queries

Due to the need for more complex queries on my ontology models I created the feature “Semantically enriched queries”. The functionality of this feature is to allow the user upload his ontology and his desirable ontology instances on Openlink Virtuoso engine and query them.

4.3.1) Virtuoso

Virtuoso Universal Server [16] is a middleware and database engine hybrid that combines the functionality of a traditional RDBMS, ORDBMS, virtual database, RDF, XML, free-text, web application server and file server functionality in a single system. Rather than have dedicated servers for each of the aforementioned functionality realms, Virtuoso is a "universal server"; it enables a single multithreaded server process that implements multiple protocols. The open source edition of Virtuoso Universal Server is also known as OpenLink Virtuoso.

4.3.2) Upload to Virtuoso

Before we query our ontology model(s) created on SeMFIS, we have firstly to upload them on Virtuoso.

Describing my feature “Upload to Virtuoso” with more details:

1) We go to the Analysis interface and we see on the Menu bar the item “Semantically enriched queries”, when we click on it, it is pointed out to click “Upload to Virtuoso” or “Query Ontology Models”.


2) When we click on “Upload to Virtuoso” we get a message:

![Figure 4.5: User selects ontology models for uploading on Virtuoso](image)

3) Then we choose the ontology model(s) that we want to import on Virtuoso for querying. The tool exports the models as an XML file into our file system and then it transforms them into an OWL file and uploads it to Virtuoso. Of course prerequisite for this functionality is to have installed Virtuoso on our system.
4) If the file uploaded successfully we get the following message otherwise we get an error message.
4.3.3) Query Ontology Models

After uploading the preferred ontology model(s) on Virtuoso, we would like to ask useful queries. So when we are clicking on the “Query Ontology Models” we get the following window that includes some predefined queries to choose so as to avoid writing the query from scratch. Also the user can choose the 7th entry that allows him to write a query on his own.

I chose the below predefined queries having in mind what a business designer would need more often while searching useful information annotated on his BPMN process models.
A short description of predefined queries:

1) "Find all instances of class .."
   User is asked to enter the name of the class that he wants to find out instances of it.
   - An editbox is appeared and you can enter the name of your class

   ![Figure 4.10: Enter the name of your class]

   Then you get the results from the SPARQL query:

   ![Figure 4.11: Results of predefined query 1]

2) "Find everything about instance .."
   The user enters the name of an instance and he gets as a result everything related to that instance.
   - An editbox is appeared and you can enter the name of your instance

   ![Figure 4.12: Enter the name of your instance]
Then you get the results from the SPARQL query:

```
> killing_1 http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#E34_Killing
> killing_1 http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/2002/07/owl#NamedIndividual
> killing_1 http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#P3_leadsTo
http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#post-MortemInspection_1
> killing_1 http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#P2_hasBusinessOperator
http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#BO_2
> killing_1 http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#P1_hasInput
http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#meat_4
> killing_1 http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#P2_hasOutput
http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#meat_5
> killing_1 http://www.semanticweb.org/katerina/ontologies/2015/3/farm2fork#P1_hasProcessId
"SAAA1"^^<http://www.w3.org/2001/XMLSchema#string>
```

Figure 4.13: Results of predefined query 2

3) ”Find the value of property .. of instance ..”

The user enters the name of a property and the name of an instance and gets the value of that property.

- An editbox is appeared and you can enter the name of your instance

```
Enter the name of an instance
```

Figure 4.14: Enter the name of your instance

then tools asks the name of your property

```
Enter the name of a property
```

Figure 4.15: Enter the name of your property
Then you get the results from the SPARQL query:

4) “Find transitive relations”
The user enters the name of an instance and the name of a property and find the transitive closure of the property.

e.g inserting instance meat_1 and property P4_wasDerivedFrom we get the following result:

5) “Find all instances of type .. that have property .. and a specific value range”
The user looks for the instances of a specific class that are related to a property with a specific value range.

e.g Let’s find all instances of class E37_Storing that has temperature greater than 10°C
6) **“Use an aggregate function – MIN/MAX/AVG/COUNT”**
At this query, dependent on the aggregate function that the user will choose, the result will be a minimum, maximum or average value. If he selects COUNT the result will be a number of instances that meet certain criteria.

e.g Let’s find the average temperature of instances of class E37_Storing

![Figure 4.19: Results of predefined query 6](image)

7) **“Write your own query”**
The user is allowed to write a query by himself.

- A message is appeared into the edit query box that prompts the user to enter as PREFIX the word farm2fork that is the prefix for every ontology instance.

![Figure 4.20: Enter PREFIX](image)

Entering the following query I will get the below results:

![Figure 4.21 Enter your query](image)
Figure 4.22: Results of query
Chapter 5

Use Case

5.1) Traceability

Traceability [1] is the ability to track any food, feed, food-producing animal or substance that will be used for consumption through all stages of production, processing and distribution. In the event of a food incident it enables the identification and subsequent withdrawal or recall of unsafe food from the market. If the food has not reached the consumer, a trade withdrawal is undertaken. If the food has reached the consumer, a product recall is undertaken which includes notification of the consumer through in-store notices and press releases.

Traceability and product recall are important as they enable food businesses to respond quickly to food safety/quality incidents thereby ensuring that consumer exposure to the affected product is prevented or minimized. A good traceability system ensures that withdrawals/recalls are limited to implicated products, thereby minimizing disruption to trade and company finances.

The recent issues surrounding the undeclared substitution of horse meat in beef products has highlighted the importance of traceability systems in identifying the source of fraudulent activities. This is essential to ensure that these activities, which are undertaken by a minority, do not undermine consumer protection, consumer confidence and the integrity of the majority of the food chain.

Regulation 178/2002 laying down the general principles and requirements of food law, requires every food and feed business in Europe and those bringing food/feed into Europe to have a traceability and recall system in place.

All food and feed businesses must be able to identify where their raw materials (e.g. ingredients and packaging) come from and where their products are going or have gone to, i.e. they must be able to identify one step back and one step forward in the food chain. The latter however, is not applicable to businesses selling directly to the final consumer.

Additionally, there are also legal requirements to keep records, apply traceability information to product and/or documents and provide this information to the competent authority on demand. Further sector-specific legislation applies to certain foods including
fruit and vegetables, sprouted seeds, beef, fish, honey, olive oil, genetically modified organisms and live animals.

There is no legal requirement to implement internal traceability which tracks food ingredients and products as they move through the manufacturing process. However, in many instances the food industry has implemented internal traceability to ensure the integrity of their overall traceability systems.

5.2) Meat Value Chain

In order to ensure a certain level of traceability I created a meat value chain schema. It consists of an E-R diagram and a process model. The idea of the value chain is based on the process view of organizations, the idea of seeing a manufacturing (or service) organization as a system, made up of subsystems each with inputs, transformation processes and outputs.

*The basic entities of the Meat Value Chain are:*

- **Product**
  The entity Product describes animals and meat products

- **Business Operator**
  Business Operators are considered to be all the enterprises, companies and individuals that take part in the processes of product manipulation and they have a specific role or multiple roles.

- **Process**
  Process is a sequence of interdependent and linked procedures which, at every stage, consume one or more products to convert them into output products. These outputs then serve as inputs for the next stage until a known end stage is reached.
5.2.1) Entity-Relationship Model

Here is the Entity-Relationship model:

![Entity Relationship Diagram](image)

Figure 5.1: Entity Relationship Diagram [31]

All entities are connected to each other. The connections are the following:

- **Process hasBusinessOperator Business Operator**
  Every process has one or more business operators with different or the same roles
- **Process hasInput Product**
  Every process has at least one product as input
- **Process hasOutput Product**
  Every process has at least one product as output
- **Process leadsTo Process**
  Every process leads to one or more processes
- **Product wasDerivedFrom Product**
  Every product is derived from one or more products
5.2.2) The Open Provenance Model Core Specification (v1.1)

5.2.2.1) Introduction

The Open Provenance Model OPM [2] is a model of provenance that is designed to meet the following requirements: (1) To allow provenance information to be exchanged between systems, by means of a compatibility layer based on a shared provenance model. (2) To allow developers to build and share tools that operate on such a provenance model. (3) To define provenance in a precise, technology agnostic manner. (4) To support a digital representation of provenance for any "thing", whether produced by computer systems or not. (5) To allow multiple levels of description to coexist. (6) To define a core set of rules that identify the valid inferences that can be made on provenance representation. This document contains the specification of the Open Provenance Model (v1.1) resulting from a community effort to achieve interoperability in the Provenance Challenge series.

5.2.2.2) Basics

The Open Provenance Model allows us to characterize what caused "things" to be, i.e., how "things" depended on others and resulted in specific states. In essence, it consists of a directed graph expressing such dependencies. We introduce here the constituents of such a graph.

Hence, from the perspective of provenance, we introduce the concept of an artifact as an immutable piece of state; likewise, we introduce the concept of a process as actions resulting in new artifacts. A process usually takes place in some context, which enables or facilitates its execution: examples of such contexts are varied and include a place where the process executes, an individual controlling the process, or an institution sponsoring the process. These entities are being referred to as Agents.

The Open Provenance Model is based on these three kinds of nodes, which we now define.

**Definition 1 (Artifact)** Immutable piece of state, which may have a physical embodiment in a physical object, or a digital representation in a computer system.

**Definition 2 (Process)** Action or series of actions performed on or caused by artifacts, and resulting in new artifacts.

**Definition 3 (Agent)** Contextual entity acting as a catalyst of a process, enabling, facilitating, controlling, or affecting its execution.

The Open Provenance Model is a model of artifacts in the past, explaining how they were derived. Likewise, processes also occurred in the past, i.e. they have already completed their execution; in addition, processes can still be currently running (i.e., they may have not completed their execution yet). In no case is OPM intended to describe the state of future artifacts and the activities of future processes.

To facilitate understanding and promote a shared visual representation, we introduce a graphical notation for provenance graphs. Specifically, artifacts are represented by
ellipses; processes are represented graphically by rectangles; finally, agents are represented by octagons.

5.2.2.3) Dependencies

The Open Provenance Model aims to capture the causal dependencies between the artifacts, processes, and agents. Therefore, a provenance graph is defined as a directed graph, whose nodes are artifacts, processes and agents, and whose edges belong to one of the following categories depicted in Figure 1. An edge represents a causal dependency, between its source, denoting the effect, and its destination, denoting the cause.

Figure 5.2: Edges in the Open Provenance Model: sources are effects, and destinations causes

**Artifact Used by a Process**

A "used" edge from process to an artifact is a causal relationship intended to indicate that the process required the availability of the artifact to be able to complete its execution. When several artifacts are connected to a same process by multiple "used" edges, all of them were required for the process to complete.

**Artifacts Generated by Processes**

A "was generated by" edge from an artifact to a process is a causal relationship intended to mean that the process was required to initiate its execution for the artifact to have been Generated”. When several artifacts are connected to a same process by multiple "was generated by" edges, the process had to have begun, for all of them to be generated.
Process Triggered by Process
An edge "was triggered by" from a process P2 to a process P1 is a causal dependency that indicates that the start of process P1 was required for P2 to be able to complete.

Artifact Derived from Artifact
An edge "was derived from" from artifact A2 to artifact A1 is a causal relationship that indicates that artifact A1 needs to have been generated for A2 to be generated. The piece of state associated with A2 is dependent on the presence of A1 or on the piece of state associated with A1.

Process Controlled by Agent
An edge "was controlled by" from a process P to an agent Ag is a causal dependency that indicates that the start and end of process P was controlled by agent Ag.

5.2.2.4) Correlation of OPM with Meat Value Chain

My schema Meat Value Chain is directly correlated with the Open Provenance Model.

The basic entities correlation

<table>
<thead>
<tr>
<th>The Open Provenance Model</th>
<th>Meat Value Chain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact</td>
<td>Product</td>
</tr>
<tr>
<td>Process</td>
<td>Process</td>
</tr>
<tr>
<td>Agent</td>
<td>Business operator</td>
</tr>
</tbody>
</table>

5.1 Entities Correlation

The basic relations correlation

<table>
<thead>
<tr>
<th>The Open Provenance Model</th>
<th>Meat Value Chain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process used Artifact</td>
<td>Process hasInput Product</td>
</tr>
<tr>
<td>Artifact wasGeneratedBy Process</td>
<td>Process hasOutput Product</td>
</tr>
<tr>
<td>Process wasControlledBy Agent</td>
<td>Process hasBusinessOperator Business Operator</td>
</tr>
<tr>
<td>Process wasTriggeredBy Process</td>
<td>Process leadsTo Process</td>
</tr>
<tr>
<td>Artifact wasDerivedBy Artifact</td>
<td>Product wasDerivedFrom Product</td>
</tr>
</tbody>
</table>

5.2 Relations correlation
5.3.2) BPMN Process model

The general process model

Our [3] diagram begins at the farm where the animals are before the slaughter process (we don’t take into account the fact that the animals may have been born or raised in different farms).
At the starting point there is an exclusive-or gate that determines whether our livestock will be distributed in a new facility for slaughter (Transportation of Livestock) or the slaughter will take place at the same facilities (Storage of Livestock).
It follows the Slaughter process that will be further analyzed below. After that optionally follows the process of Transportation for processing indicating that meat may need to be transferred in a different facility for further processing.
The process Processing is optional and indicates that meat after the slaughter process may need to be further processed.
Lastly we have the process of Distribution that depicts the transportation of the final product(s) to the customers.

Figure 5.3: The general process model

Zooming in the model:

Figure 5.4: Zooming in general process model
Figure 5.5: Zooming in general process model

**Transportation of Livestock**

The transportation of animals is the movement of animals by transport destined for slaughter.

*The general model*

First, animals are loaded (Loading) into the means of transport (TransferByTruck, TransferByRail, TransferByAircraft, TransferByVessel) that is going to be used. There are 7 different paths. The chose path is dependent on the picked means of transport and also on the duration of the trip (TransferByTruckLong, TransferByRailLong, TransferByVesselLong).

When the duration of the transportation exceeds an upper limit then there is a different path that describes the route because it is necessary the addition of a resting period process (Resting_Period) if we do not want to violate the regulation[^5]. After the resting period the flow goes back. Then, it follows the unloading process (Unloading) that describes the unloading of animals from the means of transport and the inspection process (Inspection) that is necessary for the safeguard of livestock. After that the flow possibly could go back that means that the livestock could be further transported by the same or different means of transport or follows the laid down period process (LaidDownPeriod) necessary for animals to get rest from the trip.
Figure 5.6: Transportation of livestock process model

Zooming in the model:

Figure 5.7: Zooming in transportation of livestock process model
Figure 5.8: Zooming in transportation of livestock process model

Figure 5.9: Zooming in transportation of livestock process model
Storage of Livestock

It is important having information about this process because it is necessary for animals to be stress and injury free during operations prior to slaughter, so as not to unnecessarily deplete muscle glycogen reserves. Animals should be well rested during the 24-hour period before slaughter. This process will not be further analyzed.

The Slaughter Process

The obligation in the conversion of food animals into edible products and useful by-products is to slaughter the animals in a humane manner and to process the carcasses in a hygienic and efficient way.

The general model

Firstly, there is the cleaning process (Cleaning) necessary for removing dirt before slaughter. It follows the obligatory inspection before slaughter (Ante-mortem Inspection), then the optional process of stunning (Stunning). Stunning is the process of rendering animals immobile or unconscious, without killing the animal, prior to their being slaughtered for food. Stunning is recommended in order to avoid pain and minimize distress or suffering for the animals. Then there is the killing process (Killing), the inspection after slaughter (Post-mortem Inspection), the cooling of the meat (Cooling) and the storage process (Storage).
Zooming in the model:

- 

Figure 5.12: Zooming in the slaughter process

Figure 5.13: Zooming in the slaughter process

### Transportation for processing

Meat should be transported in a correct manner, to make sure no contamination takes place nor bacteria can grow on product.

### The general model

The flow starts with the process of inspection of means of transport (InspectionOfMeansOfTransport), follows optionally the cleaning of that means (CleaningMeansOfTransport) and then there is the loading of meat process (Loading). There are two different paths, the first one is followed when the meat that should be transferred is chilled (TransferChilledMeat) and the second one when the meat is frozen (TransferFrozenMeat). Afterwards follows the process of unloading (Unloading) and the process of inspection (Inspection), then there is an exclusive gateway and the meat products could possibly be further transferred by different means of transport or the same as before.
Figure 5.14: Transportation for processing

Zooming at the model:

- Figure 5.15: Zooming in transportation for processing process

- Figure 5.16: Zooming in transportation for processing process
The Processing Process

When viewing meat products of various size, shape and color in butcher shops or meat sections of supermarkets, there appears to be a great variety of such products with different taste characteristics. In some countries there may be several hundred different meat products, each with its individual product name and taste characteristics.

At a closer look, however, it turns out that many of the different products with different product names have great similarities. Based on the processing technologies used and taking into account the treatment of raw materials and the individual processing steps, it is possible to categorize processed meat products in six broad groups. Of course each group could be further analyzed but I will not proceed in depth.

The general model

Fresh Processed Meat Products

These products are meat mixes composed of comminuted muscle meat, with varying quantities of animal fat. Products are salted only, curing is not practiced. Non-meat ingredients are added in smaller quantities for improvement of flavor and binding, in low-cost versions larger quantities are added for volume extension.

Cured Meat

Cured meat cuts are made of entire pieces of muscle meat and can be sub-divided into two groups, cured-raw meats and cured cooked meats. The curing for both groups, cured-raw and cured-cooked, is in principle similar: The meat pieces are treated with small amounts of nitrite, either as dry salt or as salt solution in water.

Raw Cooked Meat Products

The product components muscle meat, fat and non-meat ingredients which are processed raw, i.e. uncooked by comminuting and mixing. The resulting viscous mix/batter is portioned in sausages or otherwise and thereafter submitted to heat treatment, i.e. “cooked”.

Pre-cooked Cooked Products

Precooked-cooked meat products contain mixes of lower-grade muscle trimmings, fatty tissues, head meat, animal feet, animal skin, blood, liver and other edible slaughter by-products. There are two heat treatment procedures involved in the manufacture of precooked-cooked products. The first heat treatment is the precooking of raw meat materials and the second heat treatment the cooking of the finished product mix at the end of the processing stage.
**Raw (dry) Fermented Sausages**

Raw-fermented sausages are uncooked meat products and consist of more or less coarse mixtures of lean meats and fatty tissues combined with salts, nitrite (curing agent), sugars and spices and other non-meat ingredients filled into casings. They receive their characteristic properties (flavor, firm texture, red curing color) through fermentation processes. The products are not subjected to any heat treatment during processing and are in most cases distributed and consumed raw.

**Dried Meat Products**

Dried meat products are the result of the simple dehydration or drying of lean meat in natural conditions or in an artificially created environment. Dried meat is not comparable to fresh meat in terms of shape and sensory and processing properties, but has significantly longer shelf-life. Many of the nutritional properties of meat, in particular the protein content, remain unchanged through drying.

![Figure 5.17: The Processing Process](image-url)
Distribution

Meat [4] products are susceptible to contamination from a wide variety of physical, chemical, biological and radiological agents. Everyone in the food distribution system is responsible for ensuring that these products are safe, wholesome, and unadulterated. Therefore, as part of this system, those responsible for transportation and delivery should implement food defense measures to ensure the safety and security of the products throughout the supply chain.

The general model

The Distribution process starts with the process (Loading), the meat products are loading in the (first) means of transport. After that they are transferred (TransferChilledProducts, TransferFrozenProducts) to the final customer (Retail Delivery) or they are stored temporarily (Storage) and they are transferred from a different business operator or/and with a different means of transport. After any unloading (Unloading) there is a process of inspection (Inspection).

![Figure 5.18: Distribution process](image-url)
Zooming at the model:

- Figure 5.19: Zooming in Distribution process

- Figure 5.20: Zooming in Distribution process
5.3) Meat Value Chain ontology model

I created the Meant Value Chain ontology in order to express the constraints of the meat process model that I cannot with the BPMN 2.0 notation. My ontology is in OWL 2.0 and in RDF/XML syntax.

At start I created the respective classes of my basic entities of the Entity Relationship model (Process, Business Operator, Product). The classes are E0_Process, E01_BusinessOperator and E02_Product.

The object properties that connect the above classes are the following:

- **P0_hasBusinessOperator**
  - **Domain**: E0_Process
  - **Range**: E01_BusinessOperator

- **P1_hasInput**
  - **Domain**: E0_Process
  - **Range**: E02_Product

- **P2_hasOutput**
  - **Domain**: E0_Process
  - **Range**: E02_Product

- **P3_leadsTo**
  - **Domain**: E0_Process
  - **Range**: E0_Process

- **P4_wasDerivedFrom**
  - **Domain**: E02_Product
  - **Range**: E02_Product

- **E0_Process**

The class E0_Process has 6 subclasses, each one represents one activity from the general process model. These are E1_TransportationOfLivestock, E2_StorageOfLivestock, E3_Slaughtery, E4_TransportationForProcessing, E5_Processing and E6_Distribution.
Restrictions:

SubClass Of:

- $P0\_hasBusinessOperator\ min\ 1\ E01\_BusinessOperator$
  (every process has at least one business operator)
- $P1\_hasInput\ min\ 1\ E02\_Product$
  (every process has at least one product as input)
- $P2\_hasOutput\ min\ 1\ E02\_Product$
  (every process has at least one product as output)

Main Data Properties

- E0\_Process $P1\_hasProcessId$ string
- E0\_Process $P2\_hasProcessName$ string
- E0\_Process $P3\_hasDescription$ string
- E0\_Process $P4\_hasStartTime$ dateTime
- E0\_Process $P5\_hasEndTime$ dateTime
- E0\_Process $P6\_hasDuration$ double

Due to inheritance the subclasses of class E0\_Process inherit its restrictions and properties. So I will not reference them again when I will describe these subclasses below.

Describing the subclasses of process E0\_Process:

- **E1\_TransportationOfLivestock**

Disjoint With:

E2\_StorageOfLivestock, E3\_Slaughtery, E4\_TransportationForProcessing, E5\_Processing and E6\_Distribution.

The subclasses of E1\_TransportationOfLivestock are:

- **E11\_Loading**

Restrictions:

SubClass Of:

- $P3\_leadsTo\ only\ (E110\_TransferByTruckLong\ or\ E111\_TransferByRailLong\ or\ E112\_TransferByVesselLong\ or\ E12\_TransferByTruck\ or\ E13\_TransferByRail\ or\ E14\_TransferByAircraft\ or\ E15\_TransferByVessel)$

(This restriction restricts the object property P3\_leadsTo to have as range, instances that belong only to the above classes when its domain instance belongs to the class E11\_Loading. I will use this type of restriction for each class of my ontology so as to assure that each process is executed in the right order as it is designed in my process model.)
- **E12_TransferByTruck**

*Restrictions:*

*SubClass Of:*

- `P12_hasTemperature only double [>= 5.0, <= 30.0] [5]`
  (The temperature while transportation of animals should be between 5°C to 30°C)
- `P3_leadsTo only E17_Unloading`
- `P6_hasDuration only double [<= 8.0] [5]`
  (The duration of transportation should not exceed 8 hours)

- **E13_TransferByRail**

*Restrictions:*

*SubClass Of:*

- `P12_hasTemperature only double [>= 5.0, <= 30.0] [5]`
  (The temperature while transportation of animals should be between 5°C to 30°C)
- `P3_leadsTo only E17_Unloading`
- `P6_hasDuration only double [<= 14.0] [5]`
  (The duration of transportation should not exceed 14 hours)

- **E14_TransferByAircraft**

*Restrictions:*

*SubClass Of:*

- `P12_hasTemperature only double [>= 5.0, <= 30.0] [5]`
- `P3_leadsTo only E17_Unloading`

- **E15_TransferByVessel**

*Restrictions:*

*SubClass Of:*

- `P12_hasTemperature only double [>= 5.0, <= 30.0] [5]`
- `P3_leadsTo only E17_Unloading`
- **E16_Resting01**

  **Restrictions:**
  
  **SubClass Of:**
  
  - `P3_leadsTo only (E110_TransferByTruckLong or E12_TransferByTruck)`
  - `P6_hasDuration only double[>= "1"^^double] [5]`
    (It is necessary for the livestock to get rest before the continuing of the transportation)

- **E17_Unloading**

  **Restrictions:**
  
  **SubClass Of:**
  
  - `P3_leadsTo only E18_Inspection`

- **E18_Inspection**

  **Restrictions:**
  
  **SubClass Of:**
  
  - `P3_leadsTo only (E11_Loading or E19_LaidDownPeriod)`

- **E19_LaidDownPeriod**

  **Restrictions:**
  
  **SubClass Of:**
  
  - `P3_leadsTo only E31_Cleaning`
  - `P6_hasDuration only double[>= 24.0] [5]`
    (Animals should get rest after the completion of their transportation for a period over 24 hours)

- **E110_TransferByTruckLong**

  **Restrictions:**
  
  **SubClass Of:**
  
  - `P12_hasTemperature only double [>= 5.0, <= 30.0]`
  - `P3_leadsTo only (E16_Resting01 or E17_Unloading)`
  - `P6_hasDuration only double[< 14.0]`
- **E111_TransferByRailLong**

  **Restrictions:**

  **SubClass Of:**

  - `P3_leadsTo only (E113_Resting02 or E17_Unloading)`
  - `P6_hasDuration only int[<= 14] [5]`

- **E112_TransferByVesselLong**

  **Restrictions:**

  **SubClass Of:**

  - `P3_leadsTo only (E114_Resting03 or E17_Unloading)`
  - `P6_hasDuration only double[< 14.0]`

- **E113_Resting02**

  **Restrictions:**

  **SubClass Of:**

  - `P3_leadsTo only (E111_TransferByRailLong or E13_TransferByRail)`
  - `P6_hasDuration only double[>= "1"^^double]`

- **E114_Resting03**

  **Restrictions:**

  **SubClass Of:**

  - `P3_leadsTo only (E112_TransferByVesselLong or E15_TransferByVessel)`
  - `P6_hasDuration only double[>= "1"^^double]`

- **E2_StorageOfLivestock**

  This class does not have any subclasses and just inherits the properties of the class E0_Proces
Restrictions:

SubClass Of:

- P3_leadsTo only E31_Cleaning

- E3_Slaughtery

The subclasses of E3_Slaughtery are:

- E31_Cleaning

Restrictions:

SubClass Of:

- P3_leadsTo only E32_AnteMortemInspection

- E32_AnteMortemInspection

Restrictions:

SubClass Of:

- P3_leadsTo only (E33_Stunning or E34_Killing)

- E33_Stunning

Restrictions:

SubClass Of:

- P3_leadsTo only E34_Killing

- E34_Killing

Restrictions:

SubClass Of:

- P3_leadsTo only E35_PostMortemInspection
• **E35\_PostMortemInspection**

**Restrictions:**

**SubClass Of:**

- *P3\_leadsTo only E36\_Cooling*

• **E36\_Cooling**

**Restrictions:**

**SubClass Of:**

- *P3\_leadsTo only E37\_Storing*

• **E37\_Storing**

**Restrictions:**

**SubClass Of:**

- *P12\_hasTemperature only int[<= 4] [6]*
- *P3\_leadsTo only (E41\_InspectionOfMeansOfTransport or E51\_FreshProcessedMeatProducts or E52\_CuredMeat or E53\_RawCookedProducts or E54\_Precooked-CookedProducts or E55\_RawFermentedSausages or E56\_DriedMeatProducts or E61\_Loading) [7]*

- **E4\_TransportationForProcessing**

The subclasses of **E4\_TransportationForProcessing** are:

• **E41\_InspectionOfMeansOfTransport**

**Restrictions:**

**SubClass Of:**

- *P3\_leadsTo only (E42\_Cleaning or E43\_Loading)*
- **E42_Cleaning**

  **Restrictions:**

  **SubClass Of:**

  - `P13_hasWashWaterTemperature only double[>= 82.0] [7]`
  - `P3_leadsTo only E43>Loading`

- **E43>Loading**

  **Restrictions:**

  **SubClass Of:**

  - `P3_leadsTo only (E44_TransferChilledMeat or E45_TransferFrozenMeat)`

- **E44_TransferChilledMeat**

  **Restrictions:**

  **SubClass Of:**

  - `P12_hasTemperature only double[< 4.0] [6]`
  - `P3_leadsTo only E46_Unloading`

- **E45_TransferFrozenMeat**

  **Restrictions:**

  **SubClass Of:**

  - `P12_hasTemperature only double[< -18.0] [6]`
  - `P3_leadsTo only E46_Unloading`

- **E46_Unloading**

  **Restrictions:**

  **SubClass Of:**

  - `P3_leadsTo only E47_InspectionOfMeat`
- E47_InspectionOfMeat

**Restrictions:**

**SubClass Of:**

- P3_leadsTo only \((E41\_InspectionOfMeansOfTransport \text{ or } E51\_FreshProcessedMeatProducts \text{ or } E52\_CuredMeat \text{ or } E53\_RawCookedProducts \text{ or } E54\_Precooked-CookedProducts \text{ or } E55\_RawFermentedSausages \text{ or } E56\_DriedMeatProducts)\)

- E5_Processing

The subclasses of **E5_Processing** are:

- E51_FreshProcessedMeatProducts
- E52_CuredMeat
- E53_RawCookedProducts
- E54_Precooked-CookedProducts
- E55_RawFermentedSausages
- E56_DriedMeatProducts

The above classes have an abstract enough description so they do not have any more particular restrictions, except from the fact that any of them if followed by the class **E61.Loading**.

**Restrictions:**

**SubClass Of:**

- P3_leadsTo only E61.Loading

- E6_Distribution

The subclasses of **E6_Distribution** are:

- E61.Loading

**Restrictions:**

**SubClass Of:**

- P3_leadsTo only \((E62\_TransferChilledProducts \text{ or } E63\_TransferFrozenProducts)\)
- **E62_TransferChilledProducts**

  **Restrictions:**

  **SubClass Of:**

  - $P12_{\text{hasTemperature}}$ only double[$< 4.0$] [6]
  - $P3_{\text{leadsTo}}$ only $E64_{\text{Unloading}}$

- **E63_TransferFrozenProducts**

  **Restrictions:**

  **SubClass Of:**

  - $P12_{\text{hasTemperature}}$ only double[$< -18.0$] [6]
  - $P3_{\text{leadsTo}}$ only $E64_{\text{Unloading}}$

- **E64_Unloading**

  **Restrictions:**

  **SubClass Of:**

  - $P3_{\text{leadsTo}}$ only $E65_{\text{Inspection}}$

- **E65_Inspection**

  **Restrictions:**

  **SubClass Of:**

  - $P3_{\text{leadsTo}}$ only ($E61_{\text{Loading}}$ or $E66_{\text{Storage}}$ or $E67_{\text{RetailDelivery}}$)

- **E66_Storage**

  **Restrictions:**

  **SubClass Of:**

  - $P3_{\text{leadsTo}}$ only $E61_{\text{Loading}}$

- **E67_RetailDelivery**
5.4) Use case examples

5.4.1) Consistency check feature use case examples

We have instances of meat value supply chain schema, given that our schema has some restrictions expressed in OWL language, after conducting a consistency check, if they are violated, we inform the user via an error mark around the annotated inconsistent activity. Possible violations could be missing necessarily traceable activities from the supply chain, data properties with values out of proper data range, or violation of stated cardinalities.

Let’s examine the case that we have an EU organization (like EFSA) and wants to make some inspections on meat products at a supermarket about compliance with meat safety standards. We assume that there is a bar code with which we can trace back all the processes that a particular meat product has been through. Provided this information we can check if they are compliant with the schema. The incompliance could be important because it could regard consumer's health risks or inhuman slaughter of animals (e.g. not stunning animals before slaughter) either much more legal violations.

Being more descriptive about meat product chain requirements:

a) To avoid health risks and comply with regulations the animals and in later stage the meat products have to go through some processes, some of them are obligatory and others optional. A consistency check over the instances of a schema, can address this type of chain deviation inconsistencies e.g. assuming that the animals are in the stage before slaughter, the process of Ante-mortem inspection is obligatory. If we trace a meat product and find out that the animal(s) from which the product comes from, haven't pass through an ante-mortem inspection, it could be a possible threat and should be addressed.

b) Also we restrict some processes to have properties with certain value range. It can be also addressed by the Consistency check feature e.g. Some processes demand certain degrees of temperature such as Storage and Cooling to maintain the good quality of meat. Also, some other demand certain duration otherwise it is illegal, such as the transportation of animals. So setting these constraints is a necessary practice. OWL allows us to set these types of constraints via data properties and value restrictions. So a reasoner can find out possible inconsistencies.

5.4.2) Semantically enriched queries feature use case examples

Queries are of ultimate importance for getting insights on our data and learn useful things. Transforming data into OWL syntax allows us to take advantage of SPARQL's expressive power as i have mentioned before.
Describing some use cases:

a) There is a poisoning incident and it is identified that it comes from spoiled meat products. At the behest of authorities, every point of sale is commanded to withdraw all meat products of that brand and also hand in all info related to their provenance.

b) After conducting a consistency check and discover an inconsistency in a certain process, a Food Safety Authority wants to find out and withdraw all products that come from this process and also discover the business operator(s) that are responsible for it.

c) There are also use cases that somebody (person/company/authority) wants to conduct research and create knowledge from statistics. So, aggregated queries can be proven very useful for that purpose e.g. find out the average duration of some processes or the most common processing practices etc.

For all that reasons, queries can be proven a powerful tool and help BPMN designers and simple users to extract the needed information from their ontology models in a simple manner yet very accurate. From the above examples we can see that somebody can record and retrieve useful complementary information about their processes and without SPARQL that kind of queries could not be possible.
Chapter 6

Conclusion

6.1) Summary

Summing up my master thesis, I referred to the importance of BPMN process model standard and the purpose that it serves. After that we talked about the need for enriching it with semantic information provided by OWL ontologies, and the multiple gains that it will yield. SeMFIS semantic annotation tool is an open and extensible platform that helped us to create two new features that help BPMN users to extract useful information about their process models. The first is the “Consistency check” feature that allows the users to check if the instances of their ontology schema that is annotated to one or several BPMN models are consistent with the given schema or not. If there is an inconsistency the feature can address the class of the inconsistent instance and if there is an annotation to any BPMN model, automatically opens it and puts an error mark on the inconsistent activity. The second feature is the “Semantically Enriched Queries” feature that supports uploading of ontology models on Virtuoso engine and querying them. There is a number of predefined queries so users can save time than writing them on their own and an option for writing them by themselves. After creating a meat supply chain process model and an ontology model, we gave examples of real scenarios that describe how users can benefit from these features.

6.2) Limitations and Future work

SeMFIS constantly releases new versions that support more process and ontology models. Till now it supports only OWL ontology language. I added three tables that support concepts of OWL 2.0, one for data property range values and another two for object and datatype property restrictions. Still, there is room for improvement and support more concepts of OWL 2.0 so as users can take advantage of them. Also a crucial addition should be the development of an OWL import and export functionality. Now, SeMFIS support only XML and ADL import and export functionality. There is just a library for Protege tool that allows ontology extraction into XML compatible with SeMFIS format. Of course, due to the fact that SeMFIS does not support OWL 2.0, this library can just export ontology schemas expressed in OWL and not OWL 2.0. This lack prevents a lot of significant use cases. Furthermore, a useful addition would be the opposite trace direction e.g a user while studying a BPMN process model, could have the
ability to trace automatically the existing ontology instances and finds out any correlations so as to have a more complete picture of the properties of his model. Lastly, it would be very interesting if there is the capability of combining queries that query at the same time the process and ontology models and discover useful patterns.
References

[1] Retrieved April 18, 2016, from  


[3] Retrieved April 18, 2016, from  


