



GoodRelations: An Ontology for Describing Web Offerings

Martin Hepp

Technical Report 2008-05-15

08 August 2008



Important: The base URI of the GoodRelations ontology has changed to <http://purl.org/goodrelations/v1#>, and the Web page has moved to <http://purl.org/goodrelations>. Also, some features have been added.

Please make sure that you read the Errata and Change Notes available at <http://www.ebusiness-unibw.org/twiki/bin/view/EBusiness/GoodRelationsTechnicalReportErrata> prior to using GoodRelations.

This Technical Report has not yet been updated accordingly.

As a normative reference, use

- the Ontology Reference at <http://purl.org/goodrelations/v1> and
- the Primer and User's Guide at <http://www.heppnetz.de/projects/goodrelations/primer/>

The GoodRelations Ontology

<Martin Hepp>^{1,2}

Abstract: A promising application domain for Semantic Web technology is the annotation of products and services offerings on the Web so that consumers and enterprises can search for suitable suppliers using products and services ontologies. While there has been substantial progress in developing ontologies for types of products and services, namely eClassOWL, this alone does not provide the representational means required for e-commerce on the Semantic Web. Particularly missing is an ontology that allows describing the *relationships* between (1) Web resources, (2) offerings made by means of those Web resources, (3) legal entities, (4) prices, (5) terms and conditions, and (6) the aforementioned ontologies for products and services. For example, we must be able to express that a particular Web site describes an offer to sell cellphones of a certain make and model at a certain price, that a pianohouse offers maintenance for pianos that weigh less than 150 kg, or that a car rental company leases out cars of a certain make and model from a particular set of branches across the country.

In this paper, we analyse the complexity of product description on the Semantic Web and define the *GoodRelations* ontology that covers the representational needs of typical business scenarios for commodity products and commodity services.

Keywords: Semantic Web, E-Commerce, E-Procurement, eclassOWL, UNSPSC, eCI@ss

Acknowledgements: The author would like to thank Axel Polleres, Jos de Bruijn, Doug Foxvog, Katharina Siorpaes, Jacek Kopecky, Markus Linder, and Martin Schliefnig for numerous discussions and their invaluable feedback throughout three years of work in progress.

The work presented in this paper has been supported by the Austrian BMVIT/FFG under the FIT-IT Semantic Systems project myOntology (grant no. 812515/9284), by a Young Researcher's Grant (Nachwuchsförderung 2005-2006) from the Leopold-Franzens-Universität Innsbruck, and by the European Commission under the project SUPER (FP6-026850).

¹ Chair of General Management and E-Business
E-Business and Web Science Research Group
Bundeswehr University Munich
Werner-Heisenberg-Weg 39, D-85579 Neubiberg, Germany
Phone: +49 89 6004-4217
eMail: mhepp@computer.org (preferred mode of communication)
<http://www.heppnetz.de> (personal page)
<http://www.unibw.de/ebusiness/> (research group)

² Semantics in Business Information Systems Group, University of Innsbruck
Technikerstrasse 21a, A-6020 Innsbruck, Austria
Phone: +43 512 507 6465
eMail: mhepp@computer.org
<http://sebis.deri.at> (research group)

Table of Contents

1	INTRODUCTION	5
1.1	Current Situation	6
1.2	Contribution	6
2	REPRESENTATIONAL REQUIREMENTS	7
2.1	Motivating Scenarios.....	7
2.2	Requirements	8
2.3	Competency Questions	11
3	THE GOODRELATIONS ONTOLOGY	13
3.1	Domain Capture	13
3.1.1	<i>Web Resource</i>	14
3.1.2	<i>Business Entity</i>	15
3.1.3	<i>Offering</i>	15
3.1.4	<i>Business Function</i>	15
3.1.5	<i>Products or Services: Instances, Models, and Classes</i>	16
3.1.6	<i>Product or Service Properties</i>	18
3.1.7	<i>Commercial Properties of the Offering</i>	21
3.1.8	<i>Relationships Between Multiple Product Models or Product Instances</i>	25
3.2	Standards and Specifications To Be Reused	26
3.2.1	<i>Units of Measurement</i>	26
3.2.2	<i>Countries</i>	26
3.2.3	<i>Currencies</i>	26
3.2.4	<i>Business Functions</i>	26
3.2.5	<i>Product or Service Categories</i>	26
3.2.6	<i>Contact Details and Addresses</i>	27
3.3	Problems and Challenges	27
3.3.1	<i>URI Disambiguation</i>	27
3.3.2	<i>Multiple Offerings in the Same Retrievable Resource</i>	29
3.3.3	<i>Anonymous Instances and Existential Quantification</i>	29
3.3.4	<i>Ranges for Product and Service Attributes</i>	31
3.3.5	<i>Incomplete Descriptions</i>	33
3.4	Formalization: Ontology Coding in OWL DLP.....	34
3.4.1	<i>Appropriate Ontology Language: RDF-S, OWL DLP, or OWL DL?</i>	34
3.4.2	<i>N-Ary Relations in OWL</i>	36
3.4.3	<i>Product Classes, Instances, and Models</i>	38
3.4.4	<i>Top-level Ontology Part for Products and Services Ontologies</i>	40
3.4.5	<i>Reuse of External Standards</i>	41
4	PUTTING IT TO WORK: IMPLEMENTATION ISSUES	42
4.1	Necessary Modifications to eClassOWL	42
4.2	Storage of GoodRelations Annotation Data	44
4.3	Publication of the Vocabulary	45
5	USE CASE AND EVALUATION.....	45
5.1	Features.....	45
5.2	Examples of Annotations	46
5.2.1	<i>Step 1: Business Entity</i>	47
5.2.2	<i>Step 2: Products and Offerings</i>	48
5.2.3	<i>Step 3: Eligible Customers and Regions</i>	53
5.2.4	<i>Step 4: Price Specification</i>	54

5.2.5	<i>Step 5: Delivery and Delivery Charge Specification</i>	54
5.2.6	<i>Step 6: Payment Options and Payment Charge Specification</i>	55
5.2.7	<i>Step 7: Warranty Promise</i>	55
5.2.8	<i>Step 8: Bundles</i>	55
5.2.9	<i>Step 9: Services and Ranges</i>	55
5.2.10	<i>Step 10: Consumables, Accessories, Spare Parts, and Similar Products</i>	55
5.2.11	<i>Step 11: Shop Locations and Opening Hours</i>	56
5.3	Competency Questions in SPARQL	56
5.3.1	<i>Find all known Business Entities and their legal names</i>	56
5.3.2	<i>Who sells cellphones and on which Web pages can I get more information on respective offerings?</i>	56
5.3.3	<i>Which offers of cellphones exists, what is the price, and where can I find the offering on the Web?</i>	57
5.3.4	<i>Who repairs at least one type of cellphone?</i>	57
5.4	Results	58
6	DISCUSSION	58
6.1	Need for a Fully-fledged Model	58
6.2	Related Work	59
6.3	Related Ontologies	59
6.4	Trustworthiness of Annotations	59
6.5	Do Corporations Have an Incentive for Adopting GoodRelations?	60
6.6	Future Directions and Extensions	60
6.6.1	<i>Microformat Variant and GRDDL Transformation</i>	60
6.6.2	<i>Proton Grounding</i>	60
6.6.3	<i>W3C Member Submission</i>	60
6.6.4	<i>More Advanced Price Modeling</i>	60
6.6.5	<i>Resource Composition and Substitution</i>	61
6.6.6	<i>Multi-dependent Properties</i>	61
6.6.7	<i>Integration with Catalog Data Standards for Harvesting Product Data</i>	61
6.6.8	<i>WSML Variant</i>	61
6.6.9	<i>URNs for UPC/EAN</i>	61
6.6.10	<i>Axiomatization</i>	61
7	CONCLUSION	62
8	ACKNOWLEDGEMENTS	62
9	REFERENCES	62
10	APPENDIX A: DOMAIN CAPTURE	66
11	APPENDIX B: GOODRELATIONS DOCUMENTATION	67
11.1	Classes	67
11.2	Object Properties	75
11.3	Datatype Properties	77
12	APPENDIX C: GOODRELATIONS ONTOLOGY SPECIFICATION IN OWL DLP	81
13	APPENDIX D: INSTANCE DATA FOR THE USE CASES IN OWL DLP	97
14	APPENDIX E: PRICING MODEL BY KELKAR, LEUKEL, AND SCHMITZ (2002)	105

1 INTRODUCTION

A promising application domain for Semantic Web technology is the annotation of products and services offerings on the Web so that consumers and enterprises can search for suitable suppliers using products and services ontologies. In fact, there is a vast body of work in the Semantic Web and ontology communities that stresses the potential of ontologies to help managing product data in e-commerce scenarios. Related work can be roughly grouped into three waves:

The first wave: Gupta and Quasem (Gupta & Qasem, 2002) stressed that Semantic Web technology could reduce price dispersion in markets and thus make the market mechanism and e-commerce more efficient. Similarly, Fensel (Fensel et al., 2001) described how ontologies can in principle support the integration of heterogeneous and distributed information in e-commerce, mainly based on catalogs of products, and which tasks are to be mastered. Obrst, Wray and Liu (Obrst, Wray, & Liu, 2001) discuss the main challenges of building and aligning ontologies for products and services in B2B e-commerce environments. Corcho and Gómez-Pérez (Corcho & Gómez-Pérez, 2001) show how multiple standards for classifying products and services can be integrated using ontological mappings, and sketch a prototype implementation based on the WebODE platform. This first wave brought about a few very early transcripts of products and services classifications (UNSPSC and eCI@ss) into ontology languages (Klein, 2002; McGuinness, 2001) and (Christian Bizer & Wolk, 2003). The shortcomings of those early transcripts are described in (Hepp, 2006b).

The second wave: After 2003, several papers were published that analyze the technical details of implementing the overall vision. Most of those works take into account some practical problems of product data management in real-world scenarios that the early papers abstracted from; in particular regarding the amounts of data, the pace of evolution, or the need to reuse existing standards. Tolksdorf et al. (Tolksdorf, Bizer, Eckstein, & Heese, 2003) analyzed the potential impact of Semantic Web technology on B2C e-commerce and discussed the technical and organizational challenges of the vision. They were, to our knowledge, the first to stress the lack of serious business ontologies, consensual identification schemes; open security, trust, and privacy issues, and requirements for the successful diffusion of the Semantic Web in the business world. Zhao and Sandahl (Zhao & Sandahl, 2003) described the potential technical contributions and the required tools and structures for various processes of electronic procurement. Also, Zhao (Zhao, 2003) and Zhao and Lövdahl (Zhao & Lövdahl, 2003) were the first to discuss the need for and difficulties of reusing existing standards when developing e-commerce ontologies. Di Noia et al. (Di Noia, Di Sciascio, Donini, & Mongiello, 2003) analyzed the complexity of matchmaking based on formal descriptions of offerings in electronic marketplaces. Beneventano et al. (Beneventano, Guerra, Magnani, & Vincini, 2004) analyzed the mapping problems between eCI@ss and UNSPSC in detail and produced a prototype for managing respective alignment relations.

The third wave: Only recently, mature, practically usable ontologies for products and services have been derived from common classification standards, namely our ontology eClassOWL (Hepp, 2006a), as described in detail in (Hepp, 2006b). The potential contribution of Semantic Web technology on future shopbots has been renewed in (Fasli, 2006). Most importantly, the enormous body of work done in the Korea (in the context of a large-scale research project) materialized in the form of operational prototypes for ontology-based management of product data and recommender systems, as described in (H. Lee & Shim, 2007; T. Lee, Chun, Shim, & Lee, 2006; T. Lee et al., 2006).

1.1 Current Situation

The semantics of representing products and services themselves is now pretty well understood, and there exist both prototypes from controlled B2B settings (H. Lee & Shim, 2007; T. Lee, Chun, Shim, & Lee, 2006; T. Lee et al., 2006) and general products and services ontologies in the official ontology languages for the Semantic Web, namely eClassOWL (Hepp, 2006a) and a similar ontology unspscOWL, which is awaiting copyright clearance.

However, all these components alone do not yet provide the means required for e-commerce on the Semantic Web. This is because annotating offerings on the Web requires much more complex statements than “Resource A is an instance of the product class TV set”. We need to express the business relation between such a resource and the legal entity making that offering: Are they offering to *sell* this instance, or do they offer it for rent? For which audience is this offer intended and valid, both in terms of eligible regions (Austria and Germany only, all of Europe, worldwide,...) and eligible types of buyers (end users, wholesalers, ...)? Also, we need to be able specify a wealth of terms and conditions.

Besides that, we need to be able to differentiate between (a) information resources representing the description of an offering, and (b) the offering itself.

In short, we currently lack an ontology that allows describing the *relationship* between (1) Web resources, (2) offerings made by means of those Web resources, (3) legal entities, (4) prices, (5) terms and conditions, and the aforementioned ontologies for products and services (6).

1.2 Contribution

In this paper, we analyse the complexity of product description in the Semantic Web and propose the *GoodRelations* ontology that covers the representational needs of typical business scenarios in the commodity segment.

2 REPRESENTATIONAL REQUIREMENTS

In this section, we develop the requirements for the GoodRelations ontology. We first present typical application scenarios and then derive a list of functional requirements. Then, we specify the scope and purpose of the ontology using *competency questions*, which is a standard technique in common ontology engineering methodologies (Uschold & Grüninger, 1996).

2.1 Motivating Scenarios

In the following, we describe a few typical examples of offerings made on the Web.

Scenario 1: A Web resource represents an entity that, *in general*, offers items of a particular kind of good for sale, either to wholesalers or to end users, or both; they might offer concrete, identifiable instances or it may be that it is only said that such instances exist.

Example 1.1.: „Siemens, described by <http://www.siemens.com>, offers cellphones“

Example 1.2.: „Marmot, described by <http://www.marmot.com>“, produces sleeping bags (and, implicitly, sells them to wholesalers only).“

Example 1.3.: „Mediamarkt, described by <http://www.mediamarkt.at>, offers to sell one remaining tumble drier MIELE xyz123 at € 999 on the Web page

<http://www.mediamarkt.at/clearance/mielexyz123>.“

Scenario 2: A Web resource describes *the make and model of a commodity* and its properties. Such Web resources are usually within the domain name space of the respective manufacturer. There may exist actual individuals of this make and model, which share the default properties defined by the make and model, but also have additional properties of their own (e.g. the date of production, the serial number).

Example 2.1: „<http://www.sony.de/cellphones/sony123> describes the Sony cell phone model 123, which is green and manufactured by Sony.“

Example 2.2: „<http://www.marmot.com/bags/ultralightBag> describes the Marmot sleeping bag model *Ultralight Bag*. Such a sleeping bag weighs 850 grams.“

Note: For an in-depth discussion of the semantics of makes and models, see section 3.1.5.

Scenario 3: A Web resource represents (1) the description of a particular make and model of a commodity and its properties, and (2) a concrete offer to sell unidentified instances thereof. Such Web resources are usually within the domain name space of Web shops.

Example 3.1: „<http://www.mycellphoneshop.com/sony/sony123> describes the Sony cell phone model 123, which is green and manufactured by Sony. Instances can be ordered via this page for 100 euro by end users or wholesalers.“

Example 3.2: „<http://www.outdoorwholesale.com/bags/ultralightBag> describes the Marmot sleeping bag *Ultralight Bag*, which weighs 850 grams, and instances can be ordered, by wholesalers only, via this page for 200 euro.”

Scenario 4: A Web resource represents (1) the description of a particular range of products, determined either by product classes or makes and models, and property ranges, and (2) a concrete offer to *rent out* unidentified instances thereof. Such Web resources are usually within the domain name space of rental agencies or Web pages of local dealers.

Example 4.1: „<http://www.outdoorwholesale.com/tents-for-rent> describes that the shop described by <http://www.outdoorwholesale.com/> offers for rent Marmot sleeps-two tents, which weigh 2850 grams, via this page for 20 euro a day.”

Scenario 5: A particular Web resource represents (1) the description of a particular range of products, determined either by product classes, makes and models, or property ranges, and (2) a concrete offer to provide a certain type of service for this range of products (e.g. maintenance, repair, disposal). Such Web resources are usually within the domain name space of local dealers.

Example 5.1: „<http://www.volkswagen-sanfrancisco.com/service> describes that the dealer described by <http://www.volkswagen-sanfrancisco.com/> offers to repair any Volkswagen make and model.”

Example 5.2: „MyWebshop, described by <http://www.mywebshop.com>, sells red and green cellphones and repairs any cellphone made by Nokia or Siemens“.

2.2 Requirements

In this section, we summarize functional requirements on the GoodRelations ontology.

R1: The ontology must differentiate between business entities, offerings, makes and models, and actual product or services instances.

Example: <http://www.siemens.de/cellphone123> may at the same time describe (1) a make and model and (2) the offering to sell this make and model.

R2: The ontology must differentiate between information resources (that contain human-readable information about conceptual elements) and the actual objects.

Example: <http://www.mywebshop.de> is the home page of a given Web shop but at the same time the retrievable description of a concrete offering.

R3: The ontology must provide a vocabulary for specifying the type of business function that is made in the offering, e.g. to sell (or for services: to provide), to rent out, to repair, to maintain, or to dispose a set of goods or an individual object.

Example: A company may offer to sell or to repair pianos.

R4: The ontology must be suitable both for offerings relating to identified single product instances and anonymous sets of instances of a category of products which are only existentially quantified.

Example: A certain Web page may offer the sales of a clearly identified individual (e.g. a certain used Siemens S65 cell phone), or just state that they normally sell instances of a given kind (but those instances are not exposed on the Web).

R5: A single query must return matching offers of both identified product instances and offerings that relate to anonymous instances.

Example: When we search for offers of Siemens S65 cellphones, we expect both specific offers related to identified product instances (e.g. in a classified add) and offers relating to a set of anonymous instances of a different kind (e.g. offers of new products by a shop).

R6: It must be possible to query explicitly for either identified product instances or offerings that relate to anonymous instances only.

Example: We want to search for explicit offers of identified Volkswagen Golf cars only, leaving out general statements.

R7: For quantitative properties of product models and actual products (e.g. screen size of TV sets), it must be possible to represent and query for intervals of values (e.g. "cell phones that weigh between 50 and 100 grams").

Example: We want to search for a TV set that weighs less than 5 kg but has a screen size of at least 10 inches.

R8: For quantitative properties of product models and actual products, it must be possible to represent units of measurements (e.g. meter, inches, pounds, ...). (The conversion between alternative units of measurement needs not to be provided by the ontology, but will be facilitated due to a proper representation.)

Example: A certain UK brand weighs one pound, a German brand weighs 400 grammes. The unit of measurement should not be hard-wired with the product attributes (as it is the case in version 5.1 of eClassOWL).

R9: The ontology should be compatible with existing products and services ontologies, namely eClassOWL. If modifications on those ontologies are inevitable, they should be minimal. A common upper ontology for products and services ontologies would be beneficial.

eClassOWL is currently the largest ontology for products and services. The GoodRelations ontology should work with eClassOWL elements as well as with other products and services ontologies.

R10: The eligible target audience in terms of geopolitical areas and type of customer must be expressable.

Example: A company may offer to sell a certain microwave to end-users in Austria and Switzerland only.

R11: The supported shipping and delivery methods must be expressable.

Example: It must be expressed whether pick-up is possible.

R12: The supported methods of payment must be expressable.

Example: Some shops accept certain credit cards, some don't. We must be able to specify our constraints.

R13: The warranty promise included in an offer must be expressable, both in terms of its duration from the date of purchase and its scope (parts only; parts and labor; parts, labor and pick-up;...).

Example: A certain offer includes a warranty against defects for 12 months after the day of purchase; this warranty includes parts and labor and pick-up at the customer's location. For another 24 months, the vendor promises a parts-and-labor-only warranty.

R14: The price per unit must be specifiable for arbitrary quantity ranges. All common units of measurement should be supported for the quantities (pieces, pounds, liters,...).

Example: A pound of butter is € 2 per pound for 1 – 2 pounds.

R15: Different prices for non-overlapping quantity ranges should be specifiable.

Example: A pound of butter is € 2 per pound for 1 – 2 pounds and € 1.8 for 3 pounds and more.

R16: Different prices for different regions should be specifiable.

Example: A pound of butter is € 2 per pound when sold to Austria and € 2.5 when sold to Germany.

R17: Different prices for different types of customers should be specifiable.

Example: A pound of butter is € 2 per pound for end-users and € 1.8 for public institutions.

R18: The currency to which the price relates should be specifiable.

Example: A price may be expressed in GBP, Euro, Dollar, etc.

R19: The validity period for the offering should be specifiable in the form of a starting and ending date and time. (A single time-zone, e.g. GMT, for all such time date may be acceptable).

Example: A certain offer may be valid only from 2007-10-01 00:00 CET through 2007-12-31 11:59 CET.

R20: Shipping charges per order and per delivery mode should be specifiable.

Example: The shipping charge is € 5 per shipment when shipped via UPS.

R21: Payment charges per order and per method of payment should be specifiable.

Example: Payment by credit card costs an extra charge of € 5 per order.

R22: It should be possible to offer bundles of goods.

Example: A Webshop offers a bundle consisting of one Siemens S65 and a Siemens travel charger C65 for € 99.

Note: Usually, no breakdown of the total price per part included is given in such cases.

R23: The ontology should be based on current Semantic Web standards with mature tooling support, so that it can be implemented on current SW technology pillars.

In particular, it should be based on such W3C ontology formalisms for which mature infrastructure in terms of storage and reasoning is available.

R24: It should be possible to state that a certain product instance or product make and model is a consumable, accessory, or spare part for a second product make and model.

Example: The toner cartridge Brother B5677 is a consumable for the Brother printer A1234.

R25: It should be possible to state that one product instance or product make and model is a potential substitute for a second product instance or product make and model.

Example: The toner cartridge Pelikan A1234 is a potential substitute for the toner cartridge Brother B5677.

Note: From a vendor perspective, such statements may be undesirable.

R26: The ontology should support the annotation of both tangible commodity products and commodity services, like hair-cutting or car maintenance. It is not necessary to support the full description of Web services or arbitrary business services.

Example: It should be possible to express that a certain business entities provides car maintenance, oil change, or cleaning of Volkswagen cars.

2.3 Competency Questions

In the following, we give competency questions (cf. Uschold & Grüninger, 1996) for defining the scope of the GoodRelations ontology.

CQ1: Which retrievable Web resources describe an offer

- {to sell | to provide the service of | to repair| to maintain or service | to lease out | to dispose}
- {a concrete individual | some unknown individuals} of
- a {given good | given service | spare part for a given good | consumables and supplies for a given good} described by a {type of good | specific make and model}
- that meet certain requirements on {properties | intervals for properties}
- for which the offering party accepts a given method of payment and

- provides a certain method of delivery
- to {consumers|retailers}
- in a given {country| region}?

CQ2: For which time-frame is the offer valid?

CQ3: Which types of customers are eligible?

CQ4: Which are the eligible customer regions?

CQ5: Which shipping / delivery methods are available?

CQ6: Which methods of payment are accepted?

CQ7: For any such offer, what is the price and currency for a given quantity, delivery region, and type of customer, per unit of measurement? Does the price include VAT?

CQ8: What is the shipping charge and currency for a given delivery method to a given region? Does it include VAT?

CQ9: What is the payment charge and currency for a given payment method? Does it include VAT?

CQ10: What is the mail address and which are the contact details of the offering business entity?

CQ11: Which are the locations from which the product or service is being provided, what are the contact details and opening hours of each location?

CQ12: What is the scope and duration of the warranty promise or warranty promises included in the offer?

CQ13: Which offerings on the Web refer to {spare parts | consumables or supplies} for a given {type of good | make and model}?

CQ14: What are the textual description and the language of the textual description of the respective offering?

CQ15: For a given item, what is the vendor-specific item number for ordering?

3 THE GOODRELATIONS ONTOLOGY

In the following, we describe the GoodRelations ontology, which aims at complementing products and services ontologies. The GoodRelations ontology in combination with a domain-specific products and services ontology should be sufficient for expressing typical offerings made on Web pages of manufacturers, Web shops, e-marketplaces, etc., and for expressing specific queries for items of a known kind.

Note that it is out of the scope of GoodRelations and eClassOWL to help translate from individual buyer goals into suitable types of products. Searching with GoodRelations and eClassOWL requires that the category of product or service is already known. However, it is possible to create mappings between an ontology of goals on one hand, and ontology classes for product categories on the other hand. Also, we expect that Web services will evolve that can translate from a given consumer goal to a set of product categories plus appropriate constraints on properties, same as recommender services are provided already as of today.

The structure of this section is as follows. First, we borrow the presentation style from Ushold et al. (Ushold, King, Moralee, & Zorgios, 1998) and start with an overview of the core conceptual entities and their relationships. Then, we define all elements by giving an informal definition, complemented by examples and additional explanations. Second, we try to find existing standards and specifications that can be re-used and which should be integrated. Third, we discuss particular modeling challenges. Fourth, we describe the coding of the ontology in OWL-DLP.

3.1 Domain Capture

In the following, we outline the relevant types of conceptual entities in the domain.

Figure 1 gives an overview of the relevant conceptual entities and the types of relevant relationships. Note that this entity-relationship diagram is not yet the final ontology coding but just a part of the domain capture. We will see later that certain limitations of OWL, namely the lacking support for relations with a higher arity than two, require modeling workarounds that introduce new conceptual elements.

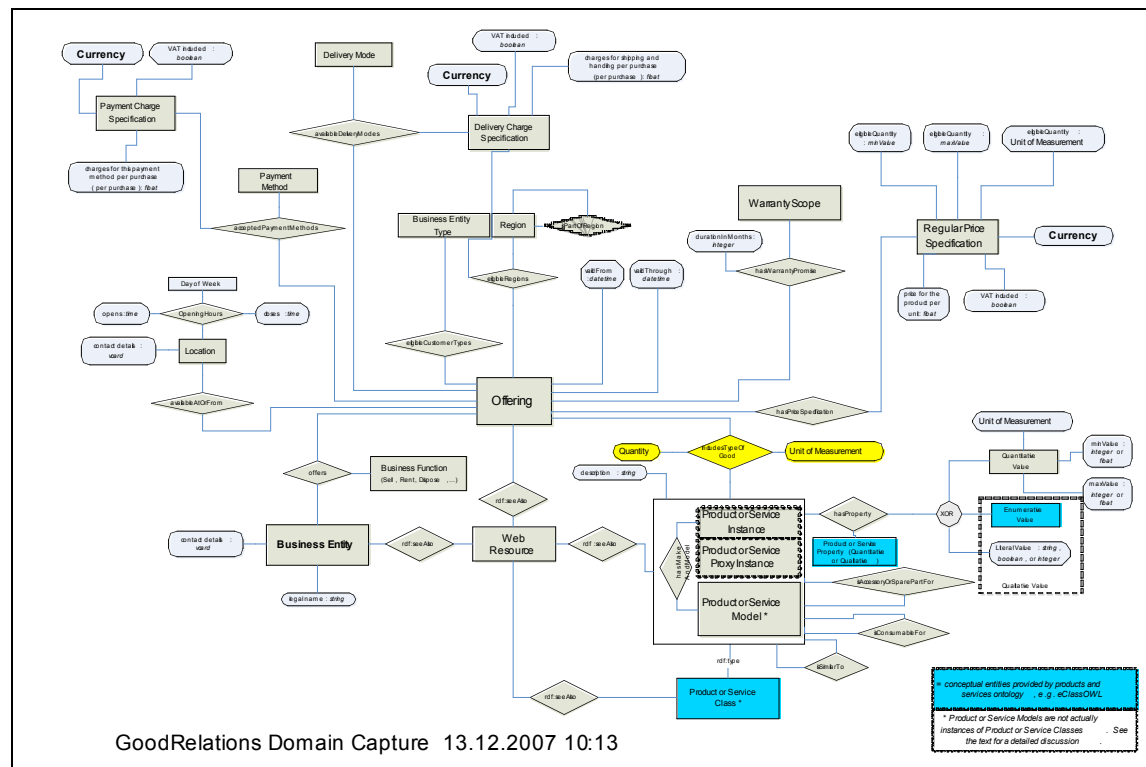


Figure 1. Conceptual Model of Web Offerings (for a larger version, see Appendix A)

3.1.1 Web Resource

A retrievable Web resource that contains information related to a business entity, an offering, a product model, or similar. A Web Resource is characterized by an URI that describes a retrievable resource (i.e., this URI is an URL). While in this document, we assume that a Web Resource is either a complete Web document or a fragment therein, it is theoretically possible to address individual parts of a Web document in the form of XPointer ranges (see <http://www.w3.org/XML/Linking>).

Examples: The pages available at <http://www.springer.com> and <http://www.springeronline.com/3-540-47703-9> are Web Resources.

Notes:

1. Very often, a Web Resource combines information about multiple conceptual entities, e.g. about the offering party and the types of products. This is why we need to clearly distinguish Web Resources from the other conceptual elements that are part of offerings on the Web.
2. In the GoodRelations ontology, we assume Web Resources to be instances of `rdfs:Resource` and use the `rdfs:seeAlso` property for linking the actual conceptual entities (business entities, offerings, product instances, etc.) to Web Resources that contain a human-readable description. This is why there is no class "Web Resource" in the GoodRelations ontology.

3. In some rare cases, the Web Resource may precisely define a single relevant conceptual entity. In this case, `rdfs:isDefinedBy` as a subproperty of `rdfs:seeAlso` could be used. However, we discourage this.

3.1.2 Business Entity

An instance of this class represents the legal agent making a particular offering. This can be a legal body or a person. A Business Entity has at least a primary mailing address and contact details. For this, typical address standards (vCard) and location data can be attached. The location may be important for finding a supplier within a given distance from our own location.

Example: Siemens Austria AG, Volkswagen Ltd., Peter Miller's Cellphone Shop

Note: Typical address standards (vcard) and location data should be attached to a business entity. Since there already exist established vocabularies for this, the GoodRelations ontology does not provide respective attributes. Instead, the use of existing vocabularies is recommended.

3.1.3 Offering

An Offering represents the public, not necessarily binding, not necessarily exclusive, announcement by a Business Entity to provide a certain Business Function for a certain Product or Service Instance to a specified target audience. An Offering is specified by the type of product or service or bundle it refers to, which Business Function is being offered (sales, rental, ...), and a set of commercial properties. It can either refer to a clearly specified instance (Product or Service Instance) or to a set of anonymous instances of a given type (existentially quantified Product or Service Instances, see also section 3.3.3). An offering may be constrained in terms of the eligible type of business partner, countries, quantities, and other commercial properties. The definition of the commercial properties, the type of product offered, and the business function are explained in the following sections in more detail.

Example: Peter Miller offers to repair TV sets made by Siemens, Volkswagen Innsbruck sells a particular instance of a Volkswagen Golf at \$10,000.

3.1.4 Business Function

The Business Function specifies the type of activity or access offered by the Business Entity on the Product or Services through the Offering. The idea of standardizing business functions was first put to practice by the UNSPSC Business Functions Identifiers (United Nations Development Programme, 2003). We take their basic types of business functions as a starting point.

Example: A particular offering made by Miller Rentals Ltd. says that they (1) sell Volkswagen Golf convertibles, (2) lease out a particular Ford pick-up truck, and (3) dispose car wrecks of any make and model.

Sell

This Business Function indicates that the Business Entity offers to *transfer permanently all property rights* on the specified Product.

Lease Out

This Business Function indicates that the Business Entity offers to *temporarily grant the right to use* the specified Product.

Maintain

This Business Function indicates that the Business Entity offers to *carry out typical maintenance tasks* for the specified Product. Maintenance tasks are actions that undo or compensate for wear or other deterioration caused by regular usage, in order to restore the originally intended function of the product, or to prevent outage or malfunction. In other words, the Business Entity expresses to be usually able and willing to maintain an object *x* if *x* is an instance of the given class of product.

Repair

This Business Function indicates that the Business Entity offers to *try to evaluate the chances for repairing, and, if positive, repair* the specified Product. Repairing means actions that restore the originally intended function of a product that suffers from outage or malfunction. In other words, the Business Entity expresses to be usually able and willing to repair an object *x* if *x* is an instance of the given class of product.

Provide Service

This Business Function indicates that the Business Entity offers to provide the type of Service.

Note: Maintain and Repair are also types of Services. However, eClassOWL and other ontologies provide classes for tangible products as well as for types of services. The business function Provide Service is to be used with such goods that are Services, while Maintain and Repair can be used with goods for which the class of product exists in the ontology, but not the respective type of service.

Example: Car maintenance could be expressed both as “Provide Service Car Maintenance” or “Maintain Cars”. Since existing ontologies for goods often tangle products and services, it seems beneficial to include Provide Service as a business function.

Dispose

This Business Function indicates that the Business Entity offers to *accept the specified Product for proper disposal, recycling, or any other kind of allowed usages*, freeing the current owner from all rights and obligations.

Buy

This Business Function indicates that the Business Entity is in general *interested in purchasing* the specified Product.

3.1.5 Products or Services: Instances, Models, and Classes

In the products and services domain, we find multiple types of conceptual entities when it comes to describing what is being offered:

First, actual products, like for example my cellphone or a concrete TV set.

Second, certain product makes and models, e.g. the cellphone make and model Sony 1234 or the car model Ford T. There usually exist actual products that are of the respective make and model, but they all have an identity of their own. In particular, they differ in several properties.

Third, classes of actual products that are similar in function or nature, like for example the class “cellphone” which subsumes all actual cellphones.

In principle, we find the same pattern in services as well. For example, there is the class “haircutting” which subsumes all instances of someone actually cutting someone else’s hair. In some cases, there may also be makes and models for services, but this is not very common in our opinion.

Accordingly, the GoodRelations ontology should contain the following three conceptual elements

Product or Service Instance

A Product or Service Instance is a single identifiable object or action that creates some increase in utility (in the economic sense) for the individual possessing or using this very object (Product) or for the individual in whose favor this very action is being taken (Service). Products or Services are types of goods in the economic sense. For an overview of goods and commodities in economics, see (Milgate, 1987).

Examples: MyThinkpad T60, the pint of beer standing in front of me, my Volkswagen Golf, the haircut that I received or will be receiving at a given date and time.

Note: In many cases, product or service instances are not explicitly exposed on the Web but only existentially quantified. For a detailed discussion and practical solutions, see section 3.3.3.

Product or Service Model

A Product or Service Model is an intangible entity that specifies some characteristics of a group of similar, usually mass-produced Products. In the case of mass-produced Products, there exists a relation hasMakeAndModel between the Products and Services Instance and the Product or Service Model.

A Product or Service Model may carry information on characteristics or features of actual products of this make and model. For example, a certain camera make and model may specify the default weight of an actual camera. It is very likely that an actual camera of that make and model will show that weight; however, it is not guaranteed. If I cut off a piece of that camera, the camera will have a different weight but still be a camera, and still be a camera of the given make and model. So the axiomatization of the properties specified at the level of a Product and Service Model is:

```
FORALL A instanceof ProductOrService, B instanceof
ProductOrServiceModel, B hasDefaultValue (Value, Feature), and A
hasMakeAndModel B --> A hasFeature(Value, Feature)
```

Example: Ford T, Thinkpad T60, Volkswagen Golf

Note: From the ontological perspective, Product or Service Models are not subclasses of Products or Services, but intangible objects in their own right (A particular model of a camera is not a camera, and the set of individual cameras of the same make and model is not semantically equivalent to the camera model. The set of all Ford Ts is not the same as the make and model Ford T). The main reason is that the model has an identity of its

own, i.e. there may exist instances that meet all property restrictions of this model may still not be of this make and model. For example, if we defined Volkswagen Golf as a subclass of the class car with constraints on the manufacturer („Volkswagen“) and on a finite set of features (max. speed, weight, length,...), then there could be objects in the universe that meet all of these constraints and are still not a Volkswagen Golf. Similarly does a particular camera model not have a weight in the same way as a camera has a weight: The model may specify a default interval for the weight of mass-produced cameras, but itself does not have a weight at all, since it is an intangible object, not subject to gravity. (However: Why is the set of all lions equivalent to the species lion? Mainly, because a species is a subset of animals and not more than that, while a model is not just a set but a conceptual entity in its own right. A species exists only if there has ever been an individual of this species. There are no „abstract species“.)

For a formal justification of that distinction, please refer to the OntoClean methodology (Guarino & Welty, 2002, 2004).

Product or Service Class

A Product or Service Class is a set of Product or Services Instances that provide the same functionality, can be used for achieving the same goal, or are similar in their physical characteristics.

Examples: Laptop computer, car, writing paper, hot beverage, haircutting as a service

3.1.6 *Product or Service Properties*

Product or Services Instances usually have certain characteristic features, often physical, chemical attributes or such related to the context of usage. The following conceptual elements are relevant in the context of the GoodRelations ontology.

Product or Service Property

The type of a characteristic feature of a Product or Service Instance, like physical or chemical attributes or such related to the context of usage. The same Product or Service Property relates to the same phenomenon (e.g. chemical or physical) or behavior but may refer to different Units of Measurement. A Product or Service Property is a relation between a Product or Services Instance and a Value. The Value may be a Quantitative Value or a Qualitative Value.

If the Value is a Quantitative Value, a Product or Service Property is a *ternary* relation between a Product or Services Instance, a Unit of Measurement, and this Quantitative Value. If it is a Qualitative Value, it is a binary relation between a Product or Services Instance and this Qualitative Value.

Products or Services Classes may be defined by constraints on Product or Services Properties. Product or Services Models may provide default Values for Product or Services Properties (for the axiomatization of the latter, see above).

Examples: weight, screenSize, maxSpeed, thickness, water-proof

Quantitative Value

A Quantitative Value is a numerical interval that represents the range of a certain quantitative Product or Service Property in terms of the lower and upper bounds for one particular Product Or Service Instance. It is to be interpreted in combination with the respective Unit Of Measurement. Most quantitative values are intervals even if they are in practice often treated as a single point.

Quantitative Values may refer to real or integer numbers or any ordinal scale.

Example: myLaptop hasWeight(thisValue, kg), thisValue instanceof Quantitative Value, thisValue hasMinValue 1.0, thisValue hasMaxValue 1.1

Qualitative Value:

A Qualitative Value is an entity that represents the state of a certain qualitative Product or Service Property. Qualitative Values are either Literal Values or Enumerative Values.

Literal Value

A Literal Value is an entity that represents the state of a certain qualitative Product or Service Property for one particular Product Or Service Instance. In practice, this refers to a few integer properties for which the integer value represents qualitative aspects, for string datatypes, and for boolean datatype properties. In OWL, Qualitative Values are simple integer, string, or boolean literals.

Example: myCellphone hasSerialNumber "S1234", myWristWatch waterproof true

Enumerative Value

An Enumerative Value is an entity that represents the state a certain qualitative Product or Service Property. The same Qualitative Value can be referred to by multiple Product or Service Instances.

Example: green, leftOpeningDoorType

Unit of Measurement

Units of Measurement specify the point of reference and the scale for Quantitative Values. They can refer to technical/physical or commercial perspectives (e.g. sales units). Work on standardizing Units of Measurement has been ongoing in various standardization bodies, but there is still no single, widely adopted standard in place. A common approach is the UN/CEFACT Recommendation No. 20 (United Nations Economic Commission for Europe (UN/CEFACT), 2006), which distinguishes between

- a) Base and Derived SI Units (Normative)
- b) Common Use / SI Equivalent Units ("Normative Equivalent"), and
- c) Informative Units.

The latter include (taken from (United Nations Economic Commission for Europe (UN/CEFACT), 2006)):

- a) Qualified Base Units
- b) Sales Units
- c) Packing Units
- d) Shipping and Transport Units
- e) Industry Specific Units
- f) Information Technology Units
- g) Integers Numbers Ratios
- h) Multiples Fractions Decimals

See also the following figure (taken from (United Nations Economic Commission for Europe (UN/CEFACT), 2006, p. 3):

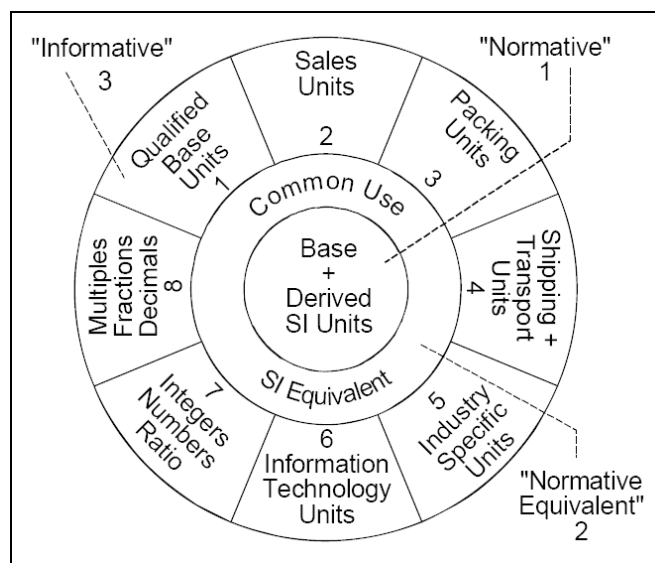


Figure 2. UN/CEFACT Units of Measure Schema Components (Copyright UN/CEFACT) (United Nations Economic Commission for Europe (UN/CEFACT), 2006, p. 3)

The UN/CEFACT recommendation specifies unique identifiers for 1189 units of measurement (United Nations Economic Commission for Europe (UN/CEFACT), 2006) as of March 16, 2006. As a primary key, the unique “Common Code” can be used. The common code is defined as follows (United Nations Economic Commission for Europe (UN/CEFACT), 2006, p. 6):

„The representation format for the code values shall be alphanumeric variable length 3 characters (an..3); wherever possible, existing code values are retained according to the following order of precedence for assigning values:

a) alphabetic code values for units of measure as in UN/ECE Recommendation 20, edition 1985

b) alphanumeric code values for units of measure as in ANSI ASC X12 data element number 355

NOTE: Where there are both UN/ECE Recommendation 20 and ASC X12 data element number 355 code values for a unit of measure, the UN/ECE Recommendation 20 code value only is retained.

c) code values for new units of measure shall be allocated by the UN/ECE Secretariat, typically based on sequential coding according to the format Alpha-Numeric-Numeric (ann) starting with A01 up to Z99.“

3.1.7 Commercial Properties of the Offering

An Offering is often more specific than just stating the general availability of a given type of product or service. In many cases, it is desirable to specify additional commercial aspects. Related work to this issue has been done in the context of formats for catalog data exchange. Kelkar et al. (Kelkar, Leukel, & Schmitz, 2002), for example, list prices, quantities, eligible regions, and eligible types of business partners as relevant details.

In the context of the GoodRelations ontology, the following aspects are most relevant:

Validity

Validity is a time interval that specifies the period during which the offer is valid. Valid does only imply that the Business Entity is in principle keeping the Offering active. It does not imply that the Business Entity gives a binding promise to provide the respective Business Function on the range of product or services.

Example: A certain offering is valid from June 1, 2007, 0:00 through December 31, 2007, 23:59 CET.

Price Specification

A Price Specification is a conceptual entity that specifies the price asked for a given Offering by the respective Business Entity. An Offering may be linked to multiple Price Specifications that specify alternative prices for non-overlapping sets of conditions (e.g. quantities or sales regions). For GoodRelations, we take the price model from Kelkar, Leukel, and Schmitz (2002, (Kelkar, Leukel, & Schmitz, 2002)) as a starting point and reuse a subset thereof. Contrary to their approach, we assume that “deliveryRegion” (=eligible region) is an attribute of an Offering, not of a Price Specification. It may be that there are Offerings with Price Specifications varying by Country or Region; however, we think that these are rather multiple offers than multiple prices. The same holds for the validity time frame (“ValidityTimePeriod”) and the type of customers (“CustomerType”). We think that discounts during a particular time interval or for a particular type of customers are rather alternative Offerings than just additional Price Specifications for the same Offering. Thus, we suggest attaching those attributes to the Offering, not to the Price Specification.

A Price Specification is characterized by (1) the lower and upper limits and the Unit of Measurement of the eligible quantity, (2) by a monetary amount per unit of the Product or Service Instance in the given Unit of Measurement specified as a literal value of type float in combination with a Currency, and (3) a whether this prices includes local sales taxes, namely VAT.

Example: The price, including VAT, for 1 kg of a given material is 5 Euros per kg for 0 – 5 kg and 4 Euros for quantities above 5 kg

Note: Due to the complexity of pricing scenarios in various industries, it may be necessary to create extensions of this fundamental model of Price Specifications. Such can be done easily by importing and refining the GoodRelations ontology.

Currency

A Currency is a Unit of Measurement for monetary values. For currencies, we suggest using the well-established ISO 4217 standard (ISO, 2001), which is the recommended encoding for currencies in international payment transactions.

Example: EUR for Euro, CHF for Swiss Francs, USD for US Dollars

Note: The UN/CEFACT recommendation does not specify currencies, though they are in principle also Units of Measurement.

Warranty Promise and Warranty Scope

A Warranty Promise is an entity representing the duration and scope of services that will be provided to a customer free of charge in case of a defect or malfunction of the Product or Service Instance. A Warranty Promise is characterized by its temporal duration (usually starting with the date of purchase) and its Warranty Scope. The Warranty Scope represents the types of services provided (e.g. labor and parts, just parts) as part of the warranty included in an Offering. The actual services may be provided by the Business Entity making the Offering, by the manufacturer of the Product, or by a third party. There may be multiple Warranty Promises associated with a particular Offering, which differ in duration and scope (e.g. pick-up service during the first 12 months, just parts and labor for 36 months).

Examples: 12 months parts and labor, 36 months parts only

Payment Method

A Payment Method is a standardized procedure for transferring the monetary amount for a purchase. Payment Methods are characterized by the legal and technical structures used, and by the organization or group carrying out the transaction.

Examples: Concept PaymentByCreditCard subConcept of PaymentMethod; Visa, Mastercard, Diners, ... instanceOf PaymentByCreditCard

Delivery Method

A Delivery Method is a standardized procedure for transferring the Product or Service Instance to the destination of fulfilment chosen by the customer. Delivery Methods are characterized by the means of transportation used, and by the organization or group that is the contracting party for the sending Business Entity (this is important, since the contracted party may subcontract the fulfilment to smaller, regional businesses).

Examples: Delivery by Mail, Delivery by Direct Download, Delivery by UPS

Delivery Charge Specification

A Delivery Charge Specification is a conceptual entity that specifies the additional costs asked for delivery of a given Offering using a particular Delivery Method by the respective Business Entity. A Delivery Charge Specification is characterized by (1) a monetary amount per order specified as a literal value of type float in combination with a Currency, (2) the Delivery Method, (3) the target Country or Region, and (4) a whether this charge includes local sales taxes, namely VAT.

An Offering may be linked to multiple Delivery Charge Specifications that specify alternative charges for disjoint combinations of target Countries or Regions and Delivery Methods.

Examples: Delivery by direct download is free of charge worldwide, delivery by UPS to Germany is 10 Euros per order, delivery by Mail within the US is 5 Euros per order.

Payment Charge Specification

A Payment Charge Specification is a conceptual entity that specifies the additional costs asked for settling the payment after accepting a given Offering using a particular Payment Method. A Payment Charge Specification is characterized by (1) a monetary amount per order, specified as a literal value of type float in combination with a Currency, (2) the Payment Method, and (3) a whether this charge includes local sales taxes, namely VAT.

An Offering may be linked to multiple Payment Charge Specifications that specify alternative charges for various Payment Methods.

Examples: Payment by VISA or Mastercard costs a fee of 3 Euros including VAT, payment by bank transfer in advance is free of charge.

Business Entity Type

A Business Entity Type is a conceptual entity representing the legal form, the size, the main line of business, the position in the value chain, or any combination thereof, of a Business Entity. From the ontological point of view, Business Entity Types are mostly roles that a Business Entity has in the market. Business Entity Types are important for specifying eligible customers, since Offerings are often meant only for Business Entities of a certain size, legal structure, or role in the value chain.

Examples: Consumers, Retailers, Wholesalers, or Public Institutions

Country or Region

Countries or Regions are geographical or geopolitical areas. In GoodRelations, they are used for specifying the areas for which the Offering is valid. Many Business Entities will accept orders from or deliver Products or Services Instances to selected Countries or Regions only – for practical or legal reasons. One Region may be located fully within

another Region. In this case, a laysCompletelyWithin relation exist between the two, which is transitive.

GoodRelations could reuse several approaches for ontologies of regions and places for specifying Countries and Regions. However, we suggest a more pragmatic approach of reusing the ISO Standard 3166, in particular ISO 3166-1 (ISO, 2006) and ISO 3166-2 (ISO, 1998). The first defines 2- or 3-letter identifiers for existing countries and a few independent geopolitical entities. ISO 3166-1 alpha-2 defines 2-letter codes for most countries. There exist alternative standards with 3-letter codes and a numerical representation. For the following reasons, we suggest using the 2-letter codes: First, they are well established and people are likely more familiar with them (they are also used for most top-level domains). Second, and more important, the 2-letter variant is the basis for ISO 3166-2, which breaks down the countries from ISO 3166-1 into administrative subdivisions (ISO, 1998). The code elements used in ISO 3166-2 consist of “the alpha-2 code element from ISO 3166-1 followed by a separator and a further string of up to three alphanumeric characters e. g.” (from: <http://www.iso.org/iso/en/prods-services/iso3166ma/04background-on-iso-3166/iso3166-2.html>).

This allows using simple string operations on the respective ISO 3166 codes in order to handle administrative subdivisions. For example, if a certain Offering is said to be valid for Canada (ISO 3166-1 two-letter code “CA”), then one can infer that any longer search string specifying an administrative subdivision of Canada (e.g. British Columbia, ISO 3166-2 “CA-BC”) is also an eligible region.

Examples: Canada (CA), Austria (AT), Canada: British Columbia (CA-BC), Italy (IT), Italy: Province of Milano (IT-MI)

Note: More complex modeling of Countries and Regions may be useful in some scenarios, and GoodRelations can be imported and extended if necessary. However, most offerings on the Web contain statements on the level of countries only, for which ISO 3166-1 is sufficient and very common.

Location of Sales or Service Provisioning

A Location of Sales or Service Provisioning is a location from which the specified Business Function on the particular Product or Service Instance is being offered by the Business Entity. Large enterprises often maintain multiple branches from which the delivery or fulfilment can be provided. In this case, the location of the main office of the Business Entity does not state from where a customer can actually get the Offering.

Locations of Sales or Service Provisioning are characterized by an address or position and a set of opening hour specifications for various days of the week.

Example: A rental car company may offer the Business Function Lease Out of cars from two locations, one in Fort Myers, Florida, and one in Boston, Massachusetts. Both stations are open 7:00 – 23:00 Mondays through Saturdays.

Note: Typical address standards (vcard) and location data should be attached to a Location of Sales or Service Provisioning. Since there already exist established vocabularies for this, the GoodRelations ontology does not provide respective attributes. Instead, the use of existing vocabularies is recommended.

3.1.8 Relationships Between Multiple Product Models or Product Instances

In the commodities sector, it is often valuable to express several relationships between Product Instances and/or Product Models. In the following, we specify the most common ones.

Relation isConsumableFor

One Product or Service Model or Product or Service Instance can be a Consumable for another Product or Service Model. Being a Consumable means that the first Product or Service Instance is needed and consumed in the course of operating the second Product or Service Instance.

Examples:

The Product or Service Model myTonerBrand123 isConsumableFor the Product or Service Model Hewlett-Packard Printer 1000.

The Product or Service Instance myToner123 (of unknown make and model) isConsumableFor the Product or Service Model Hewlett-Packard Printer 1000.

Relation isSimilarTo

One Product or Service Model or Product or Service Instance can be similar to another Product or Service Model. Being similar means that they can largely be used for the same purposes or for achieving the same goals. Since similarity between such two objects is to a high degree a subjective judgement, the exact meaning of being similar is no more precise than “see also”.

Examples:

The Product or Service Model myPowerSupply123 isSimilarTo the Product or Service Model SiemensPowerSupply123.

The Product or Service Instance myToner123 (of unknown make and model) isSimilarTo (i.e. practically equivalent) the Product or Service Model Hewlett-Packard Toner 123.

Relation isAccessoryOrSparePartFor

One Product or Service Model or Product or Service Instance can be an Accessory or Spare Part to another Product or Service Model.

Examples:

The Product or Service Model 3rdPartyPowerSupply123 isAccessoryOrSparePartFor the Product or Service Model SiemensLaptop.

The Product or Service Instance my3rdPartyPowerSupply123
isAccessoryOrSparePartFor the Product or Service Model SiemensLaptop.

3.2 Standards and Specifications To Be Reused

In the following, we summarize standards and specifications that reflect some degree of consensus and authority and will thus be reused for GoodRelations if possible. They are important for the practical realisation of the GoodRelations vision for two reasons: First, they reduce the effort for specifying conceptual elements in the ontology. Second, and most importantly, they are widely used in business, which means that they are often already included in corporate databases, from which Semantic Web data is to be generated. This simplifies the export of such data for the enterprises, because there is no need to create complex mappings between internal databases and newly created ontologies for those domains.

3.2.1 Units of Measurement

As already discussed in section 3.1.6, we suggest using the UN CEFAC Recommendation N°. 20 “Codes for Units of Measure Used in International Trade” for Units of Measurement (see http://www.unece.org/cefact/recommendations/rec20/rec20_rev4E_2006.pdf), and thereof Annexes I, II, and III (United Nations Economic Commission for Europe (UN/CEFACT), 2006).

An alternative approach is the “Unified Code for Units of Measure” (see <http://aurora.rg.iupui.edu/~shadow/units/UCUM/>); however, it is way narrower in scope and not that widely in use.

3.2.2 Countries

As explained in section 3.1.7, we suggest reusing the ISO Standard 3166, in particular ISO 3166-1 (ISO, 2006) and ISO 3166-2 (ISO, 1998).

3.2.3 Currencies

As explained in section 3.1.7, we suggest using the well-established ISO 4217 standard (ISO, 2001) for currencies, which is the recommended encoding for currencies in international payment transactions.

3.2.4 Business Functions

The idea of standardizing business functions was first put to practice by the UNSPSC Business Functions Identifiers (United Nations Development Programme, 2003). We take their basic types of business functions as a starting point. UNSPSC for BFI.

3.2.5 Product or Service Categories

GoodRelations will be complemented by an extended version of the eClassOWL ontology (Hepp, 2006b), which provides ontology classes for more than 30,000 Product or Services Classes (in Version 5.1.3). We are also working on an ontology transcript based on the UNSPSC (United Nations Development Programme, 2007).

3.2.6 Contact Details and Addresses

Business Entities and Locations of Sales or Service Provisioning will have addresses and contact details. For that, typical address standards like vCard (<http://www.imc.org/pdi/>) are already available. Since they are already established vocabularies for this, the GoodRelations ontology does not provide respective attributes. Instead, the use of existing vocabularies is recommended.

3.3 Problems and Challenges

When building the GoodRelations ontology based on the domain capture described in the previous section, we face several critical modeling issues, which we will discuss in the following.

3.3.1 URI Disambiguation

It can be expected that the largest share of existing URIs representing offerings on the Web relate to human-readable documents: Web pages that present a certain enterprise, describe one or multiple product models, or actual offerings offerings. Such Web resources that are identified by a URI and “whose essential characteristics can be conveyed in a message” are called “Information Resources”, and the task of retrieving the representation of such an information resource is called “dereferencing” (Lewis, 2007).

Now, when we want to build an ontology for accessing all the product-related data available on the Web, we need unique identifiers not only for the current Web resources, like corporate Web pages or pages in an e-shopping system. Moreover, we need unique identifiers for all core conceptual elements as described in the domain capture in section 3.1. For example, we need unique identifiers for (1) business entities, (2) makes and models, (3) offerings, and (maybe) (4) available product instances. It is tempting to assume that we can simply take the URIs of existing Web resources for those purposes, but this is not advisable in most cases, for two reasons:

First, we need identifiers for both information resources (e.g. Web pages) and non-information resources (e.g. a certain product model); for a full discussion of those issues see e.g. (Booth, 2003) and (Sauermann, Cyganiak, & Völkel, 2007). Second, and more importantly, we will face the problem that the available Web resources tangle multiple conceptual entities – the Web page describing a certain offer has one single URI, but it may contain the description of a product make and model, an offering, a price specification, a warranty promise, all at the same time – and, in the case of a small shop, even the business entity itself.

When browsing the current WWW as humans, we face no problems if the same URI represents multiple things for us. However, when we move on to the Semantic Web, we need unique identifiers for every single entity of interest. We can, for instance, not use the same URI for a Web Offering and its Warranty Promise, since those are two disjoint categories of things.

When using the GoodRelations ontology, we can use the URIs of existing Web resources only as the identifiers for the Web Resource itself, as defined in section 3.1.1. For all other conceptual entities, we need to introduce new identifiers, unless such have already been defined for the Semantic Web elsewhere.

For that, there exist three approaches:

- 1) **Use blank RDF nodes** (W3C, 2004); however, for all entities that may at any later point be linked to other entities this is discouraged, since it is then not possible to refer externally to such a node (Chris Bizer, Cyganiak, & Heath, 2007), which means that data from multiple sources cannot be merged easily. Also, there exist practical problems in using blanks nodes in several tools.
- 2) **Create new URIs for all significant entities.** When doing so, one must keep in mind several guidelines:
 - i. One should have authority to define the meaning of the respective URI. For URIs in the HTTP namespace, this requires that one is the owner of the respective domain name space. For example, no one except for Sony or some agent authorized to act on Sony's behalf can introduce a new URI <http://www.sony.com/model1234/> even if that was needed for the Semantic Web. This is the cleanest approach; however, it requires some extra work in server configuration to make sure that entering this URI in a browser will return a related information resource.
 - ii. One should obey all "standard" guidelines for good URIs, as described in (Berners-Lee, 1998), and extended in (Sauermann, Cyganiak, & Völkel, 2007). In short, the URIs used should be designed so that there is no need to change them anytime soon. In particular, parameters of dynamic Web pages and file extensions should be avoided at any cost.
- 3) **Create hash URIs derived from the URI of the original Web resource.** In RDF, adding fragment identifiers to a URI creates a new identifier. So if we have a single retrievable Web resource identified by and retrievable at <http://www.mysmallshop.com/>, which represents a shop, some products, and their offerings, then we could introduce derived hash URIs like <http://www.mysmallshop.com/#businessEntity> for the Business Entity, <http://www.mysmallshop.com/#productModel> for the Product Model, <http://www.mysmallshop.com/#offering> for the Offering, and so forth. A nice side effect of this approach is that typing any of those URIs into a standard Web browser will return the original Web page, without the need to implement server redirects (or have the application follow `rdfs:seeAlso` links). It must be noted that also in this case, one must have authority to introduce new URIs in the respective domain space. So, in our example, only the owner of the domain name "mysmallshop.com" has authority to do so.

While option 1 is not recommended for all significant entities, both alternatives 2 and 3 are possible, and it will depend on the environment which one is more recommendable. GoodRelations is agnostic of which approach you take. GoodRelations makes no assumption on who defines the URI for a respective conceptual element; it is just necessary that distinct identifiers for distinct entities exist. Blank nodes are acceptable for entities that will never be linked to outside data (e.g. workarounds for n-ary relations, see section 3.4.2). For compliance reasons with the W3C Architecture, this means practically that the owner of the domain space should create URIs for all significant individual elements, be they fragment (hash) URIs of existing ones or entirely new URIs.

Again: **The important thing is that distinct conceptual entities must use distinct URIs as identifiers.** The URI of the current, retrievable Web resource should become the URI of the conceptual entity Web Resource, and other conceptual entities, namely Offering, should point to this via `rdfs:seeAlso`.

For a full discussion on the issues of naming entities on the Semantic Web, see “Cool URIs for the Semantic Web” (Sauermaun, Cyganiak, & Völkel, 2007), “How to Publish Linked Data on the Web” (Chris Bizer, Cyganiak, & Heath, 2007), “Recipes for Server Configuration” (Miles, Baker, & Swick, 2006), the classic “Cool URIs don’t change” (Berners-Lee, 1998), and “Dereferencing HTTP URIs” (Lewis, 2007).

3.3.2 *Multiple Offerings in the Same Retrievable Resource*

A related problem to the former is that a (larger) retrievable Web resource that contains multiple offerings in the same page. In addition to the requirements from the previous section, we may want to refer to ranges in the document when specifying the human-readable resource of the offer.

This can be solved by two different approaches. The first approach is to create just additional hash URIs based on URI of the original, retrievable resource (and maybe insert respective anchors into the HTML/XHTML document). Example: If two offerings are described in the Web resource

<http://www.mywebshop.com/offering-of-the-month/>,

then we can introduce two derived hash URIs as follows:

<http://www.mywebshop.com/offering-of-the-month/#offering1>

<http://www.mywebshop.com/offering-of-the-month/#offering2>

Alternatively, one could consider using the XML Pointer Language (Xpointer, see (W3C, 2003)) for referring directly to parts of an XHTML document.

For the moment, we suggest using simple hash URIs in those cases.

3.3.3 *Anonymous Instances and Existential Quantification*

In the vast majority of e-commerce scenarios, the actual Product or Service Instances are not exposed on the Web. For example, if a Web shop says they will sell the Siemens cellphone model s1234 at \$ 100, there is mostly no information resource on the Web

that describes this particular, tangible cell phone. From a logician's perspective, most offerings of commodities on the Web are just existentially quantifying the actual Product or Services Instances. The Web shop in the aforementioned example actually says

"There exists at least one thing X, for which holds ThisWebShop sells X and X instanceof cellphone and X hasMakeAndModel Siemens-s1234".

In such cases, there exists no retrievable explicit information nor non-information resource reflecting the actual cellphone, on which ownership can be obtained, and which will be gone once it has been sold.

Of course, there are situations where actual Product or Service Instances are being exposed on the Web. The most prominent one are eBay auctions – there, a dedicated Web resource describing an actual product exists. However, we can safely assume that in most commodity scenarios, the actual Product or Services Instances are only existentially quantified.

From a theoretical perspective, this is no big deal. From a practical perspective, it is, for two reasons. First, existential quantification increases the computational cost of reasoning. Among popular ontology languages for the Semantic Web, only OWL DL and above and WSM DL support existential quantification. Second, and more importantly, the overhead for modeling Product or Services Properties on the basis of existing products and services ontologies, namely eClassOWL(<http://www.heppnetz.de/projects/eclassowl/>) is significant and modeling existential quantification in OWL is not straightforward for domain experts.

For GoodRelations, we propose a very pragmatic work-around:

- a) We create two top-level ontology classes: (1) ActualProductOrService – an instance of this class is an *actual Product or Service Instance*. (2) ProductOrServicesSomeInstancesPlaceholder – an instance of this class is a placeholder for existentially quantified, anonymous Product or Service Instances.
- b) When we describe an actual Product or Service Instance (e.g. in an eBay offering), we make it an instance of the respective Product or Service Class (e.g. cell phone) and of the class ActualProductOrService. We can then describe this instance using all of the Product or Service Properties provided by the products and services ontologies like eClassOWL.
- c) When we describe an anonymous Product or Service Instance (e.g. "we sell Siemens s1234 cellphones), we create an instance of the respective Product or Service Class (e.g. TV set), and make it also an instance of the class ProductOrServicesSomeInstancesPlaceholder. We can then describe this instance using all of the Product or Service Properties provided by the products and services ontologies like eClassOWL.

- d) A query for a given type of products will return both the instances of ActualProductOrService and of ProductOrServicesSomeInstancesPlaceholder (as described in requirement R5 in section 2.2). Still everybody can see quickly whether this instance is an actual cellphone or a placeholder, while the structure of queries regarding product features remains the same.

This work-around works with very lightweight reasoners and is very well compatible with how current products and services ontologies are designed. One may object that the proxy instances are intangible objects and can thus not be instance of a class of tangible things like products. However, we think that the practical advantages of the approach justify the limited ontological inconsistency.

3.3.4 Ranges for Product and Service Attributes

Most quantitative properties of products or services are actually intervals and not single values. Even simple characteristics like the weight of a tangible object can in principle, only be specified as intervals (even if very small).

Also, there are many scenarios in which queries for suitable products or services must allow ranges in the query and must reason properly about such ranges. For example, if someone is looking for a TV set with a screensize between 10 and 15 inches, a model with 12 inches must be reported as a match, and if someone has a piano that needs to be transported and weighs 120 kg, a company offering to transport all pianos up to 150 kg of weight must be found.

Now, there have been approaches of extending Web ontology languages, namely OWL, by support for datatype ranges. The most prominent approach is the OWL-Eu proposal by Pan and Horrocks (Pan & Horrocks, 2005). This would allow proper reasoning about value ranges for OWL datatype properties. However, this extension is currently not widely supported by standard tooling; it has yet to find its way into mainstream Semantic Web infrastructure. Since GoodRelations aims at making Semantic Web-based E-Commerce a reality on the basis of current technology components, using OWL-Eu was not an option for the moment.

Instead, we use a pragmatic workaround that shifts part of the work on the individual expressing the query but at the same time requires only a standard RDF-S or OWL reasoner that supports `rdfs:subPropertyOf`.

The approach is as follows (see Figure 3):

- a) We create an ontology class Quantitative Value.
- b) All properties reflecting quantitative characteristics of products or services are represented as object properties with the range of Quantitative Value.
- c) For each quantitative value, we create a new instance of Quantitative Value.

- d) We then attach the upper and lower limits of this value by two datatype properties (i.e. attributes) hasMinValue and hasMaxValue, and the unit of measurement by a datatype or object property hasUnitOfMeasurement.

Now, querying for suitable object requires just specifying the lower and upper limits for the respective characteristics. Figure 3 illustrates this approach.

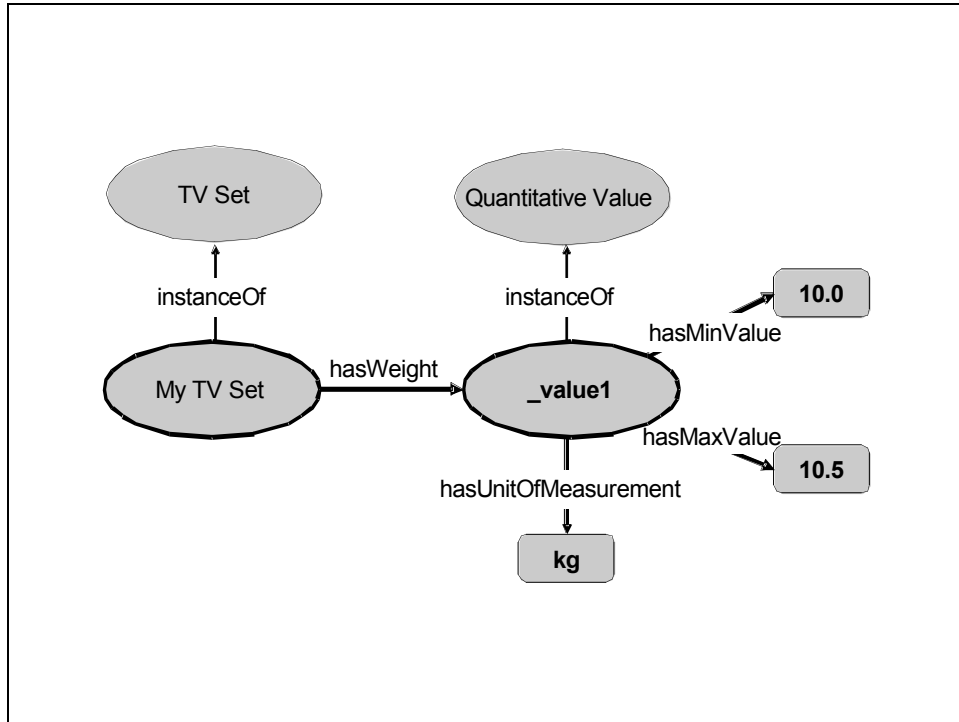


Figure 3. Work-around for Value Ranges

One may object that this means a lot of redundancy for those cases where the upper and the lower limit are practically the same. In order to mitigate this, we introduce a third datatype property hasValue, which is an `rdfs:subPropertyOf` both hasMinValue and of hasMaxValue.

We can then simply say

```

myTVSet hasWeight X
X instance of QuantitativeValue
X hasValue 10
X hasUnitOfMeasurement "kg"
  
```

As long as there is reasoner that computes the implicit fact that all

```
X hasValue Y
```

implies

```
X hasMinValue Y and
```

```
X hasMaxValue Y,
```

this shortcut will return the very same results.

Note: Users of this workaround must be advised that ranges in here mean that **all** possible values in this interval are covered. (Sometimes, the actual commitment may be less than that: we rent cars from 2 – 12 seats does often not really mean that they have cars with 2,3,4,...12 seats.). Someone renting two types of rowing boats, one that fits for 1 or 2 people, and another that must be operated by 4 people cannot claim to rent boats with a seating capacity between 1 and 4 people. He or she is renting out two boat types , one holding 1-2 and another holding 4 passengers.

In practice, we created two specializations of the classes and datatype properties, one for float values and one for integer values. Figure 4 shows the respective part in Protégé.

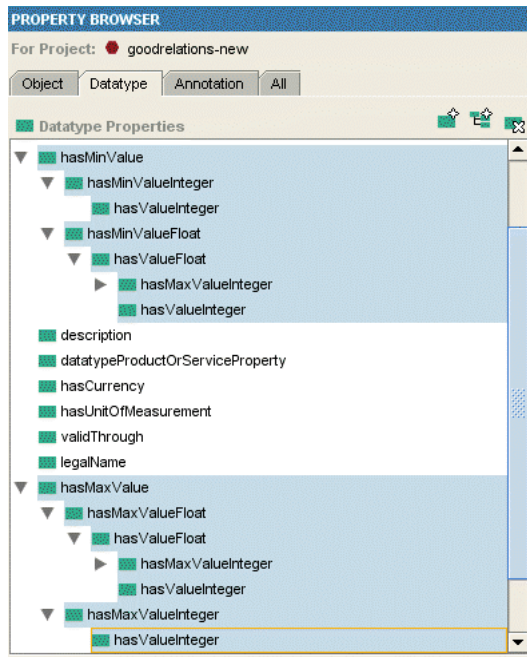


Figure 4. The work-around for ranges in Protégé

3.3.5 Incomplete Descriptions

Descriptions of product or services offerings are often incomplete, and it must thus be expected that machine-readable annotations of those offerings will also be incomplete. A common situation will be that (1) manufacturers of products publish data about their products in Web documents, (2) dealers provide additional information and prices, (3) third-parties provide additional data, information, knowledge or services (e.g. recommended products for a given purpose). That means that we will have to deal with statements stored in multiple places by different business entities, for individual purposes and driven by different incentives.

This distributed environment means that any operation on the data must be able to (1) properly handle incomplete information, (2) distinguish missing information from negative information, and (3) identify exact matches, potential matches, and partial matches (cf. Di Noia, Di Sciascio, Donini, & Mongiello, 2003, pp. 321-322) between a demand and a supply. For example, the absence of a characteristic in a description should be treated as a characteristic that could either be refined later or ignored (if irrelevant) (Di Noia, Di Sciascio, Donini, & Mongiello, 2003, p. 324). In addition to that, it must be guaranteed

that more specific descriptions are not inferior to unfairly generic or simply less specific descriptions (though the latter might contain some contradictions to the counterparty's description) (Di Noia, Di Sciascio, Donini, & Mongiello, 2003, pp. 324-325, 328).

Otherwise the strange situation would result that the less information you provide, the more often are you considered a potential match – ignoring the value of specificity for a potential business partner.

However, neither the Open World Assumption (OWA) nor Negation-as-Failure (NAF) is directly suitable for this, since the absence of a product property may require very different interpretations. As a heuristic, the absence of such a property that is perceived as *adding value* for the majority of potential customers can often be interpreted as the negation of this property. For instance, if a hotel does not state whether it has a sauna or a pool, then it likely does not have this feature. However, for characteristics that are not clearly just adding value, this heuristic will likely not work. For instance, the absence of “close to highway” may support very different interpretations depending on the context. For a leisure hotel, it may be that the absence of this property is just because the hotel owner does not want to reveal that the hotel is pretty close to a noisy highway, i.e. that it actually *is* close to highway. For a business hotel, it may be that the hotel is quite far away from a major road and thus difficult to reach, i.e. that it *is not* close to a highway. In the context of tourism offerings, we have termed this the “Swimming Pool Problem” (Hepp, Siorpaes, & Bachlechner, 2006). Such issues have also been discussed in the context of the W3C standardization activities of rule languages (cf. Hawke, Tabet, & Marie, 2005). Polleres, Feier, and Harth have recently proposed contextually-scoped negation as an innovative solution to the misfit of both OWA and NAF to many Semantic Web scenarios (Polleres, Feier, & Harth, 2006). Still, contexts do not seem to be sufficient if the most likely interpretation will depend on *the impact of an individual property on the perceived utility of an offering*, i.e. whether having this property makes the offering more attractive or less attractive.

In practice, GoodRelations leaves it up to the application how to handle incomplete information; the application must check and rank. We think that a first approach should be to penalize incomplete information, so that an offering that is incomplete will rank lower than a complete one.

3.4 Formalization: Ontology Coding in OWL DLP

In the following, we motivate our choice of the ontology language and describe the coding of the ontology in OWL DLP. The main difference between the domain capture described in section 3.2 and the OWL variant of the ontology is that we introduce new conceptual entities for n-ary relations, since OWL supports binary relations only. The full ontology coding and description of elements is in the Annexes B and C at the end of this report.

3.4.1 Appropriate Ontology Language: RDF-S, OWL DLP, or OWL DL?

Since the GoodRelations ontology is meant to be of practical value on the basis of currently available Semantic Web infrastructure, the choice of the appropriate ontology

language is important. Eventually, we suggest using the OWL DL syntax for RDF-S elements, i.e. a subset of the closure of OWL DLP, so that a lightweight RDF-S-style reasoner can compute all practically relevant inferences while the ontology could also be used together with OWL DL ontologies and knowledge bases without making the resulting model become OWL Full. We can safely do so, because the respective OWL language elements which we use are specializations of RDF-S elements (see appendix B of the OWL Web Ontology Language Reference, <http://www.w3.org/TR/owl-ref/>).

The GoodRelations ontology will only use the following language elements:

```
owl:Ontology
owl:Class
owl:ObjectProperty
owl:DatatypeProperty
rdfs:subClassOf
rdfs:subPropertyOf
rdfs:comment
rdf:datatype
rdf:type
```

It also includes `rdfs:range` and `rdfs:domain`; however, this is meant solely for helping tools generate forms for creating instance data. It is not necessary that the ontology repository computes the correct closure for the official RDF-S semantics. For a discussion of the semantics of domain and range in RDF-S and OWL, see (de Bruijn, Lara, Polleres, & Fensel, 2005).

In combination with these domain and range statements, we use very few complex class definitions that define the union of multiple classes. However, again, this is used solely for specifying domains for object and datatype properties so that tools can generate forms easily for populating the ontology. For example, the `owl:DatatypeProperty` "hasCurrencyValue" is applicable to unit prices and delivery and payment charges:

```
<rdfs:domain>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#PaymentChargeSpecification"/>
      <owl:Class rdf:about="#DeliveryChargeSpecification"/>
      <owl:Class rdf:about="#UnitPriceSpecification"/>
    </owl:unionOf>
  </owl:Class>
</rdfs:domain>
```

For the correct interpretation of the offering data on a Web scale and query answering, it is not necessary that the ontology infrastructure computes the correct closure of this.

In short, the advantages of our choice are as follows:

- a) All data annotated using GoodRelations on the Web can be properly interpreted with just an (incomplete; minimal) RDF-S-style reasoner, as long as this reasoner supports `owl:Class`, `owl:ObjectProperty`, and `owl:DatatypeProperty`. This is insofar important as SPARQL queries on everything beyond a subset of RDF-S are difficult and subject to ongoing research.

- b) At the same time, the ontology plus data in combination with OWL products and services ontologies stay within OWL DL and can be used with a DL (or even DLP) reasoner – i.e., we do not end up in OWL Full.
- c) Also, we could otherwise not define important top-level concepts for product and services ontologies in GoodRelations, since they need to be imported by future releases of eClassOWL and others (see section 4.1. for details). If GoodRelations was using RDF-S elements, importing the ontology in eclassOWL would turn eClassOWL into an OWL Full ontology.
- d) The language fragment we suggest is also, in practice, not touched by the layering problems between RDF-S and OWL DL (for a discussion of these issues, see e.g. (Eiter, Ianni, Polleres, Schindlauer, & Tompits, 2006)).

Our approach is in line with the proposal by Polleres et al. of using some restricted sets of RDF(S) inference with an extended version of SPARQL (Polleres, Scharffe, & Schindlauer, 2007). This would provide the foundation for reliable reasoning support for using (a future version of) SPARQL as a query language with GoodRelations-related data on a Web scale.

3.4.2 N-Ary Relations in OWL

We have seen in the domain capture in section 3.1 that there exist several ternary and quaternary relations. Unfortunately, OWL supports only binary relationships in the form of object properties. This means that we have to find modeling work-arounds for the higher-arity relations in the domain. The basic pattern for handling this is (1) introducing an additional class for each of those relationships, of which instances act as placeholders that keep together all but one of the parameters of the n-ary relation and (2) defining a binary relation that links the remaining parameter with this placeholder instance.

Example: There exists a quaternary relationship type “includesTypeOfGood” between (1) an Offering and (2) a Quantity, (3) a Unit of Measurement, and (4) a Product or Service Instance. This relation indicates the how much of a given product is included in a particular offering. Since there are only binary relations in OWL, we cannot specify this as

```
R.includesTypeOfGood {Offering, Quantity, Unit of Measurement,
Product or Service Instance}.
```

Instead, we create one `owl:Class` TypeAndQuantityNode and one `owl:ObjectProperty` includesObject with a domain of Offering and a range of TypeAndQuantityNode. We then create three additional `owl:DatatypeProperties` or `owl:ObjectProperties` for attaching the values for Quantity and Unit of Measurement, and linking to the Product or Service Instance.

Figure 5 illustrates this approach.

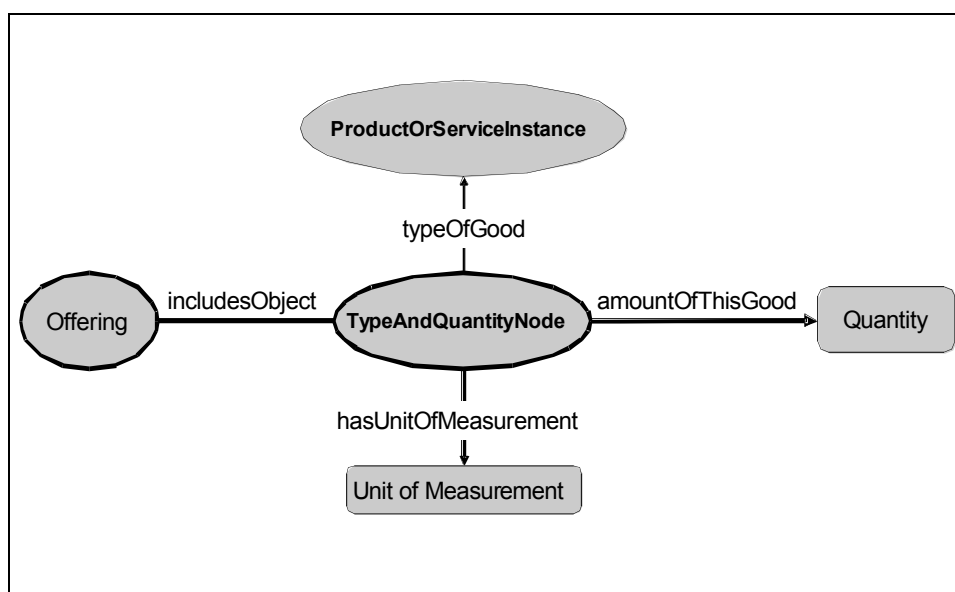


Figure 5. Modeling work-around for quaternary relations in OWL

In the OWL coding of GoodRelations, we have the following such placeholder classes for relations with a higher arity than two:

AcceptedPaymentMethods: This is a conceptual entity that holds together all aspects of the n-ary relation AcceptedPaymentMethods.

AvailableDeliveryMethods: This is a conceptual entity that holds together all aspects of the n-ary relation AvailableDeliveryMethods.

OpeningHoursSpecification: This is a conceptual entity that holds together all aspects of the n-ary relation OpeningHoursSpecification, which defines the opening hours for a given DayOfWeek for a given LocationOfSalesOrServiceProvisioning.

TypeAndQuantityNode: This is a conceptual entity that holds together all aspects of the quaternary relation includesTypeOfGood, namely the Quantity, the Unit of Measurement, the Product or Service, and the Offering to which this belongs.

Note: The link between Offering and TypeAndQuantityNode is represented by the object property includesObject. The Unit of Measurement is attached using the hasUnitOfMeasurement datatype property. The quantity is specified using the datatype property amountOfThisGood (float). The specification of the item included is represented by the object property typeOfGood.

Example: An offering may consists of 100g Butter and 1 kg of potatoes, or 1 cellphone and 2 headsets.

WarrantyPromise: This is a conceptual entity that holds together all aspects of the n-ary relation hasWarrantyPromise. A Warranty Promise is an entity representing the duration and scope of services that will be provided to a customer free of charge in case of a defect or malfunction of the Product or Service Instance. A Warranty Promise is characterized by its temporal duration (usually starting with the date of purchase) and its

Warranty Scope. The Warranty Scope represents the types of services provided (e.g. labor and parts, just parts) as part of the warranty included in an Offering. The actual services may be provided by the Business Entity making the Offering, by the manufacturer of the Product, or by a third party. There may be multiple Warranty Promises associated with a particular Offering, which differ in duration and scope (e.g. pick-up service during the first 12 months, just parts and labor for 36 months).

Examples: 12 months parts and labor, 36 months parts only

In order to group those placeholder classes and because they have the same ontological nature (they are abstract placeholders), we create a superclass "N-Ary-Relations", of which the five classes listed above are subclasses.

3.4.3 *Product Classes, Instances, and Models*

eCl@ss, eClassOWL, and many other taxonomies for products and services don't make the important but subtle distinction between instances and models of a type of product or service, as explained in section 3.1.5. In fact, eCl@ss and eClassOWL provide classes like "TV Set" and attributes like "hasScreenSize" that can, in the absence of a formal definition of their semantics, be used **both** for describing product models (e.g. the default screenSize of a particular Siemens TV set), **and** product instances (e.g. my actual TV set).

This is why eClassOWL products and services classes are all subclasses of a top-level class that is the union of Product or Service Model and Product or Service Classes. Thus, we introduce **four** classes for product or services classes, instances, and models:

ProductOrService as a top-level class, and ActualProductOrService, ProductOrServiceModel, and ProductOrServicesSomeInstancesPlaceholder as subclasses of this top-level class. They are defined as follows:

owl:Class ProductOrService

The superclass of all classes describing products or services types, either by nature or purpose. Examples for such subclasses are "TV set", "vacuum cleaner", etc. All eClassOWL "gen" classes are subclasses of this class. An instance of this class can be either an actual product or service or a placeholder instance for unknown instances of a mass-produces commodity. Since eClassOWL and other large products and services ontologies are used for both describing product and services instances and product and service makes and models, this top-level concept is the union of (1) Actual Product or Service Instances, (2) Product or Service Models, and (3) ProductOrServiceSome-Instances Placeholders. The latter are "dummy" instances representing anonymous products or services instances (i.e. such that are said to exist but not actually being exposed on the Web).

Examples: a) MyCellphone123, i.e. my personal, tangible cell phone b) Siemens1234, i.e. the Siemens cell phone make and model 123 c) dummyCellPhone123 as a placeholder for actual instances of a certain kind of cellphones.

owl:Class ActualProductOrServiceInstance

An Actual Product or Service Instance is a single identifiable object or action that creates some increase in utility (in the economic sense) for the individual possessing or using this very object (Product) or for the individual in whose favor this very action is being taken (Service). Products or Services are types of goods in the economic sense.

Examples: MyThinkpad T60, the pint of beer standing in front of me, my Volkswagen Golf, the haircut that I received or will be receiving at a given date and time. Note: In many cases, product or service instances are not explicitly exposed on the Web but only existentially quantified. For a detailed discussion and practical solutions, see section 3.3.3.

owl:Class ProductOrServiceModel

From the ontological perspective, a Product or Service Model is an intangible entity that specifies some characteristics of a group of similar, usually mass-produced Products. In case of mass-produces Products, there exists a relation hasMakeAndModel between the Products and Services Instance and the Product or Service Model. However, since eClassOWL and other products and services ontologies don't support this important distinction, Product or Service Models are a subclass of Product or Service in GoodRelations.

Examples: Ford T, Volkswagen Golf, Sony Ericsson W123 cellphone

owl:Class ProductOrServicesSomeInstancesPlaceholder

A placeholder instance for unknown instances of a mass-produces commodity. This is used as a computationally cheap workaround for such instances that are not individually exposed on the Web but just stated to exist (i.e., which are existentially quantified).

Example: An instance of this class can represent an anonymous set of green Siemens1234 phones. It is different from the ProductOrServiceModel Siemens1234, since this refers to the make and model, and it is different from a particular instance of this make and mode (e.g. my individual phone) since the latter can be sold only once. Siemens1234, i.e. the Siemens cell phone make and model 123 as a placeholder for all actual instances. Figure 6 shows the resulting subsumption hierarchy.

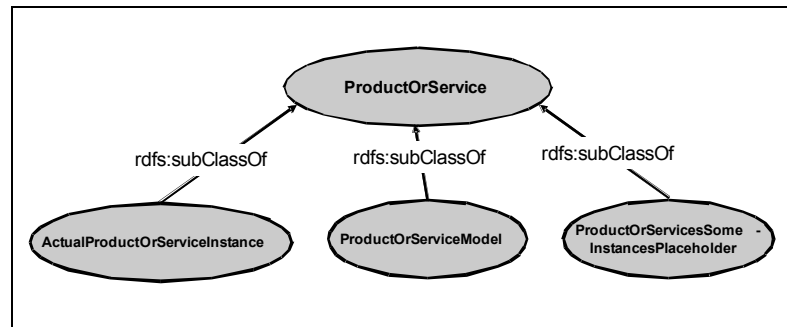


Figure 6. The GoodRelations ontology in OWL uses these four classes for products and services in order to maximize compatibility with eCl@ss and UNSPSC and derived ontologies

An key reason for this modeling workaround is the large amount of properties for product characteristics that are part of eCl@ss and eClassOWL, with more than 5,000 precisely defined elements for all kinds of aspects. Properly separating products from models, as described in the domain capture in section 3.1.5, and as would follow from OntoClean (Guarino & Welty, 2002, , 2004) would mean we have to duplicate both the hierarchy of categories of eCl@ss (now more than 30,000 categories) and the properties (more than 5,000), which is unfeasible. Quite clearly, we need to be able to specify e.g. screen sizes for both models and instances, but a model does not have a screen size in the same way an instance has a screen size. The semantics of screen size attached to a model implies that an actual product that is of the respective make and model will (likely) have the respective screen size (see section 3.1.5 for details).

When annotating an entity, we can then be more specific and say that the entity is either a model or a product or a placeholder for anonymous instances.

3.4.4 Top-level Ontology Part for Products and Services Ontologies

The GoodRelations ontology serves two purposes: It provides the vocabulary for representing the relationships between Web resources, business entities, offerings, and goods, e.g. products and services. Also, it should provide a top-level ontology for products and services ontologies so that those are well aligned with GoodRelations. For that, GoodRelations specifies the following elements, to which future releases of products and services ontologies should refer.

owl:Class ProductOrService

As defined above; this becomes the superclass of all products and services classes in eClassOWL and other products and services ontologies. In the case of eClassOWL, this will be the superclass of all generic classes, see (Hepp, 2006b) for details on generic and taxonomic classes in eClassOWL).

owl:Class QualitativeValue

A Qualitative Value is an entity that represents the state of a certain qualitative Product or Service Property. Qualitative Values are either Literal Values or Enumerative Values. Literal values are represented just as literals with respective datatype properties. For all other enumerative values, instances of this class are being created. An instance of this

class represents a qualitative value for an object property. This is the superclass of all enumerated values in eClassOWL.

owl:ObjectProperty qualitativeProductOrServiceProperty

This is the super property of all qualitative properties for products and services. All eclassOWL properties for which QualitativeValue instances are specified are subproperties of this property.

owl:ObjectProperty quantitativeProductOrServiceProperty

This is the super property of all quantitative properties for products and services. All eclassOWL properties that specify quantitative characteristics, for which an interval is at least theoretically an appropriate value, are specified are subproperties of this property.

owl:DatatypeProperty datatypeProductOrServiceProperty

This property is the super property for all pure datatype properties that can be used to describe a product and services instance, or via the instances placeholders, of a set of instances of mass-produces commodities, or product or services models. Only such eClassOWL properties that are no quantitative properties and that have no predefined QualitativeValue instances are subproperties of this property. In practice, this refers to a few integer properties for which the integer value represents qualitative aspects, for string datatypes (as long as no predefined values exist), and for boolean datatype properties.

Note: Mind that eCl@ss 5.1 sets the range of qualitative properties to „string“ even if enumerated values are defined for this property.

3.4.5 Reuse of External Standards

For specifying Currencies, Units of Measurements, and Countries or Regions, the GoodRelations ontology refers to well established international standards, in particular

- a) ISO 3166-1 (two-letter codes) and ISO 3166-2 (ISO, 2006) and (ISO, 1998) for Countries and Regions,
- b) The UN/CEFACT Code (United Nations Economic Commission for Europe (UN/CEFACT), 2006) for Units of Measurement, and
- c) ISO 4217 for Currencies (ISO, 2001).

When reusing such standards, there are two principle options: First, one can derive an ontology from the conceptual elements contained in the standard. For example, one could create an ontology of countries and administrative regions based on ISO 3166-1 and 3166-2. Second, one can define simple `owl:DatatypeProperties` with a range of string or integer, which will store the identifier of the element in the standard. For reasons of copyright, decoupling of evolution of ontologies, simplicity, and low data conversion barriers for existing corporate data assets, we chose the second option for GoodRelations. For a detailed argument, see (Hepp, 2007).

Accordingly, GoodRelations includes the following `owl:DatatypeProperties` that refer to external standards.

owl:DatatypeProperty eligibleRegions

This property specifies the geo-political region or regions for which the offer is valid using the two-character version of ISO 3166-1 (ISO 3166-1 alpha-2) for regions or ISO 3166-2, which breaks down the countries from ISO 3166-1 into administrative subdivisions.

Important: Do NOT use 3-letter ISO 3166-1 codes!

Domain: ([Offering] or [DeliveryChargeSpecification])

Range: [http://www.w3.org/2001/XMLSchema#string]

owl:DatatypeProperty hasCurrency

The currency for all prices in the PriceSpecification given using the ISO 4217 standard (3 characters).

Domain: [PriceSpecification]

Range: [http://www.w3.org/2001/XMLSchema#string]

owl:DatatypeProperty hasUnitOfMeasurement

The Unit of Measurement for a QuantitativeValue, a Unit Price Specification, or a TypeAndQuantityNode given using the UN/CEFACT Common Code (3 characters).

Domain: ([QuantitativeValue] or [UnitPriceSpecification] or [TypeAndQuantityNode])

Range: [http://www.w3.org/2001/XMLSchema#string]

4 PUTTING IT TO WORK: IMPLEMENTATION ISSUES

In this section, we discuss practical issues of publishing and using GoodRelations. We also present best practices for designing future products and services ontologies so that they are compatible with GoodRelations.

4.1 Necessary Modifications to eClassOWL

The current releases of eClassOWL (up to version 5.1.3) are not fully compatible with some modeling choices in GoodRelations. This is mainly because all quantitative `eCl@ss` properties are `owl:DatatypeProperties` in eClassOWL. The more flexible representation of quantitative properties in GoodRelations with support for multiple units of measurement and value ranges is not compatible with that.

While it would have been possible to force compatibility with eClassOWL, this would have restricted the functionality of GoodRelations. Since an update of the eClassOWL conceptual model will take place anyway (latest with the release 6.0 of `eCl@ss` in 2008), we chose to rather improve upcoming eClassOWL releases so that they support the more flexible GoodRelations approach.

In this section, we describe the necessary modification for eClassOWL. We use the following namespace prefixes:

eco: eClassOWL

gr: GoodRelations

1. Import the GoodRelations ontology by default.
2. Make all –gen classes in eClassOWL subclasses of `gr:ProductOrService`.
3. Make the value class `eco:Value` a subclass of `gr:QualitativeValue`, and by that all instances of the class `eco:Value` instances of `gr:QualitativeValue`.
4. Make all product properties in eCl@ssOWL subproperties of either
 - (a) `gr:qualitativeProductOrServiceProperty`,
 - (b) `gr:quantitativeProductOrServiceProperty`, or
 - (c) `gr:datatypeProductOrServiceProperty`
 following the external algorithm below.
5. Indicate the default eCl@ss unit of measurement in the description of all properties (maybe also in the label); this is important, since eCl@ss has one single recommended unit of measurement per each property.
6. Create a mapping document from old eClassOWL properties to UN/CEFACT UoM strings, so that users can convert old data or internal eCl@ss-compliant data more easily. (This is optional.)
7. Set the domain and range of all properties appropriately! (This was not the case in eClassOWL 5.1).
8. For those properties that have changed (old datatype properties in eclassOWL, new quantitative or qualitative object properties in eclassOWL), use a new identifier, unless the base URI for the new release will be changed anyway.
9. It may be helpful for users to describe algorithmically the conversion of old datatype data to new ones (easy with a mapping document).

In the following, we sketch an algorithm for mapping eCl@ss properties to either

- (a) `gr:qualitativeProductOrServiceProperty`,
- (b) `gr:quantitativeProductOrServiceProperty`, or
- (c) `gr:datatypeProductOrServiceProperty`:

For all properties in the eCl@ss library:

```
IF there exists at least one value recommendation(*) THEN make
this property a subproperty of
gr:qualitativeProductOrServiceProperty; its range be
gr:QualitativeValue
```

```
IF there exists no value recommendation THEN
```

SEBIS Technical Report

IF datatype == boolean ("V?") THEN make this property a subproperty of gr:datatypeProductOrServiceProperty; its range be boolean

IF datatype == NR2* or NE3* THEN make this property a subproperty of gr:quantitativeProductOrServiceProperty, the range be gr:QuantitativeValueFloat

IF datatype == NR1* THEN

decide whether the integer represents a qualitative or quantitative value

IF qualitative THEN make this property a subproperty of gr:datatypeProductOrServiceProperty, the range be integer

IF quantitative THEN make this property a subproperty of gr:quantitativeProductOrServiceProperty, the range be gr:QuantitativeValueInteger

ELSE (i.e. datatype == other) THEN make this property a subproperty of gr:datatypeProductOrServiceProperty, the range be string

decide whether the integer represents a qualitative or quantitative value

This must be manually done for the ca. 585 properties of type N1* in eCI@ss. Most of them are quantitative properties (all except maybe 10), so the task should be rather easy (maybe 2 hours of work).

(*) We may be able to exploit the attribute type field in eCI@ss for that (direct vs. indirect)

Also, the server configuration for eClassOWL must be checked to include "rdf" as appropriate content type in the rewrite rule, see (Miles, Baker, & Swick, 2006), section "Recipe 1".

4.2 Storage of GoodRelations Annotation Data

For storing the annotations data related to Web offerings, there exist mainly two alternative approaches.

- a) One large RDF file per each shop or Web site, available either at a dedicated URI and made visible via a link on the main page, or by content negotiation (HTTP get with content type RDF/XML to the main page).
- b) One small RDF module per each offering or retrievable Web resource.

Such issues are discussed in (Lewis, 2007; Miles, Baker, & Swick, 2006; Sauermann, Cyganiak, & Völkel, 2007), in particular (Sauermann, Cyganiak, & Völkel, 2007), and, most importantly in (Chris Bizer, Cyganiak, & Heath, 2007). (Chris Bizer, Cyganiak, & Heath, 2007) section 7.1 suggest that large RDF files should be avoided, and instead

small ones linked by triples that involve resources from both should be used. In a recent e-mail discussion, Andreas Harth held against that, stating that for crawlers feeding a repository, many chunks of RDF data create much more overhead.

The GoodRelations ontology is neutral to this debate, though we slightly favor the “one big RDF file per shop” approach.

4.3 Publication of the Vocabulary

All element URIs of the Ontology should be dereferencable, see also (Chris Bizer, Cyganiak, & Heath, 2007) section 4.2. When publishing the ontology, one should take into account (Sauermann, Cyganiak, & Völkel, 2007), (Berners-Lee, 1998), (Miles, Baker, & Swick, 2006), and (Chris Bizer, Cyganiak, & Heath, 2007).

In a nutshell, we are using hash URIs for the GoodRelations ontology for it is very moderate in size. We will be preparing a lightweight XSLT stylesheet or provide content negotiating for pointing the user of the ontology to a human-readable specification.

The base URI of GoodRelations Version 1.0 is

<http://www.heppnetz.de/ontologies/goodrelations/v1>

The class “BusinessEntity” will then have the URI

<http://www.heppnetz.de/ontologies/goodrelations/v1#BusinessEntity>

5 USE CASE AND EVALUATION

In this section, we demonstrate and evaluate the GoodRelations ontology in a use case.

5.1 Features

This section summarizes the key features of the GoodRelations ontology. It is meant as input to a future factsheet and for promoting GoodRelations.

- Lightweight
- Support for ranges and units of measurements
- Support for all common business functions, like sell, lease, dispose, repair, etc.
- Compatible with eclassOWL and unspscOWL
- Supports all ISO 4217 currencies
- Supports defining eligible regions
- Suits both for explicit instances, product models, and anonymous instances
- Supports common delivery and shipping methods
- Supports accepted payment methods
- Offerings can be constrained to certain eligible business entities
- A warranty promise, i.e., its duration and scope can be specified
- Different prices for different types of customers or for different quantities can be expressed

- Charges for certain payment or delivery options can be specified; the latter also individually per region.
- Support for product bundles, for all kinds of units of measurements (2 kg butter + 2 cellphones for € 99 would be no problem).
- Compatible with international standards: ISO 3166, ISO 4217, UN/CEFACT, eCl@ss, and UNSPSC
- Minimal requirements on reasoner support – any RDF-S-style reasoner, OWL DLP, DL, or ter Horst reasoner will work.

5.2 Examples of Annotations

In the following, we give examples of how the GoodRelations ontology is to be used in typical scenarios. As a future extension, we may additionally model the full set of examples from sections 2.1 and 2.2. Since eClassOWL is pretty large and uses non-intuitiv identifiers for its elements, which would make the examples hard to read, and since the modifications described in section 4.1 have not yet taken place, we first produce a toy products and services ontology that contains three product classes “Cellphone”, “Piano”, and “Battery”, and the quantitative properties “hasWeight” and “hasTalkTime”. The resulting ontology is shown below.

```

<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY ex "http://www.domain2.com#" >
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY gr "http://www.heppnetz.de/ontologies/goodrelations/v1#" > ]>
<rdf:RDF xmlns="http://www.heppnetz.de/ontologies/goodrelations/examples#"
  xml:base="http://www.heppnetz.de/ontologies/goodrelations/examples#"
  xmlns:gr="http://www.heppnetz.de/ontologies/goodrelations/v1#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:ex="http://www.domain2.com#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Ontology rdf:about="">
    <owl:imports
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1"/>
    </owl:Ontology>
    <owl:Class rdf:ID="Battery">
      <rdfs:subClassOf rdf:resource="&gr;ProductOrService"/>
    </owl:Class>
    <owl:Class rdf:ID="Cellphone">
      <rdfs:subClassOf rdf:resource="&gr;ProductOrService"/>
    </owl:Class>
    <owl:ObjectProperty rdf:ID="hasTalkTime">
      <rdfs:subPropertyOf
rdf:resource="&gr;quantitativeProductOrServiceProperty"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="hasWeight">
      <rdfs:subPropertyOf
rdf:resource="&gr;quantitativeProductOrServiceProperty"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="Piano">
      <rdfs:subClassOf rdf:resource="&gr;ProductOrService"/>
    </owl:Class>
  </rdf:RDF>

```

5.2.1 Step 1: Business Entity

We assume there exist three business entities, Sony AG as a manufacturer, Amazon as a mail order vendor, and Peter Miller as a small business that sells cellphones on the Web, mostly via eBay. All three have a company Web site at <http://www.sony.com>, <http://www.amazon.com>, <http://www.peter-millers-shop.com> respectively.

The resulting RDF/XML code is shown below.

```

<gr:BusinessEntity rdf:ID="Sony">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Sony AG</rdfs:comment>
  <gr:legalName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Sony AG</gr:legalName>
  <rdfs:seeAlso rdf:resource="http://www.sony.com"/>
</gr:BusinessEntity>

<gr:BusinessEntity rdf:ID="Amazon">
  <gr:legalName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Amazon Inc.</gr:legalName>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Amazon</rdfs:comment>
  <rdfs:seeAlso rdf:resource="http://www.amazon.com"/>
</gr:BusinessEntity>

<gr:BusinessEntity rdf:ID="PeterMiller">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Peter Miller</rdfs:comment>
  <rdfs:seeAlso rdf:resource="http://www.peter-millers-shop.com"/>
  <gr:legalName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Peter Miller's Shop</gr:legalName>
</gr:BusinessEntity>

```

5.2.2 Step 2: Products and Offerings

There is a Sony cellphone model s1234. It has a weight of 100g and runs for 120 hours.

The model is described on the Web page <http://www.sony.com/cellphones/s1234/>.

The resulting RDF/XML code is shown in the following textbox. Note that the respective UN/CEFACT Common Codes for the Units of Measurement are as follows:

Unit of Measurement	UN/CEFACT Common Code
Minute [unit of time]	MIN
Gram	GRM
Unit or piece	C62

We can see how the flexible modeling of quantitative properties and the lack of ternary relations in OWL increase the size of the code; however, we do not think this is a practical issue, since most data will be generated and consumed by machines rather than humans. We are also planning an on-line conversion service that helps users generate GoodRelations data.

Note that the cellphone model is an instance of both the class “cellphone” from the products and services ontology, and of the class ProductOrServiceModel from GoodRelations for the reasons explained in sections 3.4.3 and 3.4.4.


```

<Cellphone rdf:ID="SonyCellPhoneModel_s1234">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Sony cellphone model s1234</rdfs:comment>
  <rdfs:seeAlso rdf:resource="http://www.sony.com/cellphones/s1234/" />
  <rdf:type
  rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrServiceModel" />
  <hasTalkTime>
    <gr:QuantitativeValueInteger rdf:ID="QuantitativeValueInteger_9">
      <gr:hasUnitOfMeasurement
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >MIN</gr:hasUnitOfMeasurement>
      <gr:hasMaxValueInteger
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >120</gr:hasMaxValueInteger>
      <gr:hasMinValueInteger
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >120</gr:hasMinValueInteger>
      <rdfs:comment
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >The node representing a time duration of 120
      minutes.</rdfs:comment>
    </gr:QuantitativeValueInteger>
  </hasTalkTime>
  <hasWeight>
    <gr:QuantitativeValueFloat rdf:ID="QuantitativeValueFloat_10">
      <gr:hasUnitOfMeasurement
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >GRM</gr:hasUnitOfMeasurement>
      <gr:hasMaxValueFloat
      rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >100.0</gr:hasMaxValueFloat>
      <gr:hasMinValueFloat
      rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >100.0</gr:hasMinValueFloat>
      <rdfs:comment
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >The value node representing a weight of 100 grams</rdfs:comment>
    </gr:QuantitativeValueFloat>
  </hasWeight>
</Cellphone>

```

Now, we want to model some explicit or implicit offerings:

- a) Amazon sells bundles of this model plus two batteries via its shop. This is announced on the Web page <http://www.amazon.com/cellphones/> too. The general offer is valid from July 1 – December 31, 2007.
- b) Peter Miller sells a used instance of this model via eBay and offers to repair any Sony s1234 cellphone. This is announced on the Web page <http://www.ebay.com/auction1234/> and <http://www.peter-millers-shop.com/service/> respectively. Both of his offers are valid from December 1 – December 31, 2007.
- c) Sony produces that model and implicitly sells instances thereof. This is announced on the Web page <http://www.sony.com/cellphones/s1234/>, too. The general offer is valid from January 1 – December 31, 2007.

All times are given in UTC.

The Amazon offerings look as follows in RDF/XML:

```
<rdf:Description rdf:about="&p1;Amazon">
  <gr:offers rdf:resource="&p1;AmazonOfferingABundle"/>
</rdf:Description>
<rdf:Description rdf:about="&p1;PeterMiller">
  <gr:offers rdf:resource="&p1;MillersOfferToRepair"/>
  <gr:offers rdf:resource="&p1;MillersOfferingEbay"/>
</rdf:Description>
<rdf:Description rdf:about="&p1;Sony">
  <gr:offers rdf:resource="&p1;SonyOffering_s1234_phones"/>
</rdf:Description>
<gr:Offering rdf:ID="AmazonOfferingABundle">
  <gr:validThrough
rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-12-31T23:59:59Z</gr:validThrough>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Amazon is offering a bundle, composed of s1234 phones and two
batteries.</rdfs:comment>
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-01-01T00:00:00Z</gr:validFrom>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_Amazon1"/>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_Amazon2"/>
  <rdfs:seeAlso rdf:resource="http://www.amazon.com/cellphones/">
  <gr:hasBusinessFunction
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Sell"/>
  </gr:Offering>

<gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_Amazon1">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This node represents that the Amazon offering includes two
batteries.</rdfs:comment>
  <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >C62</gr:hasUnitOfMeasurement>
  <gr:typeOfGood>
  <gr:ProductOrServicesSomeInstancesPlaceholder
rdf:ID="CellPhoneBattery_InstancePlaceholder">
    <rdf:type rdf:resource="#Battery"/>
  </gr:ProductOrServicesSomeInstancesPlaceholder>
  </gr:typeOfGood>
  <gr:amountOfThisGood
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.0</gr:amountOfThisGood>
  </gr:TypeAndQuantityNode>

<gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_Amazon2">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This instance reflects that the Amazon offering includes 1 unit (Code
62) of the instances placeholder Cellphone_3.</rdfs:comment>
  <gr:amountOfThisGood
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</gr:amountOfThisGood>
  <gr:typeOfGood>
  <gr:ProductOrServicesSomeInstancesPlaceholder rdf:ID="Cellphone_3">
    <gr:hasMakeAndModel rdf:resource="#SonyCellPhoneModel_s1234"/>
    <rdf:type rdf:resource="#Cellphone"/>
  </gr:ProductOrServicesSomeInstancesPlaceholder>
  </gr:typeOfGood>
  <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >C62</gr:hasUnitOfMeasurement>
  </gr:TypeAndQuantityNode>
```

The Peter Miller offerings will look as shown on the next page: First, we show how his offering on eBay would be annotated. Then follows how his generic offer to try to repair any cellphone that is a Sony s1234 make and mode will be expressed.

```

<gr:Offering rdf:ID="MillersOfferingEbay">
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-12-01T00:00:00Z</gr:validFrom>
  <gr:hasBusinessFunction
  rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Sell"/>
  <gr:validThrough
  rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-12-31T23:59:59Z</gr:validThrough>
  <rdfs:seeAlso rdf:resource="http://www.ebay.com/auction1234/" />
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Peter Miller's Offering to sell his cellphone on eBay.</rdfs:comment>
  <gr:includesObject>
    <gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_PeterEbay">
      <gr:typeOfGood>
        <gr:ActualProductOrServiceInstance rdf:ID="PetersUsedCellphone">
          <gr:hasMakeAndModel rdf:resource="#SonyCellPhoneModel_s1234"/>
          <rdf:type rdf:resource="#Cellphone"/>
        </gr:ActualProductOrServiceInstance>
      </gr:typeOfGood>
      <gr:hasUnitOfMeasurement
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >C62</gr:hasUnitOfMeasurement>
      <gr:amountOfThisGood
      rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >1.0</gr:amountOfThisGood>
    </gr:TypeAndQuantityNode>
  </gr:includesObject>
</gr:Offering>

```

```

<gr:Offering rdf:ID="MillersOfferToRepair">
  <gr:hasBusinessFunction
  rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Repair"/>
  <rdfs:seeAlso rdf:resource="http://www.peter-millers-
  shop.com/service/" />
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-12-01T00:00:00Z</gr:validFrom>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Peter Miller's Offering to repair Sony s1234
  cellphones.</rdfs:comment>
  <gr:validThrough
  rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-12-31T23:59:59Z</gr:validThrough>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_MillersRepair"/>
</gr:Offering>
<gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_MillersRepair">
  <gr:amountOfThisGood
  rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</gr:amountOfThisGood>
  <gr:typeOfGood
  rdf:resource="#AnonymousCellphoneInstancesOfTypeSony_s1234"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The node that represents that Peter Miller's offer to repair Sony
  s1234 cellphones refers to 1 cellphone. This node may become significant in
  combination with Unit Price Specifications.</rdfs:comment>
  <gr:hasUnitOfMeasurement
  rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >C62</gr:hasUnitOfMeasurement>
</gr:TypeAndQuantityNode>
<Cellphone rdf:ID="AnonymousCellphoneInstancesOfTypeSony_s1234">
  <rdf:type
  rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrS
  ervicesSomeInstancesPlaceholder"/>
  <gr:hasMakeAndModel rdf:resource="#SonyCellPhoneModel_s1234"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This node represents all of the anonymous cellphone instances of Sony
  s1234 which Peter Miller is willing to try to repair.</rdfs:comment>
</Cellphone>

```

The implicit offer by Sony to sell their cellphones, too, will be expressed as follows:

```
<gr:Offering rdf:ID="SonyOffering_sl234_phones">
  <gr:validThrough
rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-12-31T23:59:59Z</gr:validThrough>
  <rdfs:seeAlso rdf:resource="http://www.sony.com/cellphones/sl234/">
  <gr:hasBusinessFunction
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Sell"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>The general Sony offer to sell sl234s</rdfs:comment>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_Sony"/>
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-01-01T00:00:00Z</gr:validFrom>
</gr:Offering>

<gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_Sony">
  <gr:typeOfGood>
    <Cellphone rdf:ID="Cellphone_15">
      <gr:hasMakeAndModel rdf:resource="#SonyCellPhoneModel_sl234"/>
      <rdf:type
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOr
ServicesSomeInstancesPlaceholder"/>
    </Cellphone>
  </gr:typeOfGood>
  <gr:amountOfThisGood
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>1.0</gr:amountOfThisGood>
  <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>C62</gr:hasUnitOfMeasurement>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>The node representing the fact that the general Sony offer refers to
one cellphone. C62 is the common code for "unit".</rdfs:comment>
</gr:TypeAndQuantityNode>
```

5.2.3 Step 3: Eligible Customers and Regions

The Sony offering is for resellers only.

Amazon ships this phone only to Austria, Germany, and Switzerland.

Peter Miller will sell to Austria and Italy only.

```
<rdf:Description rdf:about="&p1;AmazonOfferingABundle">
  <gr:eligibleRegions
rdf:datatype="&xsd:string">DE</gr:eligibleRegions>
  <gr:eligibleRegions
rdf:datatype="&xsd:string">AT</gr:eligibleRegions>
  <gr:eligibleRegions
rdf:datatype="&xsd:string">CH</gr:eligibleRegions>
</rdf:Description>
<rdf:Description rdf:about="&p1;MillersOfferingEbay">
  <gr:eligibleRegions
rdf:datatype="&xsd:string">AT</gr:eligibleRegions>
  <gr:eligibleRegions
rdf:datatype="&xsd:string">IT</gr:eligibleRegions>
</rdf:Description>
<rdf:Description rdf:about="&p1;SonyOffering_sl234_phones">
  <gr:eligibleCustomerTypes rdf:resource="&gr;Reseller"/>
</rdf:Description>
```

5.2.4 Step 4: Price Specification

The price for the cellphone at Amazon is 99 Euros per cellphone including VAT.

The validity of that price is from Dec 1 – Dec 31, 2007.

The price for the cellphone in Peter's fix price auction at eBay is 80 Euros including VAT.

The validity of that price is from Dec 1 – Dec 31, 2007.

Sony does not publish a price specification.

```
<rdf:Description rdf:about="&p1;AmazonOfferingABundle">
  <gr:hasPriceSpecification
rdf:resource="#UnitPriceSpecification_Amazon99"/>
</rdf:Description>
<rdf:Description rdf:about="&p1;MillersOfferingEbay">
  <gr:hasPriceSpecification
rdf:resource="#UnitPriceSpecification_MillersCellPhone"/>
</rdf:Description>
<gr:UnitPriceSpecification rdf:ID="UnitPriceSpecification_Amazon99">
  <rdfs:comment rdf:datatype="&xsd:string"
>The price specification that one unit of the bundle costs 99
Euros including VAT.</rdfs:comment>
  <gr:hasCurrencyValue
rdf:datatype="&xsd;float">99.0</gr:hasCurrencyValue>
  <gr:hasCurrency rdf:datatype="&xsd:string">EUR</gr:hasCurrency>
  <gr:validThrough rdf:datatype="&xsd;dateTime"
>2007-12-31T23:59:59Z</gr:validThrough>
  <gr:valueAddedTaxIncluded
rdf:datatype="&xsd;boolean">true</gr:valueAddedTaxIncluded>
  <gr:hasUnitOfMeasurement
rdf:datatype="&xsd:string">C62</gr:hasUnitOfMeasurement>
  <gr:validFrom rdf:datatype="&xsd;dateTime"
>2007-12-01T00:00:00Z</gr:validFrom>
</gr:UnitPriceSpecification>
<gr:UnitPriceSpecification
rdf:ID="UnitPriceSpecification_MillersCellPhone">
  <rdfs:comment rdf:datatype="&xsd:string"
>The price specification for Peter Miller's fix-price
auction.</rdfs:comment>
  <gr:hasCurrencyValue
rdf:datatype="&xsd;float">80.0</gr:hasCurrencyValue>
  <gr:hasCurrency rdf:datatype="&xsd:string">EUR</gr:hasCurrency>
  <gr:validThrough rdf:datatype="&xsd;dateTime"
>2007-12-31T23:59:59Z</gr:validThrough>
  <gr:valueAddedTaxIncluded
rdf:datatype="&xsd;boolean">true</gr:valueAddedTaxIncluded>
  <gr:hasUnitOfMeasurement
rdf:datatype="&xsd:string">C62</gr:hasUnitOfMeasurement>
  <gr:validFrom rdf:datatype="&xsd;dateTime"
>2007-12-01T00:00:00Z</gr:validFrom>
</gr:UnitPriceSpecification>
```

5.2.5 Step 5: Delivery and Delivery Charge Specification

Amazon ships via DHL or Mail.

The shipment charge via DHL is 8 Euros to Austria, Germany, and Switzerland.

The other offerings have no information of delivery modes and respective charges.

The respective RDF/XML code is shown on the next page.

```

<rdf:Description rdf:about="#p1;AmazonOfferingABundle">
  <gr:hasPriceSpecification rdf:resource="#DeliveryChargeSpecification_3-
Amazon-8EUR"/>
  <gr:availableDeliveryMethods rdf:resource="#gr;DeliveryModeMail"/>
  <gr:availableDeliveryMethods rdf:resource="#gr;DHL"/>
</rdf:Description>

<gr:DeliveryChargeSpecification rdf:ID="DeliveryChargeSpecification_3-
Amazon-8EUR">
  <rdfs:comment rdf:datatype="xsd:string"
    >The specification of shipment charges for the Amazon
offering.</rdfs:comment>
  <gr:hasCurrencyValue
rdf:datatype="xsd:float">8.0</gr:hasCurrencyValue>
  <gr:hasCurrency rdf:datatype="xsd:string">EUR</gr:hasCurrency>
  <gr:eligibleRegions rdf:datatype="xsd:string">AT</gr:eligibleRegions>
  <gr:eligibleRegions rdf:datatype="xsd:string">DE</gr:eligibleRegions>
  <gr:eligibleRegions rdf:datatype="xsd:string">CH</gr:eligibleRegions>
  <gr:valueAddedTaxIncluded
rdf:datatype="xsd:boolean">true</gr:valueAddedTaxIncluded>
  <gr:appliesToDeliveryMethod rdf:resource="#gr;DHL"/>
</gr:DeliveryChargeSpecification>

```

5.2.6 Step 6: Payment Options and Payment Charge Specification

Amazon accepts Visa and Mastercard.

Peter Miller accepts payment by bank transfer in advance only.

The RDF/XML code for this extension is included in the full sample data in Annex D only.

5.2.7 Step 7: Warranty Promise

Sony grants a warranty of 24 months parts and labor, customer bring-in.

Amazon grants a warranty of 12 months parts and labor, pick-up, and 36 months covering labor only.

Peter Miller grants a warranty on parts and labor for 1 month, customer bring-in.

The RDF/XML code for this extension is included in the full sample data in Annex D only.

5.2.8 Step 8: Bundles

We can also specify that a certain offering refers to a bundle of objects in arbitrary units of measurement, e.g. 500 grams of butter, 1 m of wire, and one piece of cellphone.

The sample code for this scenario is not included in the current version of the Technical Report.

5.2.9 Step 9: Services and Ranges

Peter Miller promises to repair cellphones that weigh between 10 and 120 grams.

The RDF/XML code for this extension is included in the full sample data in Annex D only.

5.2.10 Step 10: Consumables, Accessories, Spare Parts, and Similar Products

Peter Miller also sells one battery, which is an accessory and a spare part for the Sony cellphone model s1234.

The sample code for this scenario is not included in the current version of the Technical Report.

5.2.11 Step 11: Shop Locations and Opening Hours

Peter Miller's shop, from which he provides the service described in step 9, is located in Boston and opens Mondays through Saturdays, 10:00 a.m. – 8:00 p.m.

The sample code for this scenario is not included in the current version of the Technical Report.

5.3 Competency Questions in SPARQL

In the following, we give examples in SPARQL on how to query respective product and services data on the Semantic Web. This section may be extended in a future version of the report. In particular, we may model the full set of competency questions in SPARQL.

5.3.1 Find all known Business Entities and their legal names

```
PREFIX gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#>
SELECT ?entity ?legalname
WHERE { ?entity rdf:type gr:BusinessEntity.
?entity gr:legalName ?legalname.}
```

5.3.2 Who sells cellphones and on which Web pages can I get more information on respective offerings?

```
PREFIX gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#>
PREFIX ex: <http://www.heppnetz.de/ontologies/goodrelations/examples#>
SELECT ?business ?uri
WHERE {
?business gr:offers ?offering .
?offering gr:includesObject ?TypeAndQuantityNode .
?TypeAndQuantityNode gr:typeOfGood ?something .
?something rdf:type ex:Cellphone .
?offering gr:hasBusinessFunction gr:Sell.
?offering rdfs:seeAlso ?uri
}
```

Returns:

Query		Results									
<pre>PREFIX gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#> PREFIX ex: <http://www.heppnetz.de/ontologies/goodrelations/examples#> SELECT ?business ?uri WHERE { ?business gr:offers ?offering . ?offering gr:includesObject ?TypeAndQuantityNode . ?TypeAndQuantityNode gr:typeOfGood ?something . ?something rdf:type ex:Cellphone . ?offering gr:hasBusinessFunction gr:Sell. ?offering rdfs:seeAlso ?uri }</pre>		<table><thead><tr><th>business</th><th>uri</th></tr></thead><tbody><tr><td>p1:Sony</td><td>http://www.sony.com/cellphones/s1234/</td></tr><tr><td>p1:PeterMiller</td><td>http://www.ebay.com/auction1234/</td></tr><tr><td>p1:Amazon</td><td>http://www.amazon.com/cellphones/</td></tr></tbody></table>		business	uri	p1:Sony	http://www.sony.com/cellphones/s1234/	p1:PeterMiller	http://www.ebay.com/auction1234/	p1:Amazon	http://www.amazon.com/cellphones/
business	uri										
p1:Sony	http://www.sony.com/cellphones/s1234/										
p1:PeterMiller	http://www.ebay.com/auction1234/										
p1:Amazon	http://www.amazon.com/cellphones/										
<div>Execute Query</div>											

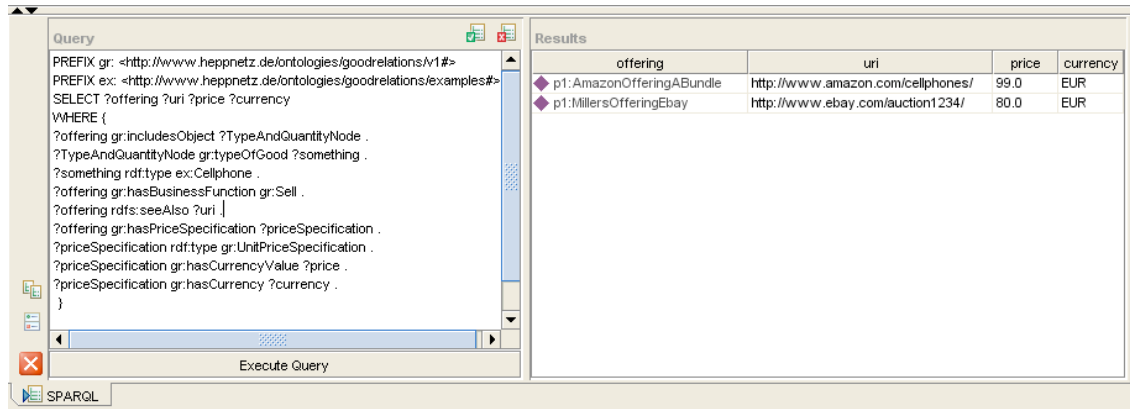
5.3.3 Which offers of cellphones exists, what is the price, and where can I find the offering on the Web?

```

PREFIX gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#>
PREFIX ex:
<http://www.heppnetz.de/ontologies/goodrelations/examples#>
SELECT ?offering ?uri ?price ?currency
WHERE {
  ?offering gr:includesObject ?TypeAndQuantityNode .
  ?TypeAndQuantityNode gr:typeOfGood ?something .
  ?something rdf:type ex:Cellphone .
  ?offering gr:hasBusinessFunction gr:Sell .
  ?offering rdfs:seeAlso ?uri .
  ?offering gr:hasPriceSpecification ?priceSpecification .
  ?priceSpecification rdf:type gr:UnitPriceSpecification .
  ?priceSpecification gr:hasCurrencyValue ?price .
  ?priceSpecification gr:hasCurrency ?currency .
}

```

The result is as expected (note that offerings that do not have a price specification do not appear in here due to the structure of the query):



The screenshot shows a SPARQL query interface. The 'Query' tab on the left contains the same SPARQL query as in the previous block. The 'Results' tab on the right displays the query results in a table format.

offering	uri	price	currency
p1:AmazonOfferingABundle	http://www.amazon.com/cellphones/	99.0	EUR
p1:MillersOfferingEbay	http://www.ebay.com/auction1234/	80.0	EUR

5.3.4 Who repairs at least one type of cellphone?

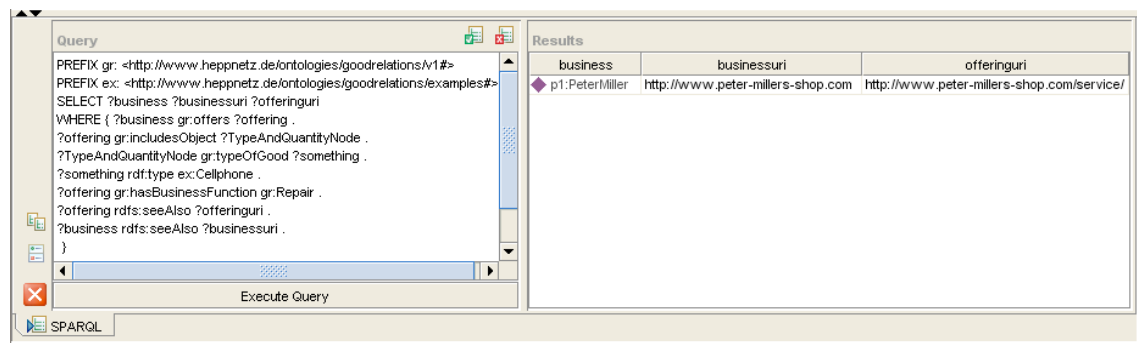
```

PREFIX gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#>
PREFIX ex:
<http://www.heppnetz.de/ontologies/goodrelations/examples#>
SELECT ?business ?businessuri ?offeringuri
WHERE {
  ?business gr:offers ?offering .
  ?offering gr:includesObject ?TypeAndQuantityNode .
  ?TypeAndQuantityNode gr:typeOfGood ?something .
  ?something rdf:type ex:Cellphone .
  ?offering gr:hasBusinessFunction gr:Repair .
  ?offering rdfs:seeAlso ?offeringuri .
  ?business rdfs:seeAlso ?businessuri .
}

```

Note that this query does not guarantee that this shop will repair any make and model. We only know that it has offered to repair some objects of which is known that they are cellphones.

Again, the result is as expected: We are pointed to Peter Miller's Website and the URI describing his offer to repair cellphones.



5.4 Results

While a large-scale evaluation of the GoodRelations ontology is just starting, we can see that the ontology can be used for many typical usage scenarios. In particular, it is both generic and flexible, while at the same time requiring only minimal reasoning support.

6 DISCUSSION

In the following, we briefly discuss several key issues of the GoodRelations ontology.

6.1 Need for a Fully-fledged Model

One could argue that the ontology is overly complex and very detailed. At least when first presenting it to domain experts without a Semantic Web background, some considered the approach rather complex. However, we think that the current balance between a very generic model and the ontology size is very balanced. In total, the ontology defines

- 29 classes,
- 21 object properties,
- 22 datatype properties, and
- 33 ontologically significant individuals

Also, at least a third of the ontology elements are caused by limitations of OWL, namely the lack of support for higher arity relations.

We want to stress that GoodRelations is meant as a global standard for expressing the key aspects of commodity offerings on the Web. Our first experiences with using the ontology are very encouraging.

While we had initially considered deriving a “light” subset which includes only the most important elements, we do now think that this would rather hamper the adoption and diffusion of GoodRelations.

It must also be said that a large deal of the complexity of creating proper annotations can be easily handled by a script or tool. In fact, we are already planning a GoodRelations annotation tool that simplifies the creation of GoodRelations annotations.

6.2 Related Work

Apart from the work described in the introduction, there is to our knowledge no serious proposal on how to encode commodity products and service offerings on the Semantic Web. In the context of Semantic Web services, some related work has been carried out. The work done in Korea (H. Lee & Shim, 2007; T. Lee, Chun, Shim, & Lee, 2006; T. Lee et al., 2006) is very impressive but in our opinion rather focussed on B2B scenarios in closed settings. In particular, it relies on rather heavyweight formalisms. Also, the focus of their work is different. In particular, they are not addressing typical B2C non-functional properties of offerings.

Remotely related is also work on describing Web Services semantically; the major difference is that we are focussing on the commercial and functional properties of actual goods and actual services, while Web Services and Semantic Web services research targets the discovery and invocation of computational functionality.

Eventually, we will try to align our work with ongoing efforts for lightweight Web Services annotation frameworks.

6.3 Related Ontologies

The two most relevant existing ontologies for our work are Proton (SEKT Consortium, 2005) and eClassOWL (Hepp, 2006a, 2006b). As for Proton, we will consider grounding GoodRelations in Proton in the future. As for eClassOWL, a full alignment with GoodRelations is scheduled for the release of version 6.0 later in 2008.

6.4 Trustworthiness of Annotations

An open issue for e-commerce on the Semantic Web is the trustworthiness of annotations. This has first been raised by Tolksdorf et al. (Tolksdorf, Bizer, Eckstein, & Heese, 2003). GoodRelations does not solve this problem. However, there are pragmatic approaches when implementing GoodRelations-based applications. First, one can take the reputation of the source of annotations into account. Despite numerous malicious attacks on corporate servers, it can be assumed that RDF/XML data fetched from resources within the domain space of known businesses is usually endorsed by the domain owner. Second, one may use digital signatures on RDF/XML documents, and recommender systems or product search engines can accept only RDF/XML triples from trusted origins.

Third, GoodRelations mainly aims at improving the accuracy of search for Web offerings. In the end, the purchasing transaction will be carried out by a human; often even offline. Thus, human intelligence for assessing the trustworthiness of the offerings in the result set can be used, too.

Again, our main point is that the issue of trustworthiness is orthogonal to the GoodRelations ontology.

6.5 Do Corporations Have an Incentive for Adopting GoodRelations?

Also in early days of e-commerce, there was a lot of debate on whether vendors would support e-commerce, for it fuels price comparison and competition. Such concerns are also discussed in (Tolksdorf, Bizer, Eckstein, & Heese, 2003).

We are confident that Semantic Web-based e-commerce will be adopted by businesses rather quickly, for two reasons: First, one cannot stop comparison shopping on the Web by not taking part. A few strong competitors on the Web are usually sufficient for a critical mass of offerings on the Web. Thus, just not annotating one's own offerings means simply they are not visible and not considered in purchasing decisions.

Second, the dominant approach for keeping up profits in competitive markets for cooperations is specialization and the individualization of products. For example, we do not escape fierce competition on markets by obscuring our terms and conditions, but rather by market segmentation, i.e. creating specific offerings for a particular audience in order to exploit their higher willingness to pay.

In a nutshell: Improving the accuracy of search for suppliers of goods on the Web on the basis of Semantic Web technology is in the interest of both consumers and vendors, for it reduces the costs of using the market mechanism for coordination, and it takes away obstacles towards further specialization and individualization of products.

6.6 Future Directions and Extensions

The following extensions and modifications of GoodRelations will be evaluated in the near future:

6.6.1 Microformat Variant and GRDDL Transformation

It may be desirable to create a microformat representation and GRDDL transformation scripts for GoodRelations. However, that requires some degree of adoption of GoodRelations first.

6.6.2 Proton Grounding

Proton (SEKT Consortium, 2005) is a popular general-purpose ontology and has some overlap with GoodRelations. We are evaluating the advantages and costs of grounding GoodRelations in Proton.

6.6.3 W3C Member Submission

We have plans to submit GoodRelations as a W3C Member Submission in 2008.

6.6.4 More Advanced Price Modeling

Several markets use more complex pricing schemas than those currently supported; see e.g. (Kelkar, Leukel, & Schmitz, 2002), BMEcat 1.2 (Schmitz, Kelkar, Pastoors, Renner, & Hümpel, 2001), and BMEcat 2005 (Schmitz, Leukel, & Kelkar, 2005). Since GoodRelations can be easily imported and extended, we currently favor the external development of more sophisticated subclasses of UnitPriceSpecifications. One may e.g. think about configurable products and bundles, and the respective impact on pricing; see e.g. the BMEcat 2005 documents, in particular (Schmitz, Leukel, & Kelkar, 2005). Many

scenarios can be handled with the current version of GoodRelations, but they may either cause a lack of details or a lot of redundancy. We will monitor the need for such extensions.

6.6.5 Resource Composition and Substitution

A more sophisticated vocabulary for specifying resource composition and substitution knowledge would be often beneficial in the context of Semantic Web-based e-commerce. However, we think that this should rather be provided by a separate ontology.

6.6.6 Multi-dependent Properties

Some properties of product instances or models are dependent on others. Currently, GoodRelations does not support modeling such properties.

6.6.7 Integration with Catalog Data Standards for Harvesting Product Data

For catalog data exchange on the basis of persistent publication on the Web, it may be beneficial to provide a Semantic Web vocabulary based on popular XML standards for B2B catalog data exchange, e.g. BMEcat 1.2 (Schmitz, Kelkar, Pastoors, Renner, & Hümpele, 2001) and BMEcat 2005 (Schmitz, Leukel, & Kelkar, 2005). We think that this should rather be ontologies in their own right, albeit aligned with GoodRelations.

An alternative approach is using those standards for simplifying the generation of GoodRelations data from existing catalogs. We are already working on the latter approach.

6.6.8 WSMML Variant

A variant of this ontology in the WSMML family of languages is under consideration.

6.6.9 URNs for UPC/EAN

There is a draft for using URNs for identifying products on the Web:

<http://www.ietf.org/internet-drafts/draft-mealling-epc-urn-02.txt> .

We will evaluate whether this can or should be integrated with GoodRelations. In general, it is desirable to add properties for popular coding schemas, like UPC, ISBN, etc.

6.6.10 Axiomatization

Currently, the full semantics of the relationship between product models and products instances being of a particular model (as specified in the domain capture) are not part of the ontology coding. So attributes of their product models are not automatically taken as default for products. We are evaluating ways of modeling this without leaving our minimal ontology language fragment.

In addition, the spare part and consumable relations would gain from an axiom that says that products that are of a certain make and model are spare parts or consumables for another make and model if that holds for their make and model.

7 CONCLUSION

We have developed the representational requirements for an ontology that can be used for describing offerings of tangible goods and commodity services on the Web. Our ontology is very flexible, while moderate in size. It poses minimal requirements on the reasoning support of the ontology management infrastructure and should thus scale well on current Semantic Web technology. Also, it should be compatible with some pragmatic reasoning support for SPARQL.

8 ACKNOWLEDGEMENTS

Acknowledgements: The author would like to thank Axel Polleres, Jos de Bruijn, Doug Foxvog, Katharina Siorpaes, Jacek Kopecky, Markus Linder, and Martin Schliefnig for numerous discussions and their invaluable feedback throughout three years of work in progress.

The work presented in this paper has been supported by the Austrian BMVIT/FFG under the FIT-IT Semantic Systems project myOntology (grant no. 812515/9284), by a Young Researcher's Grant (Nachwuchsförderung 2005-2006) from the Leopold-Franzens-Universität Innsbruck, and by the European Commission under the project SUPER (FP6-026850).

9 REFERENCES

- Beneventano, D., Guerra, F., Magnani, S., & Vincini, M. (2004). A Web Service based framework for the semantic mapping amongst product classification. *Journal of Electronic Commerce Research*, 5(2), 114-127.
- Berners-Lee, T. (1998). Cool URIs don't change. Retrieved Nov 8, 2004, from <http://www.w3.org/Provider/Style/URI.html>
- Bizer, C., Cyganiak, R., & Heath, T. (2007). How to Publish Linked Data on the Web. Retrieved July 25, 2007, from <http://sites.wiwiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/>
- Bizer, C., & Wolk, J. (2003). RDF Version of the eClass 4.1 Product Classification Schema. Retrieved August 16, 2005, from <http://www.wiwiwiss.fu-berlin.de/suhl/bizer/ecommerce/eClass-4.1.rdf>
- Booth, D. (2003). Four Uses of a URL: Name, Concept, Web Location and Document Instance. Retrieved January 15, 2007, from http://www.w3.org/2002/11/dbooth-names/dbooth-names_clean.htm
- Corcho, O., & Gómez-Pérez, A. (2001, August 5, 2001). *Solving Integration Problems of E-commerce Standards and Initiatives through Ontological Mappings*. Proceedings of the Workshop on E-Business and Intelligent Web at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001), Seattle, USA, pp. 1-10.
- de Bruijn, J., Lara, R., Polleres, A., & Fensel, D. (2005, May 10-14). *OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web*. Proceedings of the 14th International World Wide Web Conference (WWW2005), Chiba, Japan, pp. 623-632.
- Di Noia, T., Di Sciascio, E., Donini, F. M., & Mongiello, M. (2003, May 20-24, 2003). *A System for Principled Matchmaking in an Electronic Marketplace*. Proceedings of the Twelfth International World Wide Web Conference (WWW2003), Budapest, Hungary, pp. 321-330.
- Eiter, T., Ianni, G., Polleres, A., Schindlauer, R., & Tompits, H. (2006, September 4-8). *Reasoning with Rules and Ontologies*. Proceedings of the Reasoning Web 2006 Summer School, Lisbon, Portugal, pp. 93-127.
- Fasli, M. (2006). Shopbots: A Syntactic Present, A Semantic Future. *IEEE Internet Computing*, 10(6), 69-75.
- Fensel, D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M., et al. (2001). Product Data Integration in B2B E-Commerce. *IEEE Intelligent Systems*, 16(4), 54-59.
- Guarino, N., & Welty, C. A. (2002). Evaluating Ontological Decisions with OntoClean. *Communications of the ACM*, 45(2), 61-65.
- Guarino, N., & Welty, C. A. (2004). An Overview of OntoClean. In S. Staab & R. Studer (Eds.), *The Handbook on Ontologies* (pp. 151-172). Berlin: Springer.

- Gupta, T., & Qasem, A. (2002, May 7). *Reduction of price dispersion through Semantic E-commerce: A Position Paper*. Proceedings of the Semantic Web Workshop 2002, Hawaii, USA, pp. 1-2.
- Hawke, S., Tabet, S., & Marie, C. d. S. (2005). Rule Language Standardization. Report from the W3C Workshop on Rule Languages for Interoperability. Retrieved November 16, 2006, from <http://www.w3.org/2004/12/rules-ws/report/>
- Hepp, M. (2006a). eCI@ssOWL. The Products and Services Ontology. Retrieved May 20, 2006, from <http://www.heppnetz.de/eclassowl/>
- Hepp, M. (2006b). Products and Services Ontologies: A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards. *Int'l Journal on Semantic Web and Information Systems (IJSWIS)*, 2(1 (January-March)), 72-99.
- Hepp, M. (2007, April 25-27). *ProdLight: A Lightweight Ontology for Product Description Based on Datatype Properties*. Proceedings of the 10th International Conference on Business Information Systems (BIS 2007), Poznan, Poland, pp. 260-272.
- Hepp, M., Siorpaes, K., & Bachlechner, D. (2006, June 12-14). *Towards the Semantic Web in E-Tourism: Can Annotation Do the Trick?* Proceedings of the 14th European Conference on Information System (ECIS 2006), Gothenburg, Sweden, pp. 1-12.
- ISO. (1998). ISO 3166-2:1998 Codes for the representation of names of countries and their subdivisions - Part 2: Country subdivision code: ISO.
- ISO. (2001). ISO 4217:2001: Codes for the representation of currencies and funds.
- ISO. (2006). ISO 3166-1:2006 Codes for the representation of names of countries and their subdivisions - Part 1: Country codes ISO.
- Kelkar, O., Leukel, J., & Schmitz, V. (2002, May 7-11). *Price Modeling in Standards for Electronic Product Catalogs Based on XML*. Proceedings of the 11th International World Wide Web Conference (WWW2002), Honolulu, Hawaii, USA, pp. 366-375.
- Klein, M. (2002). DAML+OIL and RDF Schema representation of UNSPSC. Retrieved April 23, 2004, from <http://www.cs.vu.nl/~mcaklein/unspsc/>
- Lee, H., & Shim, J. (2007, July 11-13). *Conceptual Modeling of Product Information in e-Commerce*. Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS2007), Melbourne, Australia, pp. 937-942.
- Lee, T., Chun, J., Shim, J., & Lee, S.-g. (2006). An Ontology-Based Product Recommender System for B2B Marketplaces. *International Journal of Electronic Commerce*, 11(2), 125-154.
- Lee, T., Lee, I.-h., Lee, S., Lee, S.-g., Kim, D., Chun, J., et al. (2006). Building an operational product ontology system. *Electronic Commerce Research and Applications*, 5(1), 16-28.
- Lewis, R. (2007). Dereferencing HTTP URIs. Draft Tag Finding 31 May 2007. Retrieved July 25, 2007, from <http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14.html>
- McGuinness, D. L. (2001). UNSPSC Ontology in DAML+OIL. Retrieved November 5, 2004, from <http://www.ksl.stanford.edu/projects/DAML/UNSPSC.daml>
- Miles, A., Baker, T., & Swick, R. (2006). Best Practice Recipes for Publishing RDF Vocabularies. *W3C Working Draft 14 March 2006* Retrieved July 25, 2007, from <http://www.w3.org/TR/swbp-vocab-pub/>
- Milgate, M. (1987). Goods and Commodities. In J. Eatwell, M. Milgate & P. Newman (Eds.), *The New Palgrave: A Dictionary of Economics* (Vol. 2, pp. 546-548). London and New York: Macmillan and Stockton.
- Obrst, L., Wray, R. E., & Liu, H. (2001, October 17-19). *Ontological Engineering for B2B E-Commerce*. Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'01), Ogunquit, Maine, USA, pp. 117-126.
- Pan, J. Z., & Horrocks, I. (2005, May 29 - June 1). *OWL-Eu: Adding Customised Datatypes into OWL*. Proceedings of the Second European Semantic Web Conference (ESWC 2005), Heraklion, Crete, Greece, pp. 153-166.
- Polleres, A., Feier, C., & Harth, A. (2006). *Rules with Contextually Scoped Negation*. Proceedings of the 3rd European Semantic Web Conference (ESWC2006), Budva, Montenegro, pp. 332-347.
- Polleres, A., Scharffe, F., & Schindlauer, R. (2007, November 27-29). *SPARQL++ for Mapping between RDF Vocabularies*. Proceedings of the 6th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2007), OTM 2007, Part I, Vilamoura, Algarve, Portugal, pp. 878-896.
- Sauermann, L., Cyganiak, R., & Völkel, M. (2007). *Cool URIs for the Semantic Web* (No. TM-07-01): German Research Center for Artificial Intelligence.

- Schmitz, V., Kelkar, O., Pastoors, T., Renner, T., & Hümpel, C. (2001). *Specification BMEcat Version 1.2*. Stuttgart/Essen.
- Schmitz, V., Leukel, J., & Kelkar, O. (2005). *Specification BMEcat 2005*. Stuttgart/Essen.
- SEKT Consortium. (2005). PROTON Ontology. Retrieved January 15, 2007, from <http://proton.semanticweb.org/>
- Tolksdorf, R., Bizer, C., Eckstein, R., & Heese, R. (2003, November 3-7). *Business to Consumer Markets on the Semantic Web*. Proceedings of the On The Move to Meaningful Internet Systems OTM 2003 Workshops, Catania, Sicily, Italy, pp. 816-828.
- United Nations Development Programme. (2003). Business Function Identifiers (BFI). Retrieved 19.04.2003, 2003, from www.un-spssc.org/AdminFolder/documents/BFI.doc
- United Nations Development Programme. (2007). United Nations Standard Products and Services Code (UNSPSC). Retrieved May 22, 2007, from <http://www.unspssc.org/>
- United Nations Economic Commission for Europe (UN/CEFACT). (2006). Recommendation No. 20: Codes for Units of Measure Used in International Trade (CEFACT/ICG/2006/IC001 ed.): UN/CEFACT Information Content Management Group.
- Uschold, M., & Grüninger, M. (1996). Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 11(2), 93-155.
- Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998). The Enterprise Ontology. *The Knowledge Engineering Review*, 13(1), 31-89.
- W3C. (2003). XPointer Framework. W3C Recommendation 25 March 2003. Retrieved Dec 10, 2007, from <http://www.w3.org/TR/xptr-framework/>
- W3C. (2004). RDF Primer. W3C Recommendation 10 February 2004. Retrieved December 10, 2007, from <http://www.w3.org/TR/rdf-primer/>
- Zhao, Y. (2003, March 12). *Develop the Ontology for Internet Commerce by Reusing Existing Standards*. Proceedings of the International Workshop on Semantic Web Foundations and Application Technologies (SWFAT), Nara, Japan, pp. 51-57.
- Zhao, Y., & Lövdahl, J. (2003, Nov 20-21, 2003). *A Reuse-Based Method of Developing the Ontology for E-Procurement*. Proceedings of the Nordic Conference on Web Services (NCWS), Växjö, Sweden, pp. 101-112.
- Zhao, Y., & Sandahl, K. (2003, April 23-26, 2003). *Potential Advantages of Semantic Web for Internet Commerce*. Proceedings of the International Conference on Enterprise Information Systems (ICEIS), Angers, France, pp. 151-158.

Appendices

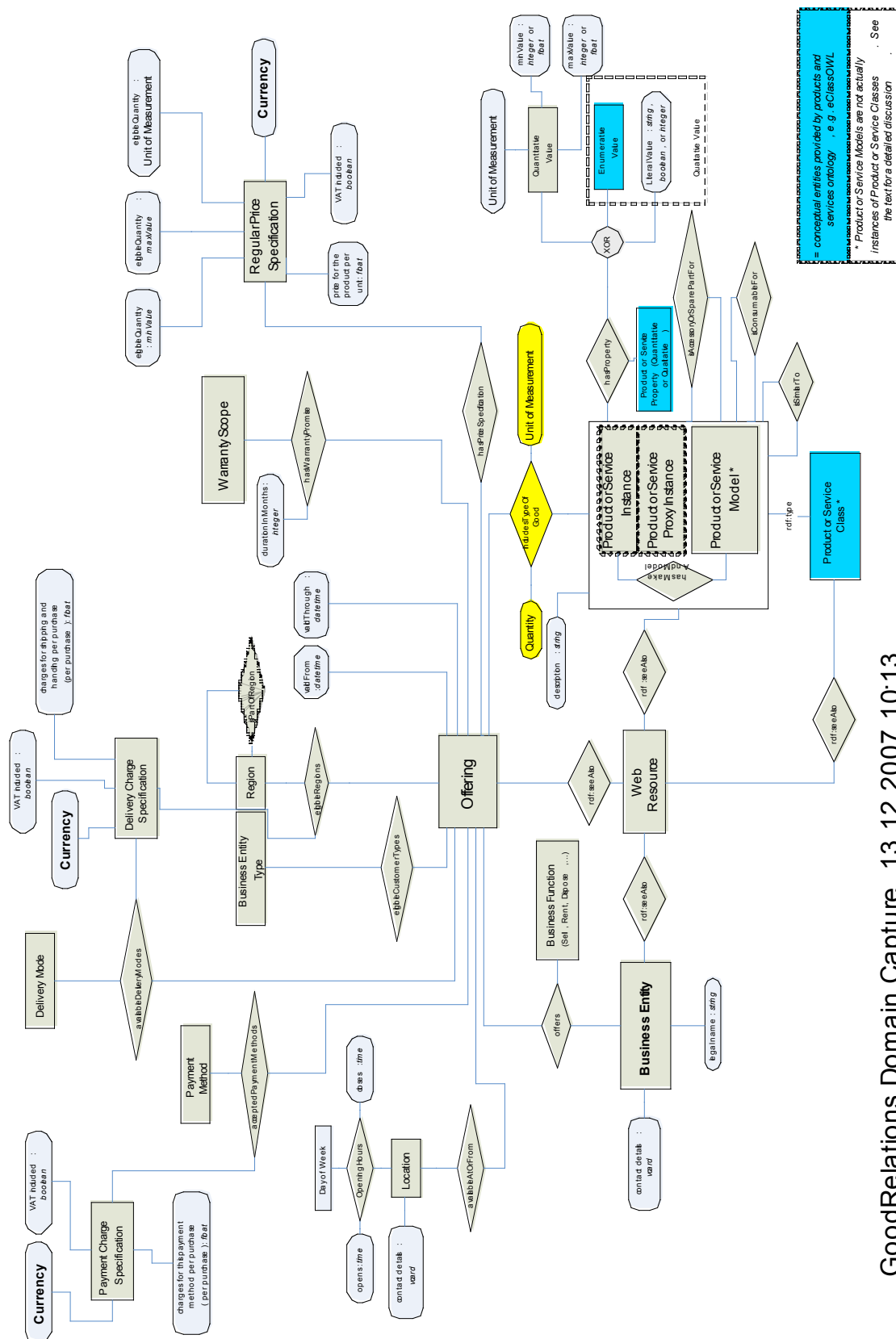


Figure 7. GoodRelations Domain Capture

11 APPENDIX B: GOODRELATIONS DOCUMENTATION

In the following we give a complete list of all elements in the GoodRelations OWL ontology, plus their definition.

Note: This documentation was generated semi-automatically from the ontology. It may not yet reflect the very latest status, since the ongoing testing and evaluation of the ontology keeps on requiring small changes. Only the ontology published on the Web is the authoritative source.

11.1 Classes

BusinessEntity

An instance of this class represents the legal agent making a particular offering. This can be a legal body or a person. A Business Entity has at least a primary mailing address and contact details. For this, typical address standards (vCard) and location data can be attached. The location may be important for finding a supplier within a given distance from our own location. Example: Siemens Austria AG, Volkswagen Ltd., Peter Miller's Cellphone Shop Note: Typical address standards (vcard) and location data should be attached to a business entity. Since there already exist established vocabularies for this, the GoodRelations ontology does not provide respective attributes. Instead, the use of respective vocabularies is recommended.

Domain of: [\[legalName\]](#) [\[offers\]](#)

BusinessEntityType

A Business Entity Type is a conceptual entity representing the legal form, the size, the main line of business, the position in the value chain, or any combination thereof, of a Business Entity. From the ontological point of view, Business Entity Types are mostly roles that a Business Entity has in the market. Business Entity Types are important for specifying eligible customers, since Offerings are often meant only for Business Entities of a certain size, legal structure, or role in the value chain. Examples: Consumers, Retailers, Wholesalers, or Public Institutions

Range of: [\[eligibleCustomerTypes\]](#)

Known Instances:

Business

The BusinessEntityType representing such agents that are themselves offering commercial services or products on the market. Usually, businesses are characterized that they are officially registered with the public administration and strive for profits by their activities.

Type: [\[BusinessEntityType\]](#)

Enduser

The BusinessEntityType representing such agents that are purchasing the good or service for private consumption, in particular not for resale or for usage within an industrial enterprise. By default, a BusinessEntity is an Enduser.

Type: [\[BusinessEntityType\]](#)

PublicInstitution

The BusinessEntityType representing such agents that are part of the administration or owned by the public.

Type: [\[BusinessEntityType\]](#)

Reseller

The BusinessEntityType representing such agents that are purchasing the scope of products included in the Offering for resale on the market. Resellers are also businesses, i.e. they are officially registered with the public administration and strive for profits by their activities.

Type: [\[BusinessEntityType\]](#)

BusinessFunction

The Business Function specifies the type of activity or access offered by the Business Entity on the Product or Services through the Offering. The idea of standardizing business functions was first put to practice by the UNSPSC Business Functions Identifiers (UNSPSC BFI). We take their basic types of business functions as a starting point. Typical are sell, rental or lease, maintenance or repair, manufacture / produce, recycle / dispose, engineering / construction, or installation Examples: A particular offering made by Miller Rentals Ltd. says that they (1) sell Volkswagen Golf convertibles, (2) lease out a particular Ford pick-up truck, and (3) dispose car wrecks of any make and model.

Range of: [\[hasBusinessFunction\]](#)

Final version

Note that the base URI has changed to <http://purl.org/goodrelations/v1#>

Known Instances:

Buy

This Business Function indicates that the Business Entity is in general interested in purchasing the specified Product.

Type: [\[BusinessFunction\]](#)

Dispose

This Business Function indicates that the Business Entity offers to accept the specified Product for proper disposal, recycling, or any other kind of allowed usages, freeing the current owner from all rights and obligations.

Type: [\[BusinessFunction\]](#)

LeaseOut

This Business Function indicates that the Business Entity offers to temporarily grant the right to use the specified Product.

Type: [\[BusinessFunction\]](#)

Maintain

This Business Function indicates that the Business Entity offers to carry out typical maintenance tasks for the specified Product. Maintenance tasks are actions that undo or compensate for wear or other deterioration caused by regular usage, in order to restore the originally intended function of the product, or to prevent outage or malfunction. In other words, the Business Entity is usually able and willing to maintain an object x if x is an instance of the given class of product.

Type: [\[BusinessFunction\]](#)

ProvideService

This Business Function indicates that the Business Entity offers to provide the type of Service. Note: Maintain and Repair are also types of Services. However, eClassOWL and other ontologies provide classes for tangible products as well as for types of services. The business function Provide Service is to be used with such goods that are Services, while Maintain and Repair can be used with goods for which the class of product exists in the ontology, but not the respective type of service. Example: Car maintenance could be expressed both as "Provide Service Car Maintenance" or "Maintain Cars". Since existing ontologies for goods often tangle products and services, it seems beneficial to include Provide Service as a business function.

Type: [\[BusinessFunction\]](#)

Repair

This Business Function indicates that the Business Entity offers to try to evaluate the chances for repairing, and, if positive, repair the specified Product. Repairing means actions that restore the originally intended function of a product that suffers from outage or malfunction. In other words, the Business Entity is usually able and willing to repair an object x if x is an instance of the given class of product.

Type: [\[BusinessFunction\]](#)

Sell

This Business Function indicates that the Business Entity offers to transfer permanently all property rights on the specified Product.

Type: [\[BusinessFunction\]](#)

DayOfWeek

The day of the week, used to specify to which the opening hours of an OpeningHoursSpecification refer. Examples: Monday, Tuesday, Wednesday,...

Range of: [\[hasOpeningHoursDayOfWeek\]](#)

Known Instances:

Friday

Friday as a day of the week.

Type: [\[DayOfWeek\]](#)

Monday

Monday as a day of the week.

Type: [\[DayOfWeek\]](#)

Saturday

Saturday as a day of the week.

Type: [\[DayOfWeek\]](#)

Sunday

Sunday as a day of the week.

Type: [\[DayOfWeek\]](#)

Thursday

Thursday as a day of the week.

Type: [\[DayOfWeek\]](#)

Tuesday

Tuesday as a day of the week.

Type: [\[DayOfWeek\]](#)

Wednesday

Wednesday as a day of the week.

Type: [\[DayOfWeek\]](#)

DeliveryMethod

A Delivery Method is a standardized procedure for transferring the Product or Service Instance to the destination of fulfilment chosen by the customer. Delivery Methods are characterized by the means of transportation used, and by the organization or group that is the contracting party for the sending Business Entity (this is important, since the contracted party may subcontract the fulfilment to smaller, regional businesses). Examples: Delivery by Mail, Delivery by Direct Download, Delivery by UPS

Range of: [\[appliesToDeliveryMethod\]](#) [\[availableDeliveryMethods\]](#)

Known Instances:

DeliveryModeDirectDownload

Delivery of the goods via direct download from the Internet, i.e., the offering Business Entity provides the buying party with details on how to retrieve the goods online. Connection fees and other costs of using the infrastructure are to be carried by the buying party.

Type: [\[DeliveryMethod\]](#)

DeliveryModeMail

Delivery via regular mail service (private or public postal services).

Type: [\[DeliveryMethod\]](#)

Known Subclasses:

DeliveryModeParcelService

A private parcel service as the delivery mode available for a certain offering. Examples: UPS, DHL

Subclass of: [\[DeliveryMethod\]](#)

Known Instances:

DHL

Delivery via the parcel service DHL.

Type: [\[DeliveryModeParcelService\]](#)

FederalExpress

Delivery via the parcel service Federal Express.

Type: [\[DeliveryModeParcelService\]](#)

UPS

Delivery via the parcel service UPS.

Type: [\[DeliveryModeParcelService\]](#)

LocationOfSalesOrServiceProvisioning

A Location of Sales or Service Provisioning is a location from which the specified Business Function on the particular Product or Service Instance is being offered by the Business Entity. Large enterprises often maintain multiple branches from which the delivery or fulfilment can be provided. In this case, the location of the main office of the Business Entity does not state from where a customer can actually get the Offering. Locations of Sales or Service Provisioning are characterized by an address or position and a set of opening hour specifications for various days of the week. Example: A rental car company may offer the Business Function Lease Out of cars from two locations, one in Fort Myers, Florida, and one in Boston, Massachusetts. Both stations are open 7:00 – 23:00 Mondays through Saturdays. Note: Typical address standards (vcard) and location data should be attached to a Location of Sales or Service Provisioning. Since there already exist established vocabularies for this, the GoodRelations ontology does not provide respective attributes. Instead, the use of respective vocabularies is recommended.

Final version

Note that the base URI has changed to <http://purl.org/goodrelations/v1#>

A location at / from which a certain offering is available. In the case of a chain store, it may be all the actual shops. For mail order companies, the location will usually be the headquarter of the BusinessEntity. Examples: InnsbruckBranch, 5thAvenueStore

Range of: [\[availableAtOrFrom\]](#)

N-Ary-Relations

This is the superclass for all classes that are placeholders for n-ary relations, which OWL cannot represent.

Known Subclasses:

AcceptedPaymentMethods

This is a conceptual entity that holds together all aspects of the n-ary relation AcceptedPaymentMethods.

Subclass of: [\[N-Ary-Relations\]](#)

AvailableDeliveryMethods

This is a conceptual entity that holds together all aspects of the n-ary relation AvailableDeliveryMethods

Subclass of: [\[N-Ary-Relations\]](#)

OpeningHoursSpecification

This is a conceptual entity that holds together all aspects of the n-ary relation OpeningHoursSpecification, which defines the opening hours for a given DayOfWeek for a given LocationOfSalesOrServiceProvisioning.

Subclass of: [\[N-Ary-Relations\]](#)

Domain of: [\[closes\]](#) [\[hasOpeningHoursDayOfWeek\]](#) [\[opens\]](#)

TypeAndQuantityNode

This is a conceptual entity that holds together all aspects of the quaternary relation includesTypeOfGood, namely the Quantity, the Unit of Measurement, the Product or Service, and the Offering to which this belongs. Note: The link between Offering and TypeAndQuantityNode is represented by the object property includesObject. The Unit of Measurement is attached using the hasUnitOfMeasurement datatype property. The quantity is specified using the datatype property amountOfThisGood (float). The specification of the item included is represented by the object property typeOfGood. Example: An offering may consists of 100g Butter and 1 kg of potatoes, or 1 cellphone and 2 headsets.

Subclass of: [\[N-Ary-Relations\]](#)

Domain of: [\[amountOfThisGood\]](#) [\[typeOfGood\]](#)

Range of: [\[includesObject\]](#)

WarrantyPromise

This is a conceptual entity that holds together all aspects of the n-ary relation hasWarrantyPromise. A Warranty Promise is an entity representing the duration and scope of services that will be provided to a customer free of charge in case of a defect or malfunction of the Product or Service Instance. A Warranty Promise is characterized by its temporal duration (usually starting with the date of purchase) and its Warranty Scope. The Warranty Scope represents the types of services provides (e.g. labor and parts, just parts) of the warranty included in an Offering. The actual services may be provided by the Business Entity making the Offering, by the manufacturer of the Product, or by a third party. There may be multiple Warranty Promises associated with a particular Offering, which differ in duration and scope (e.g. pick-up service during the first 12 months, just parts and labor for 36 months). Examples: 12 months parts and labor, 36 months parts

Subclass of: [\[N-Ary-Relations\]](#)

Domain of: [\[durationOfWarrantyInMonths\]](#) [\[hasWarrantyScope\]](#)

Range of: [\[hasWarrantyPromise\]](#)

Offering

An Offering represents the public, not necessarily binding, not necessarily exclusive, announcement by a Business Entity to provide a certain Business Function for a certain Product or Service Instance to a specified target audience. An Offering is specified by the type of product or service or bundle it refers to, what Business Function is offered (sales, rental, ...), and a set of commercial properties. It can either refer to a clearly specified instance (Product or Service Instance) or to a set of anonymous instances of a given type (existentially quantified

Product or Service Instances, see also section 3.3.3 of the GoodRelations Technical Report). An offering may be constrained in terms of the eligible type of business partner, countries, quantities, and other commercial properties. The definition of the commercial properties, the type of product offered, and the business function are explained in the following sections in more detail. Example: Peter Miller offers to repair TV sets made by Siemens, Volkswagen Innsbruck sells a particular instance of a Volkswagen Golf at \$10,000

Domain of: [\[acceptedPaymentMethods\]](#) [\[availableAtOrFrom\]](#) [\[availableDeliveryMethods\]](#) [\[eligibleCustomerTypes\]](#) [\[hasBusinessFunction\]](#) [\[hasPriceSpecification\]](#) [\[hasWarrantyPromise\]](#) [\[includesObject\]](#)

Range of: [\[offers\]](#)

PaymentMethod

A Payment Method is a standardized procedure for transferring the monetary amount for a purchase. Payment Methods are characterized by the legal and technical structures used, and by the organization or group carrying out the transaction. It is mostly used for specifying the types of payment accepted by a Business Entity. Examples: Visa, Mastercard, Diners, Cash, Bank transfer in advance

Range of: [\[acceptedPaymentMethods\]](#) [\[appliesToPaymentMethod\]](#)

Known Instances:

ByBankTransferInAdvance

Payment by bank transfer in advance, i.e. the offering Business Entity will inform the buying party about their bank account details and will deliver the goods upon receipt of the due amount.

Type: [\[PaymentMethod\]](#)

ByInvoice

Payment by bank transfer after delivery, i.e. the offering Business Entity will deliver first, inform the buying party about the due amount and their bank account details, and expect payment shortly after delivery.

Type: [\[PaymentMethod\]](#)

Cash

Payment by cash upon delivery or pickup.

Type: [\[PaymentMethod\]](#)

Known Subclasses:

PaymentMethodCreditCard

The subclass of Payment Method represents all variants and brands of credit cards as a standardized procedure for transferring the monetary amount for a purchase. It is mostly used for specifying the types of payment accepted by a Business Entity. Examples: VISA, MasterCard, American Express

Subclass of: [\[PaymentMethod\]](#)

Known Instances:

AmericanExpress

Payment by credit cards issued by the American Express network.

Type: [\[PaymentMethodCreditCard\]](#)

DinersClub

Payment by credit cards issued by the Diner's Club network.

Type: [\[PaymentMethodCreditCard\]](#)

MasterCard

Payment by credit cards issued by the MasterCard network.

Type: [\[PaymentMethodCreditCard\]](#)

VISA

Payment by credit cards issued by the VISA network.

Type: [\[PaymentMethodCreditCard\]](#)

PriceSpecification

The superclass of all price specifications.

Domain of: [\[hasCurrency\]](#) [\[hasCurrencyValue\]](#) [\[hasEligibleQuantity\]](#) [\[valueAddedTaxIncluded\]](#)

Range of: [\[hasPriceSpecification\]](#)

Known Subclasses:

DeliveryChargeSpecification

A Delivery Charge Specification is a conceptual entity that specifies the additional costs asked for delivery of a given Offering using a particular Delivery Method by the respective Business Entity. A Delivery Charge Specification is characterized by (1) a monetary amount per order specified as a literal value of type float in combination with a Currency, (2) the Delivery Method, (3) the target Country or Region, and (4) a whether this charge includes local sales taxes, namely VAT. An Offering may be linked to multiple Delivery Charge Specifications that specify alternative charges for disjoint combinations of target Countries or Regions and Delivery Methods. Examples: Delivery by direct download is free of charge worldwide, delivery by UPS to Germany is 10 Euros per order, delivery by Mail within the US is 5 Euros per order. The total amount of this surcharge is specified as a float value of hasCurrencyValue. The currency is specified via the hasCurrency datatype property. Whether the price includes VAT or not is indicated by the ValueAddedTaxIncluded datatype property. The Delivery Method to which this charge applies is specified using the appliesToDeliveryMethod object property. The region or regions to which this charge applies is specified using the eligibleRegions datatype property, which uses ISO 3166-1 and ISO 3166-2 codes.

Subclass of: [\[PriceSpecification\]](#)

Domain of: [\[appliesToDeliveryMethod\]](#)

PaymentChargeSpecification

A Payment Charge Specification is a conceptual entity that specifies the additional costs asked for settling the payment after accepting a given Offering using a particular Payment Method. A Payment Charge Specification is characterized by (1) a monetary amount per order specified as a literal value of type float in combination with a Currency, (2) the Payment Method, and (3) a whether this charge includes local sales taxes, namely VAT. An Offering may be linked to multiple Payment Charge Specifications that specify alternative charges for various Payment Methods. Examples: Payment by VISA or Mastercard costs a fee of 3 Euros including VAT, Payment by bank transfer in advance is free of charge. The total amount of this surcharge is specified as a float value of hasCurrencyValue. The currency is specified via the hasCurrency datatype property. Whether the price includes VAT or not is indicated by the ValueAddedTaxIncluded datatype property. The Payment Method to which this charge applies is specified using the appliesToPaymentMethod object property.

Subclass of: [\[PriceSpecification\]](#)

Domain of: [\[appliesToPaymentMethod\]](#)

UnitPriceSpecification

A Price Specification is a conceptual entity that specifies the price asked for a given Offering by the respective Business Entity. An Offering may be linked to multiple Price Specifications that specify alternative prices for non-overlapping sets of conditions (e.g. quantities or sales regions). A Price Specification is characterized by (1) the lower and upper limits and the Unit of Measurement of the eligible quantity, (2) by a monetary amount per unit of the Product or Service Instance in the given Unit of Measurement specified as a literal value of type float in combination with a Currency, and (3) a whether this price includes local sales taxes, namely VAT. Example: The price, including VAT, for 1 kg of a given material is 5 Euros per kg for 0 – 5 kg and 4 Euros for quantities above 5 kg. Note: Due to the complexity of pricing scenarios in various industries, it may be necessary to create extensions of this fundamental model of Price Specifications. Such can be done easily by importing and refining the GoodRelations ontology. A specification of the price per unit. This can be constrained to a certain quantity of products. More complex price specifications can be implemented as subclasses of this class. The eligible quantity interval for a given price is specified using the object property hasEligibleQuantity, which points to an instance of Quantitative Value. The currency is specified using the hasCurrency datatype property, which points to an ISO 4217 currency code. The unit of measurement for the eligible quantity is specified using the hasUnitOfMeasurement datatype property, which points to a UN/CEFACT Common Code (3 characters). Whether VAT and sales taxes are included in this price is specified using the datatype property valueAddedTaxIncluded (boolean). The price per unit of measurement is specified using the datatype property priceForProductOrServicePerUnit.

Subclass of: [\[PriceSpecification\]](#)

ProductOrService

The superclass of all classes describing products or services types, either by nature or purpose. Examples for such subclasses are "TV set", "vacuum cleaner", etc. All eClassOWL

"gen" classes are subclasses of this class. An instance of this class can be either an actual product or service or a placeholder instance for unknown instances of a mass-produces commodity. Since eClassOWL and other large products and services ontologies are used for both describing product and services instances and product and service makes and models, this top-level concept is the union of (1) Actual Product or Service Instances, (2) Product or Service Models, and (3) ProductOrServiceSomeInstances Placeholders. The latter are "dummy" instances representing anonymous productst or services instances (i.e. such that are said to exist but not actually being exposed on the Web). See the GoodRelations Technical Report for more details on this. Examples: a) MyCellphone123, i.e. my personal, tangible cell phone b) Siemens1234, i.e. the Siemens cell phone make and model 123 c) dummyCellPhone123 as a placeholder for actual instances of a certain kind of cellphones.

Domain of: [\[datatypeProductOrServiceProperty\]](#) [\[description\]](#) [\[isAccessoryOrSparePartFor\]](#) [\[isConsumableFor\]](#) [\[isSimilarTo\]](#) [\[qualitativeProductOrServiceProperty\]](#) [\[quantitativeProductOrServiceProperty\]](#)

Range of: [\[isAccessoryOrSparePartFor\]](#) [\[isConsumableFor\]](#) [\[isSimilarTo\]](#) [\[typeOfGood\]](#)

Known Subclasses:

ActualProductOrServiceInstance

An Actual Product or Service Instance is a single identifiable object or action that creates some increase in utility (in the economic sense) for the individual possessing or using this very object (Product) or for the individual in whoses favor this very action is being taken (Service). Products or Services are types of goods in the economic sense. For an overview of goods and commodities in economics, see Milgate (1987). Examples: MyThinkpad T60, the pint of beer standing in front of me, my Volkswagen Golf, the haircut that I received or will be receiving at a given date and time. Note: In many cases, product or service instances are not explicitly exposed on the Web but only existentially quantified. For a detailed discussion and practical solutions, see section 3.3.3 of the GoodRelations Technical Report.

Subclass of: [\[ProductOrService\]](#)

ProductOrServiceModel

From the ontological perspective, a Product or Service Model is an intangible entity that specifies some characteristics of a group of similar, usually mass-produced Products. In case of mass-produces Products, there exists a relation hasMakeAndModel between the Products and Services Instance and the Product or Service Model. However, since eClassOWL and other products and services ontologies don't support this important distinction, Product or Service Models are a subclass of Product or Service in GoodRelations. Examples: Ford T, Volkswagen Golf, Sony Ericsson W123 cellphone

Subclass of: [\[ProductOrService\]](#)

Range of: [\[hasMakeAndModel\]](#)

ProductOrServicesSomeInstancesPlaceholder

A placeholder instance for unknown instances of a mass-produces commodity. This is used as a computationally cheap workaround for such instances that are not individually exposed on the Web but just stated to exist (i.e., which are existentially quantified). Example: An instance of this class can represent an anonymous set of green Siemens1234 phones. It is different from the ProductOrServiceModel Siemens1234, since this refers to the make and model, and it is different from a particular instance of this make and mode (e.g. my individual phone) since the latter can be sold only once. Siemens1234, i.e. the Siemens cell phone make and model 123 as a placeholder for all actual instances.

Subclass of: [\[ProductOrService\]](#)

QualitativeValue

A Qualitative Value is an entity that represents the state of a certain qualitative Product or Service Property. Qualitative Values are either Literal Values or Enumerative Values. Literal values are represented just as literals with respective datatype properties. For all other enumerative values, instances of this class are being created. An instance of this class represents a qualitative value for an object property. This is the superclass of all enumerated values in eClassOWL. Examples: the color "green", the power cord plug type "US" Note: Currently, neither value sets nor ordinal relations between values are supported. This can be implemented when needed by importing and refining GoodRelations.

Range of: [\[qualitativeProductOrServiceProperty\]](#)

QuantitativeValue

A Quantitative Value is a numerical interval that represents the range of a certain quantitative Product or Service Property in terms of the lower and upper bounds for one particular Product Or Service. It is to be interpreted in combination with the respective Unit Of Measurement. Most quantitative values are intervals even if they are in practice often treated as a single point. An instance of this class is an actual value for a quantitative property of a product. This instance is usually characterized by a minimal value, a maximal value, and a unit of measurement. This class is a work-around caused by the fact that OWL does only support binary relations, and that datatype ranges cannot be easily handled in OWL. Example: a weight between 10 and 25 kilograms, a length between 10 and 15 millimeters

Domain of: [\[hasMaxValue\]](#) [\[hasMinValue\]](#)

Range of: [\[hasEligibleQuantity\]](#) [\[quantitativeProductOrServiceProperty\]](#)

Known Subclasses:

QuantitativeValueFloat

An instance of this class is an actual float value for a quantitative property of a product. This instance is usually characterized by a minimal value, a maximal value, and a unit of measurement. This class is a work-around caused by the fact that OWL does only support binary relations, and that datatype ranges cannot be easily handled in OWL. Examples: The intervals "between 10.0 and 25.4 kilograms" or "10.2 and 15.5 millimeters"

Subclass of: [\[QuantitativeValue\]](#)

Domain of: [\[hasMaxValueFloat\]](#) [\[hasMinValueFloat\]](#)

QuantitativeValueInteger

An instance of this class is an actual integer value for a quantitative property of a product. This instance is usually characterized by a minimal value, a maximal value, and a unit of measurement. This class is a work-around caused by the fact that OWL does only support binary relations, and that datatype ranges cannot be easily handled in OWL. Example: A seating capacity between 1 and 8 persons Note: Users must keep in mind that ranges in here mean that ALL possible values in this interval are covered. (Sometimes, the actual commitment may be less than that: we rent cars from 2 – 12 seats does often not really mean that they have cars with 2,3,4,...12 seats.). Someone renting two types of rowing boats, one that fits for 1 or 2 people, and another that must be operated by 4 people cannot claim to rent boats with a seating capacity between 1 and 4 people. He or she is renting two boat types for 1-2 and 4.

Subclass of: [\[QuantitativeValue\]](#)

Domain of: [\[hasMaxValueInteger\]](#) [\[hasMinValueInteger\]](#)

WarrantyScope

The Warranty Scope represents types of services that will be provided free of charge by the vendor or manufacturer in the case of a defect (e.g. labor and parts, just parts), as part of the warranty included in an Offering. The actual services may be provided by the Business Entity making the Offering, by the manufacturer of the Product, or by a third party. Examples: Parts and Labor, Parts

Range of: [\[hasWarrantyScope\]](#)

Known Instances:

Labor-BringIn

In case of a defect or malfunction, the buying party has the right to transport the good to a service location determined by the the selling Business Entity and will be charged only for parts and materials needed to fix the problem. Labor will be covered by the selling Business Entity or one of its partnering Business Entities. Note: This is just a rough classification for filtering offers. It is up to the buying party to check the exact scope and terms and conditions of the Warranty Promise.

Type: [\[WarrantyScope\]](#)

PartsAndLabor-BringIn

In case of a defect or malfunction, the buying party has the right to transport the good to a service location determined by the the selling Business Entity and will not be charged for labor, parts, and materials needed to fix the problem. All those costs will be covered by the selling Business Entity or one of its partnering Business Entities. Note: This is just a rough classification for filtering offers. It is up to the buying party to check the exact scope and terms and conditions of the Warranty Promise.

Type: [\[WarrantyScope\]](#)

PartsAndLabor-PickUp

In case of a defect or malfunction, the buying party has the request from the selling Business Entity to pick-up the good from its current location to a suitable service location, where the functionality of the good will be restored. All labor, parts, and materials needed to fix the problem will be covered by the selling Business Entity or one of its partnering Business Entities.

Type: [\[WarrantyScope\]](#)

11.2 Object Properties

acceptedPaymentMethods

The PaymentMethods accepted by the BusinessEntity for the given Offering.

Domain: [\[Offering\]](#)

Range: [\[PaymentMethod\]](#)

appliesToDeliveryMethod

This property specifies the Delivery Method to which the Delivery Charge Specification applies.

Domain: [\[DeliveryChargeSpecification\]](#)

Range: [\[DeliveryMethod\]](#)

appliesToPaymentMethod

This property specifies the Payment Method to which the Payment Charge Specification applies.

Domain: [\[PaymentChargeSpecification\]](#)

Range: [\[PaymentMethod\]](#)

availableAtOrFrom

This states that a particular Offering is available at or from the given LocationOfSalesOrServiceProvisioning (e.g. shop or branch).

Domain: [\[Offering\]](#)

Range: [\[LocationOfSalesOrServiceProvisioning\]](#)

availableDeliveryMethods

This specifies the DeliveryMethods available for a given Offering.

Domain: [\[Offering\]](#)

Range: [\[DeliveryMethod\]](#)

eligibleCustomerTypes

The types of customers (CustomerType) for which the given Offering is valid.

Domain: [\[Offering\]](#)

Range: [\[BusinessEntityType\]](#)

hasBusinessFunction

This specifies the BusinessFunction of the Offering, i.e. whether the BusinessEntity is offering to sell, to lease, or to repair the particular type of product. Note: While it is possible that an entity is offering multiple types of business functions, this should usually not be stated by multiple statements attached to the same offering, since the UnitPriceSpecification for the varying BusinessFunctions will usually be very different.

Domain: [\[Offering\]](#)

Range: [\[BusinessFunction\]](#)

hasEligibleQuantity

This specifies the interval and unit of measurement of ordering quantities for which the PriceSpecification is valid. This allows e.g. specifying that a certain freight charge is valid only for a certain quantity. Note that if an offering is a bundle, i.e. it consists of more than one unit of a single type of good, or if the unit of measurement for the good is different from unit (Common Code C62), then hasEligible Quantity refers to units of this bundle.

Domain: [\[PriceSpecification\]](#)

Range: [\[QuantitativeValue\]](#)

hasMakeAndModel

This states that an actual product instance (`ActualProductOrServiceInstance`) or a placeholder instance for multiple, unidentified such instances (represented by an instance of `ProductOrServicesSomeInstancesPlaceholder`) is one occurrence of a particular Product or Service Model. Example: `myFordT` hasMakeAndModel `FordT`

Domain: ([[ActualProductOrServiceInstance](#)] or [[ProductOrServicesSomeInstancesPlaceholder](#)])

Range: [[ProductOrServiceModel](#)]

hasOpeningHoursDayOfWeek

This specifies the DayOfWeek to which the OpeningHoursSpecification is related.

Domain: [[OpeningHoursSpecification](#)]

Range: [[DayOfWeek](#)]

hasPriceSpecification

This links an Offering to one or more PriceSpecifications. There can be UnitPriceSpecifications, Payment Charge Specifications, and Delivery Charge Specifications. For each type multiple PriceSpecifications for the same Offering are possible, e.g. for different quantity ranges or for different currencies, or for different combinations of Delivery Method and target destination.

Domain: [[Offering](#)]

Range: [[PriceSpecification](#)]

hasWarrantyPromise

This specified the WarrantyPromise made by the BusinessEntity for the given Offering.

Domain: [[Offering](#)]

Range: [[WarrantyPromise](#)]

hasWarrantyScope

This states the WarrantyScope of a given WarrantyPromise.

Domain: [[WarrantyPromise](#)]

Range: [[WarrantyScope](#)]

includesObject

This object property links an Offering to one or multiple TypeAndQuantityNode that specify the components that are included in the respective offer.

Domain: [[Offering](#)]

Range: [[TypeAndQuantityNode](#)]

isAccessoryOrSparePartFor

This states that a particular ProductOrService is an accessory or spare part for another ProductOrService.

Domain: [[ProductOrService](#)]

Range: [[ProductOrService](#)]

isConsumableFor

This states that a particular ProductOrService is a consumable for another ProductOrService.

Domain: [[ProductOrService](#)]

Range: [[ProductOrService](#)]

isSimilarTo

This states that a given ProductOrService is similar to another ProductOrService. Of course, this is a subjective statement; when interpreting it, the trust in the origin of the statement should be taken into account.

Domain: [[ProductOrService](#)]

Range: [[ProductOrService](#)]

offers

This links a BusinessEntity to the respective Offering.

Domain: [[BusinessEntity](#)]

Range: [[Offering](#)]

qualitativeProductOrServiceProperty

This is the super property of all qualitative properties for products and services. All eclassOWL properties for which QualitativeValue instances are specified are subproperties of this property.

Domain: [\[ProductOrService\]](#)

Range: [\[QualitativeValue\]](#)

quantitativeProductOrServiceProperty

This is the super property of all quantitative properties for products and services. All eclassOWL properties that specify quantitative characteristics, for which an interval is at least theoretically an appropriate value, are specified as subproperties of this property.

Domain: [\[ProductOrService\]](#)

Range: [\[QuantitativeValue\]](#)

typeOfGood

This specifies the type of Product or Service the TypeAndQuantityNode is referring to.

Domain: [\[TypeAndQuantityNode\]](#)

Range: [\[ProductOrService\]](#)

11.3 Datatype Properties

amountOfThisGood

This property specifies the quantity of the goods included in the Offering via this TypeAndQuantityNode. The quantity is given in the UnitOfMeasurement attached to the TypeAndQuantityNode.

Domain: [\[TypeAndQuantityNode\]](#)

Range: [\[http://www.w3.org/2001/XMLSchema#float\]](http://www.w3.org/2001/XMLSchema#float)

closes

The closing hour of the LocationOfSalesOrServiceProvisioning on the given DayOfWeek given in the local time valid at the Location.

Domain: [\[OpeningHoursSpecification\]](#)

Range: [\[http://www.w3.org/2001/XMLSchema#time\]](http://www.w3.org/2001/XMLSchema#time)

datatypeProductOrServiceProperty

This property is the super property for all pure datatype properties that can be used to describe a product and services instance, or via the instances placeholders, of a set of instances of mass-produces commodities. Only such eClassOWL properties that are no quantitative properties and that have no predefined QualitativeValue instances are subproperties of this property. In practice, this refers to a few integer properties for which the integer value represents qualitative aspects, for string datatypes (as long as no predefined values exist), and for boolean datatype properties.

Domain: [\[ProductOrService\]](#)

description

A short textual description of the product or service. This can be easily extracted by search engines and other applications.

Domain: [\[ProductOrService\]](#)

Range: [\[http://www.w3.org/2001/XMLSchema#string\]](http://www.w3.org/2001/XMLSchema#string)

durationOfWarrantyInMonths

This property specifies the duration of the WarrantyPromise in months.

Domain: [\[WarrantyPromise\]](#)

Range: [\[http://www.w3.org/2001/XMLSchema#int\]](http://www.w3.org/2001/XMLSchema#int)

eligibleRegions

This property specifies the geo-political region or regions for which the offer is valid using the two-character version of ISO 3166-1 (ISO 3166-1 alpha-2) for regions or ISO 3166-2, which breaks down the countries from ISO 3166-1 into administrative subdivisions. Important: Do NOT use 3-letter ISO 3166-1 codes!

Domain: ([\[Offering\]](#) or [\[DeliveryChargeSpecification\]](#))

Range: [\[http://www.w3.org/2001/XMLSchema#string\]](http://www.w3.org/2001/XMLSchema#string)

hasCurrency

The currency for all prices in the PriceSpecification given using the ISO 4217 standard (3 characters).

Domain: [\[PriceSpecification\]](#)

Range: [<http://www.w3.org/2001/XMLSchema#string>]

hasCurrencyValue

This property specifies the amount of money for a price per unit, shipping charges, or payment charges. The currency and other relevant details are attached to the respective PriceSpecification etc. For a Unit Price Specification, this is the price for one unit (as specified in the unit of measurement of the UnitPriceSpecification) of the respective ProductOrService. For a Delivery Charge Specification or a Payment Charge Specification, it is the price per delivery or payment.

Domain: [[PriceSpecification](#)]

Range: [<http://www.w3.org/2001/XMLSchema#float>]

hasMaxValue

This property captures the upper limit of a QuantitativeValue instance.

Domain: [[QuantitativeValue](#)]

Known Subproperties:

hasMaxValueFloat

This property captures the upper limit of a QuantitativeValueFloat instance.

Subproperty of: [[hasMaxValue](#)]

Domain: [[QuantitativeValueFloat](#)]

Range: [<http://www.w3.org/2001/XMLSchema#float>]

Known Subproperties:

hasValueFloat

This subproperty specifies that the upper and lower limit of the given QuantitativeValueFloat are identical and have the respective float value. It is a shortcut for such cases where a quantitative property is (at least practically) a single value and not an interval.

Subproperty of: [[hasMaxValueFloat](#)] [[hasMinValueFloat](#)]

hasMaxValueInteger

This property captures the upper limit of a QuantitativeValueInteger instance.

Subproperty of: [[hasMaxValue](#)]

Domain: [[QuantitativeValueInteger](#)]

Range: [<http://www.w3.org/2001/XMLSchema#int>]

Known Subproperties:

hasValueInteger

This subproperty specifies that the upper and lower limit of the given QuantitativeValueInteger are identical and have the respective integer value. It is a shortcut for such cases where a quantitative property is (at least practically) a single value and not an interval.

Subproperty of: [[hasMaxValueInteger](#)] [[hasMinValueInteger](#)]

hasMinValue

This property captures the lower limit of a QuantitativeValue instance.

Domain: [[QuantitativeValue](#)]

Known Subproperties:

hasMinValueFloat

This property captures the lower limit of a QuantitativeValueFloat instance.

Subproperty of: [[hasMinValue](#)]

Domain: [[QuantitativeValueFloat](#)]

Range: [<http://www.w3.org/2001/XMLSchema#float>]

Known Subproperties:

hasValueFloat

This subproperty specifies that the upper and lower limit of the given QuantitativeValueFloat are identical and have the respective float value. It is a shortcut for such cases where a quantitative property is (at least practically) a single value and not an interval.

Subproperty of: [[hasMaxValueFloat](#)] [[hasMinValueFloat](#)]

hasMinValueInteger

This property captures the lower limit of a QuantitativeValueInteger instance.

Subproperty of: [[hasMinValue](#)]

Domain: [\[QuantitativeValueInteger\]](#)

Range: [\[http://www.w3.org/2001/XMLSchema#int\]](http://www.w3.org/2001/XMLSchema#int)

Known Subproperties:

hasValueInteger

This subproperty specifies that the upper and lower limit of the given QuantitativeValueInteger are identical and have the respective integer value. It is a shortcut for such cases where a quantitative property is (at least practically) a single value and not an interval.

Subproperty of: [\[hasMaxValueInteger\]](#) [\[hasMinValueInteger\]](#)

hasUnitOfMeasurement

The unit of measurement for a QuantitativeValue, a Unit Price Specification, or a TypeAndQuantityNode given using the UN/CEFACT Common Code (3 characters).

Domain: ([\[QuantitativeValue\]](#) or [\[UnitPriceSpecification\]](#) or [\[TypeAndQuantityNode\]](#))

Range: [\[http://www.w3.org/2001/XMLSchema#string\]](http://www.w3.org/2001/XMLSchema#string)

legalName

The legal name of the business entity.

Domain: [\[BusinessEntity\]](#)

Range: [\[http://www.w3.org/2001/XMLSchema#string\]](http://www.w3.org/2001/XMLSchema#string)

opens

The opening hour of the LocationOfSalesOrServiceProvisioning on the given DayOfWeek given in the local time valid at the Location.

Domain: [[OpeningHoursSpecification](#)]

Range: [<http://www.w3.org/2001/XMLSchema#time>]

validFrom

This property specifies the beginning of the validity of the Offering. The point in time must be in GMT.

Domain: ([[Offering](#)] or [[UnitPriceSpecification](#)])

Range: [<http://www.w3.org/2001/XMLSchema#dateTime>]

validThrough

This property specifies the end of the validity of the Offering. The point in time must be in GMT.

Domain: ([[Offering](#)] or [[UnitPriceSpecification](#)])

Range: [<http://www.w3.org/2001/XMLSchema#dateTime>]

valueAddedTaxIncluded

This property specifies whether the applicable value-added tax (VAT) is included in the prices of the PriceSpecification or not. It determines this feature for all types of PriceSpecifications, i.e. UnitPriceSpecifications, DeliveryChargeSpecifications, and PaymentChargeSpecifications.

Note: This is a simple representation which may not properly reflect all details of local taxation.

Domain: [[PriceSpecification](#)]

Range: [<http://www.w3.org/2001/XMLSchema#boolean>]

12 APPENDIX C: GOODRELATIONS ONTOLOGY SPECIFICATION IN OWL DLP

Note: This section may not reflect the very latest status, since the ongoing testing and evaluation of the ontology keeps on requiring small changes. Only the ontology published on the Web is the authoritative source.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.heppnetz.de/ontologies/goodrelations/v1#"
  xmlns:assert="http://www.owl-ontologies.com/assert.owl#"
  xml:base="http://www.heppnetz.de/ontologies/goodrelations/v1">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.owl-ontologies.com/assert.owl"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      >The GoodRelations ontology provides the vocabulary for annotating e-
      commerce offerings (1) to sell, lease, repair, dispose, and maintain
      commodity products and (2) to provide commodity services.
```

GoodRelations allows describing the relationship between (1) Web resources, (2) offerings made by those Web resources, (3) legal entities, (4) prices, (5) terms and conditions, and the aforementioned ontologies for products and services (6).

For more information, see

<http://www.heppnetz.de/projects/goodrelations/></rdfs:comment>

```
  <rdfs:seeAlso
    rdf:resource="http://www.heppnetz.de/projects/goodrelations/">
  </owl:Ontology>
  <owl:Class rdf:ID="PaymentChargeSpecification">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="PriceSpecification"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      >A Payment Charge Specification is a conceptual entity that specifies the
      additional costs asked for settling the payment after accepting a given
      Offering using a particular Payment Method. A Payment Charge Specification is
      characterized by (1) a monetary amount per order specified as a literal value
      of type float in combination with a Currency, (2) the Payment Method, and (3)
      a whether this charge includes local sales taxes, namely VAT.
      An Offering may be linked to multiple Payment Charge Specifications that
      specify alternative charges for various Payment Methods.
```

Examples: Payment by VISA or Mastercard costs a fee of 3 Euros including VAT, Payment by bank transfer in advance is free of charge.

The total amount of this surcharge is specified as a float value of hasCurrencyValue. The currency is specified via the hasCurrency datatype property. Whether the price includes VAT or not is indicated by the ValueAddedTaxIncluded datatype property. The Payment Method to which this charge applies is specified using the appliesToPaymentMethod object property.</rdfs:comment>

```
  </owl:Class>
  <owl:Class rdf:about="#PriceSpecification">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      >The superclass of all price specifications.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="QuantitativeValueFloat">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="QuantitativeValue"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      >An instance of this class is an actual float value for a quantitative
      property of a product. This instance is usually characterized by a minimal
      value, a maximal value, and a unit of measurement. This class is a work-around
      caused by the fact that OWL does only support binary relations, and that
      datatype ranges cannot be easily handled in OWL.
```

SEBIS Technical Report

Examples: The intervals "between 10.0 and 25.4 kilograms" or "10.2 and 15.5 milimeters"</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="WarrantyPromise">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="N-Ary-Relations"/>
  </rdfs:subClassOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This is a conceptual entity that holds together all aspects of the n-ary
    relation hasWarrantyPromise.
  </rdfs:comment>
</owl:Class>
```

A Warranty Promise is an entity representing the duration and scope of services that will be provided to a customer free of charge in case of a defect or malfunction of the Product or Service Instance. A Warranty Promise is characterized by its temporal duration (usually starting with the date of purchase) and its Warranty Scope. The Warranty Scope represents the types of services provides (e.g. labor and parts, just parts) of the warranty included in an Offering. The actual services may be provided by the Business Entity making the Offering, by the manufacturer of the Product, or by a third party. There may be multiple Warranty Promises associated with a particular Offering, which differ in duration and scope (e.g. pick-up service during the first 12 months, just parts and labor for 36 months).

Examples: 12 months parts and labor, 36 months parts</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="QualitativeValue">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >A Qualitative Value is an entity that represents the state of a certain
    qualitative Product or Service Property. Qualitative Values are either Literal
    Values or Enumerative Values. Literal values are represented just as literals
    with respective datatype properties. For all other enumerative values,
    instances of this class are being created.
  </rdfs:comment>
</owl:Class>
```

An instance of this class represents a qualitative value for an object property. This is the superclass of all enumerated values in eClassOWL.

Examples: the color "green", the power cord plug type "US"

Note: Currently. neither value sets nor ordinal relations between values are supported. This can be implemented when needed by importing and refining GoodRelations.</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="BusinessFunction">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >The Business Function specifies the type of activity or access offered by
    the Business Entity on the Product or Services though the Offering. The idea
    of standardizing business functions was first put to practice by the UNSPSC
    Business Functions Identifiers (UNSPSC BFI). We take their basic types of
    business functions as a starting point. Typical are sell, rental or lease,
    maintenance or repair, manufacture / produce, recycle / dispose, engineering /
    construction, or installation
  </rdfs:comment>
</owl:Class>
```

Examples: A particular offering made by Miller Rentals Ltd. says that they (1) sell Volkswagen Golf convertibles, (2) lease out a particular Ford pick-up truck, and (3) dispose car wrecks of any make and model.</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="DeliveryMethod">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >A Delivery Method is a standardized procedure for transferring the
    Product or Service Instance to the destination of fulfilment chosen by the
    customer. Delivery Methods are characterized by the means of transportation
    used, and by the organization or group that is the contracting party for the
    sending Business Entity (this is important, since the contracted party may
    subcontract the fulfilment to smaller, regional businesses).
  </rdfs:comment>
</owl:Class>
```

Examples: Delivery by Mail, Delivery by Direct Download, Delivery by UPS</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="LocationOfSalesOrServiceProvisioning">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >A Location of Sales or Service Provisioning is a standardized procedure
    for transferring the Product or Service Instance to the destination of
    fulfilment chosen by the customer. Location of Sales or Service Provisioning
    are characterized by the means of transportation used, and by the
    organization or group that is the contracting party for the sending
    Business Entity (this is important, since the contracted party may
    subcontract the fulfilment to smaller, regional businesses).
  </rdfs:comment>
</owl:Class>
```

SEBIS Technical Report

>A Location of Sales or Service Provisioning is a location from which the specified Business Function on the particular Product or Service Instance is being offered by the Business Entity. Large enterprises often maintain multiple branches from which the delivery or fulfilment can be provided. In this case, the location of the main office of the Business Entity does not state from where a customer can actually get the Offering. Locations of Sales or Service Provisioning are characterized by an address or position and a set of opening hour specifications for various days of the week.

Example: A rental car company may offer the Business Function Lease Out of cars from two locations, one in Fort Myers, Florida, and one in Boston, Massachusetts. Both stations are open 7:00 – 23:00 Mondays through Saturdays.

Note: Typical address standards (vcard) and location data should be attached to a Location of Sales or Service Provisioning. Since there already exist established vocabularies for this, the GoodRelations ontology does not provide respective attributes. Instead, the use of respective vocabularies is recommended.

A location at / from which a certain offering is available. In the case of a chain store, it may be all the actual shops. For mail order companies, the location will usually be the headquarter of the BusinessEntity.

Examples: InnsbruckBranch, 5thAvenueStore</rdfs:comment>

```
</owl:Class>
```

```
<owl:Class rdf:ID="ActualProductOrServiceInstance">
```

```
<rdfs:subClassOf>
```

```
<owl:Class rdf:ID="ProductOrService"/>
```

```
</rdfs:subClassOf>
```

```
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

>An Actual Product or Service Instance is a single identifiable object or action that creates some increase in utility (in the economic sense) for the individual possessing or using this very object (Product) or for the individual in whose favor this very action is being taken (Service). Products or Services are types of goods in the economic sense. For an overview of goods and commodities in economics, see Milgate (1987).

Examples: MyThinkpad T60, the pint of beer standing in front of me, my Volkswagen Golf, the haircut that I received or will be receiving at a given date and time.

Note: In many cases, product or service instances are not explicitly exposed on the Web but only existentially quantified. For a detailed discussion and practical solutions, see section 3.3.3 of the GoodRelations Technical Report.</rdfs:comment>

```
</owl:Class>
```

```
<owl:Class rdf:ID="BusinessEntity">
```

```
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

>An instance of this class represents the legal agent making a particular offering. This can be a legal body or a person. A Business Entity has at least a primary mailing address and contact details. For this, typical address standards (vCard) and location data can be attached. The location may be important for finding a supplier within a given distance from our own location.

Example: Siemens Austria AG, Volkswagen Ltd., Peter Miller's Cellphone Shop

Note: Typical address standards (vcard) and location data should be attached to a business entity. Since there already exist established vocabularies for this, the GoodRelations ontology does not provide respective attributes. Instead, the use of respective vocabularies is recommended.</rdfs:comment>

```
</owl:Class>
```

```
<owl:Class rdf:ID="PaymentMethod">
```

```
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

>A Payment Method is a standardized procedure for transferring the monetary amount for a purchase. Payment Methods are characterized by the legal and technical structures used, and by the organization or group carrying out

SEBIS Technical Report

the transaction. It is mostly used for specifying the types of payment accepted by a Business Entity.

Examples: Visa, Mastercard, Diners, Cash, Bank transfer in advance</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="DayOfWeek">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The day of the week, used to specify to which the opening hours of an
    OpeningHoursSpecification refer.
```

Examples: Monday, Tuesday, Wednesday,...</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="DeliveryChargeSpecification">
  <rdfs:subClassOf rdf:resource="#PriceSpecification"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A Delivery Charge Specification is a conceptual entity that specifies the
    additional costs asked for delivery of a given Offering using a particular
    Delivery Method by the respective Business Entity. A Delivery Charge
    Specification is characterized by (1) a monetary amount per order specified as
    a literal value of type float in combination with a Currency, (2) the Delivery
    Method, (3) the target Country or Region, and (4) a whether this charge
    includes local sales taxes, namely VAT.
    An Offering may be linked to multiple Delivery Charge Specifications that
    specify alternative charges for disjoint combinations of target Countries or
    Regions and Delivery Methods.
```

Examples: Delivery by direct download is free of charge worldwide, delivery by UPS to Germany is 10 Euros per order, delivery by Mail within the US is 5 Euros per order

The total amount of this surcharge is specified as a float value of hasCurrencyValue. The currency is specified via the hasCurrency datatype property. Whether the price includes VAT or not is indicated by the ValueAddedTaxIncluded datatype property. The Delivery Method to which this charge applies is specified using the appliesToDeliveryMethod object property. The region or regions to which this charge applies is specified using the eligibleRegions datatype property, which uses ISO 3166-1 and ISO 3166-2 codes.</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:about="#QuantitativeValue">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A Quantitative Value is a numerical interval that represents the range of
    a certain quantitative Product or Service Property in terms of the lower and
    upper bounds for one particular Product Or Service. It is to be interpreted in
    combination with the respective Unit Of Measurement. Most quantitative values
    are intervals even if they are in practice often treated as a single point.
```

An instance of this class is an actual value for a quantitative property of a product. This instance is usually characterized by a minimal value, a maximal value, and a unit of measurement. This class is a work-around caused by the fact that OWL does only support binary relations, and that datatype ranges cannot be easily handled in OWL.

Example: a weight between 10 and 25 kilograms, a length between 10 and 15 millimeters</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="UnitPriceSpecification">
  <rdfs:subClassOf rdf:resource="#PriceSpecification"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A Price Specification is a conceptual entity that specifies the price
    asked for a given Offering by the respective Business Entity. An Offering may
    be linked to multiple Price Specifications that specify alternative prices for
    non-overlapping sets of conditions (e.g. quantities or sales regions).
```

A Price Specification is characterized by (1) the lower and upper limits and the Unit of Measurement of the eligible quantity, (2) by a monetary amount per unit of the Product or Service Instance in the given Unit of Measurement specified as a literal value of type float in combination with a Currency, and (3) a whether this prices includes local sales taxes, namely VAT.

SEBIS Technical Report

Example: The price, including VAT, for 1 kg of a given material is 5 Euros per kg for 0 ≤ 5 kg and 4 Euros for quantities above 5 kg

Note: Due to the complexity of pricing scenarios in various industries, it may be necessary to create extensions of this fundamental model of Price Specifications.

Such can be done easily by importing and refining the GoodRelations ontology.

A specification of the price per unit. This can be constrained to a certain quantity of products. More complex price specifications can be implemented as subclasses of this class

The eligible quantity interval for a given price is specified using the object property `hasEligibleQuantity`, which points to an instance of `Quantitative Value`.

The currency is specified using the `hasCurrency` datatype property, which point to an ISO 4217 currency code.

The unit of measurement for the eligible quantity is specified using the `hasUnitOfMeasurement` datatype property, which points to a UN/CEFACT Common Code (3 characters).

Whether VAT and sales taxes are included in this price is specified using the datatype property `valueAddedTaxIncluded` (boolean).

The price per unit of measurement is specified using the datatype property `priceForProductOrServicePerUnit`.

```
</rdf:comment>
</owl:Class>
<owl:Class rdf:about="#N-Ary-Relations">
  <rdf:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This is the superclass for all classes that are placeholders for n-ary
    relations, which OWL cannot represent.</rdf:comment>
  </owl:Class>
<owl:Class rdf:ID="ProductOrServicesSomeInstancesPlaceholder">
  <rdf:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >A placeholder instance for unknown instances of a mass-produces
    commodity. This is used as a computationally cheap workaround for such
    instances that are not individually exposed on the Web but just stated to
    exist (i.e., which are existentially quantified).
```

Example: An instance of this class can represent an anonymous set of green Siemens1234 phones. It is different from the `ProductOrServiceModel Siemens1234`, since this refers to the make and model, and it is different from a particular instance of this make and mode (e.g. my individual phone) since the latter can be sold only once.

`Siemens1234`, i.e. the Siemens cell phone make and model 123 as a placeholder for all actual instances.

```
<rdf:subClassOf>
  <owl:Class rdf:about="#ProductOrService"/>
</rdf:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AvailableDeliveryMethods">
  <rdf:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This is a conceptual entity that holds together all aspects of the n-ary
    relation AvailableDeliveryMethods</rdf:comment>
  <rdf:subClassOf rdf:resource="#N-Ary-Relations"/>
</owl:Class>
<owl:Class rdf:ID="AcceptedPaymentMethods">
  <rdf:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This is a conceptual entity that holds together all aspects of the n-ary
    relation AcceptedPaymentMethods.</rdf:comment>
  <rdf:subClassOf rdf:resource="#N-Ary-Relations"/>
</owl:Class>
<owl:Class rdf:about="#ProductOrService">
  <rdf:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >The superclass of all classes describing products or services types,
    either by nature or purpose. Examples for such subclasses are "TV set",
    "vacuum cleaner", etc. All eClassOWL "gen" classes are subclasses of this
    class.
  An instance of this class can be either an actual product or service or a
  placeholder instance for unknown instances of a mass-produces commodity.
```

SEBIS Technical Report

Since eClassOWL and other large products and services ontologies are used for both describing product and services instances and product and service makes and models, this top-level concept is the union of (1) Actual Product or Service Instances, (2) Product or Service Models, and (3) ProductOrServiceSomeInstances Placeholders. The latter are "dummy" instances representing anonymous productst or services instances (i.e. such that are said to exist but not actually being exposed on the Web).

See the GoodRelations Technical Report for more details on this.

Examples:

- a) MyCellphone123, i.e. my personal, tangible cell phone
- b) Siemens1234, i.e. the Siemens cell phone make and model 123
- c) dummyCellPhone123 as a placeholder for actual instances of a certain kind of cellphones.

```
</owl:Class>
<owl:Class rdf:ID="Offering">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >An Offering represents the public, not necessarily binding, not
    necessarily exclusive, announcement by a Business Entity to provide a certain
    Business Function for a certain Product or Service Instance to a specified
    target audience. An Offering is specified by the type of product or service or
    bundle it refers to, what Business Function is offered (sales, rental, etc.),
    and a set of commercial properties. It can either refer to a clearly specified
    instance (Product or Service Instance) or to a set of anonymous instances of a
    given type (existentially quantified Product or Service Instances, see also
    section 3.3.3 of the GoodRelations Technical Report). An offering may be
    constrained in terms of the eligible type of business partner, countries,
    quantities, and other commercial properties. The definition of the commercial
    properties, the type of product offered, and the business function are
    explained in the following sections in more detail.
```

Example: Peter Miller offers to repair TV sets made by Siemens, Volkswagen
Innsbruck sells a particular instance of a Volkswagen Golf at
\$10,000

```
</owl:Class>
<owl:Class rdf:ID="WarrantyScope">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The Warranty Scope represents types of services that will be provided
    free of charge by the vendor or manufacturer in the case of a defect (e.g.
    labor and parts, just parts), as part of the warranty included in an Offering.
    The actual services may be provided by the Business Entity making the
    Offering, by the manufacturer of the Product, or by a third party.
```

Examples: Parts and Labor, Parts

```
</owl:Class>
<owl:Class rdf:ID="OpeningHoursSpecification">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This is a conceptual entity that holds together all aspects of the n-ary
    relation OpeningHoursSpecification, which defines the opening hours for a
    given DayOfWeek for a given
    LocationOfSalesOrServiceProvisioning.
  <rdfs:subClassOf rdf:resource="#N-Ary-Relations"/>
</owl:Class>
<owl:Class rdf:ID="ProductOrServiceModel">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >From the ontological perspective, a Product or Service Model is an
    intangible entity that specifies some characteristics of a group of similar,
    usually mass-produced Products. In case of mass-produces Products, there
    exists a relation hasMakeAndModel between the Products and Services Instance
    and the Product or Service Model.
```

However, since eClassOWL and other products and services ontologies don't support this important distinction, Product or Service Models are a subclass of Product or Service in GoodRelations.

```
Examples: Ford T, Volkswagen Golf, Sony Ericsson W123 cellphone
  <rdfs:subClassOf rdf:resource="#ProductOrService"/>
</owl:Class>
<owl:Class rdf:ID="DeliveryModeParcelService">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

SEBIS Technical Report

>A private parcel service as the delivery mode available for a certain offering.

Examples: UPS, DHL</rdfs:comment>

```
<rdfs:subClassOf rdf:resource="#DeliveryMethod"/>
</owl:Class>
<owl:Class rdf:ID="TypeAndQuantityNode">
  <rdfs:subClassOf rdf:resource="#N-Ary-Relations"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This is a conceptual entity that holds together all aspects of the
    quaternary relation includesTypeOfGood, namely the Quantity, the Unit of
    Measurement, the Product or Service, and the Offering to which this belongs.
```

Note: The link between Offering and TypeAndQuantityNode is represented by the object property includesObject. The Unit of Measurement is attached using the hasUnitOfMeasurement datatype property. The quantity is specified using the datatype property amountOfThisGood (float). The specification of the item included is represented by the object property typeOfGood.

Example: An offering may consists of 100g Butter and 1 kg of potatoes, or 1 cellphone and 2 headsets.</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="PaymentMethodCreditCard">
  <rdfs:subClassOf rdf:resource="#PaymentMethod"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The subclass of Payment Method represents all variants and brands of
    credit cards as a standardized procedure for transferring the monetary amount
    for a purchase. It is mostly used for specifying the types of payment accepted
    by a Business Entity.
```

Examples: VISA, MasterCard, American Express</rdfs:comment>

```
</owl:Class>
<owl:Class rdf:ID="QuantitativeValueInteger">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >An instance of this class is an actual integer value for a quantitative
    property of a product. This instance is usually characterized by a minimal
    value, a maximal value, and a unit of measurement. This class is a work-around
    caused by the fact that OWL does only support binary relations, and that
    datatype ranges cannot be easily handled in OWL.
```

Example: A seating capacity between 1 and 8 persons

Note: Users must keep in mind that ranges in here mean that ALL possible values in this interval are covered. (Sometimes, the actual commitment may be less than that: we rent cars from 2 to 12 seats does often not really mean that they have cars with 2,3,4,...,12 seats.). Someone renting two types of rowing boats, one that fits for 1 or 2 people, and another that must be operated by 4 people cannot claim to rent boats with a seating capacity between 1 and 4 people. He or she is renting two boat types for 1-2 and 4.</rdfs:comment>

```
<rdfs:subClassOf rdf:resource="#QuantitativeValue"/>
</owl:Class>
<owl:Class rdf:ID="BusinessEntityType">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A Business Entity Type is a conceptual entity representing the legal
    form, the size, the main line of business, the position in the value chain, or
    any combination thereof, of a Business Entity. From the ontological point of
    view, Business Entity Types are mostly roles that a Business Entity has in the
    market. Business Entity Types are important for specifying eligible customers,
    since Offerings are often meant only for Business Entities of a certain size,
    legal structure, or role in the value chain.
```

Examples: Consumers, Retailers, Wholesalers, or Public Institutions</rdfs:comment>

```
</owl:Class>
<owl:ObjectProperty rdf:ID="hasBusinessFunction">
  <rdfs:domain rdf:resource="#Offering"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This specifies the BusinessFunction of the Offering, i.e. whether the
    BusinessEntity is offering to sell, to lease, or to repair the particular type
    of product.
```

Note: While it is possible that an entity is offering multiple types of business functions, this should usually not be stated by multiple statements attached to the same offering, since the UnitPriceSpecification for the varying BusinessFunctions will usually be very different.</rdfs:comment>

```
<rdfs:range rdf:resource="#BusinessFunction"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasMakeAndModel">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#ActualProductOrServiceInstance"/>
        <owl:Class rdf:about="#ProductOrServicesSomeInstancesPlaceholder"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="#ProductOrServiceModel"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This states that an actual product instance
    (ActualProductOrServiceInstance) or a placeholder instance for multiple,
    unidentified such instances (represented by an instance of
    ProductOrServicesSomeInstancesPlaceholder) is one occurrence of a particular
    Product or Service Model.
```

```
Example: myFordT hasMakeAndModel FordT</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="appliesToPaymentMethod">
  <rdfs:domain rdf:resource="#PaymentChargeSpecification"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This property specifies the Payment Method to which the Payment Charge
    Specification applies.</rdfs:comment>
  <rdfs:range rdf:resource="#PaymentMethod"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="typeOfGood">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This specifies the type of Product or Service the TypeAndQuantityNode is
    referring to.</rdfs:comment>
  <rdfs:domain rdf:resource="#TypeAndQuantityNode"/>
  <rdfs:range rdf:resource="#ProductOrService"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPriceSpecification">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This links an Offering to one or more PriceSpecifications. There can be
    UnitPriceSpecifications, Payment Charge Specifications, and Delivery Charge
    Specifications. For each type
    multiple PriceSpecifications for the same Offering are possible, e.g. for
    different quantity ranges or for different currencies, or for different
    combinations of Delivery Method and target destination.</rdfs:comment>
  <rdfs:range rdf:resource="#PriceSpecification"/>
  <rdfs:domain rdf:resource="#Offering"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOpeningHoursDayOfWeek">
  <rdfs:range rdf:resource="#DayOfWeek"/>
  <rdfs:domain rdf:resource="#OpeningHoursSpecification"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This specifies the DayOfWeek to which the OpeningHoursSpecification is
    related.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="appliesToDeliveryMethod">
  <rdfs:range rdf:resource="#DeliveryMethod"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This property specifies the Delivery Method to which the Delivery Charge
    Specification applies.</rdfs:comment>
  <rdfs:domain rdf:resource="#DeliveryChargeSpecification"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="includesObject">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This object property links an Offering to one or multiple
    TypeAndQuantityNode that specify the components that are included in the
    respective offer.</rdfs:comment>
  <rdfs:range rdf:resource="#TypeAndQuantityNode"/>
```


SEBIS Technical Report

```
<rdfs:domain rdf:resource="#Offering"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasWarrantyScope">
  <rdfs:domain rdf:resource="#WarrantyPromise"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This states the WarrantyScope of a given WarrantyPromise.</rdfs:comment>
  <rdfs:range rdf:resource="#WarrantyScope"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="availableDeliveryMethods">
  <rdfs:range rdf:resource="#DeliveryMethod"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This specifies the DeliveryMethods available for a given
Offering.</rdfs:comment>
  <rdfs:domain rdf:resource="#Offering"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="eligibleCustomerTypes">
  <rdfs:range rdf:resource="#BusinessEntityType"/>
  <rdfs:domain rdf:resource="#Offering"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The types of customers (CustomerType) for which the given Offering is
valid.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEligibleQuantity">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This specifies the interval and unit of measurement of ordering
quantities for which the PriceSpecification is valid. This allows e.g.
specifying that a certain freight charge is valid only for a certain quantity.
Note that if an offering is a bundle, i.e. it consists of more than one unit
of a single type of good, or if the unit of measurement for the good is
different from unit (Common Code C62), then hasEligible Quantity refers to
units of this bundle.</rdfs:comment>
  <rdfs:domain rdf:resource="#PriceSpecification"/>
  <rdfs:range rdf:resource="#QuantitativeValue"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasWarrantyPromise">
  <rdfs:range rdf:resource="#WarrantyPromise"/>
  <rdfs:domain rdf:resource="#Offering"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This specified the WarrantyPromise made by the BusinessEntity for the
given Offering.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="quantitativeProductOrServiceProperty">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This is the super property of all quantitative properties for products
and services. All eclassOWL properties that specify quantitative
characteristics, for which an interval is at least theoretically an
appropriate value, are specified are subproperties of this
property.</rdfs:comment>
  <rdfs:domain rdf:resource="#ProductOrService"/>
  <rdfs:range rdf:resource="#QuantitativeValue"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="offers">
  <rdfs:domain rdf:resource="#BusinessEntity"/>
  <rdfs:range rdf:resource="#Offering"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This links a BusinessEntity to the respective Offering.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isSimilarTo">
  <rdfs:range rdf:resource="#ProductOrService"/>
  <rdfs:domain rdf:resource="#ProductOrService"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This states that a given ProductOrService is similar to another
ProductOrService. Of course, this is a subjective statement; when interpreting
it, the trust in the origin of the statement should be taken into
account.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="acceptedPaymentMethods">
  <rdfs:range rdf:resource="#PaymentMethod"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The PaymentMethods accepted by the BusinessEntity for the given
Offering.</rdfs:comment>
```

SEBIS Technical Report

```
<rdfs:domain rdf:resource="#Offering"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="qualitativeProductOrServiceProperty">
  <rdfs:domain rdf:resource="#ProductOrService"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This is the super property of all qualitative properties for products and
    services. All eclassOWL properties for which QualitativeValue instances are
    specified are subproperties of this property.</rdfs:comment>
  <rdfs:range rdf:resource="#QualitativeValue"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isConsumableFor">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This states that a particular ProductOrService is a consumable for
    another ProductOrService.</rdfs:comment>
  <rdfs:domain rdf:resource="#ProductOrService"/>
  <rdfs:range rdf:resource="#ProductOrService"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="availableAtOrFrom">
  <rdfs:range rdf:resource="#LocationOfSalesOrServiceProvisioning"/>
  <rdfs:domain rdf:resource="#Offering"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This states that a particular Offering is available at or from the given
    LocationOfSalesOrServiceProvisioning (e.g. shop or branch).</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isAccessoryOrSparePartFor">
  <rdfs:range rdf:resource="#ProductOrService"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This states that a particular ProductOrService is an accessory or spare
    part for another ProductOrService.</rdfs:comment>
  <rdfs:domain rdf:resource="#ProductOrService"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="durationOfWarrantyInMonths">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#WarrantyPromise"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property specifies the duration of the WarrantyPromise in
    months.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasCurrencyValue">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property specifies the amount of money for a price per unit,
    shipping charges, or payment charges. The currency and other relevant details
    are attached to the respective PriceSpecification etc.
    For a Unit Price Specification, this is the price for one unit (as specified
    in the unit of measurement of the UnitPriceSpecification) of the respective
    ProductOrService. For a Delivery Charge Specification or a Payment Charge
    Specification, it is the price per delivery or payment.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#PriceSpecification"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasMinValueInteger">
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasMinValue"/>
  </rdfs:subPropertyOf>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property captures the lower limit of a QuantitativeValueInteger
    instance.</rdfs:comment>
  <rdfs:domain rdf:resource="#QuantitativeValueInteger"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasValueFloat">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This subproperty specifies that the upper and lower limit of the given
    QuantitativeValueFloat are identical and have the respective float value. It
    is a shortcut for such cases where a quantitative property is (at least
    practically) a single value and not an interval.</rdfs:comment>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasMaxValueFloat"/>
  </rdfs:subPropertyOf>
  <rdfs:subPropertyOf>
    <owl:DatatypeProperty rdf:ID="hasMinValueFloat"/>
  </rdfs:subPropertyOf>
```

```

    </rdfs:subPropertyOf>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasCurrency">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The currency for all prices in the PriceSpecification given using the ISO
4217 standard (3 characters).</rdfs:comment>
    <rdfs:domain rdf:resource="#PriceSpecification"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="hasMaxValueInteger">
    <rdfs:domain rdf:resource="#QuantitativeValueInteger"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property captures the upper limit of a QuantitativeValueInteger
instance.</rdfs:comment>
    <rdfs:subPropertyOf>
      <owl:DatatypeProperty rdf:ID="hasMaxValue"/>
    </rdfs:subPropertyOf>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#hasMinValueFloat">
    <rdfs:subPropertyOf>
      <owl:DatatypeProperty rdf:about="#hasMinValue"/>
    </rdfs:subPropertyOf>
    <rdfs:domain rdf:resource="#QuantitativeValueFloat"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property captures the lower limit of a QuantitativeValueFloat
instance.</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#hasMaxValue">
    <rdfs:domain rdf:resource="#QuantitativeValue"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property captures the upper limit of a QuantitativeValue
instance.</rdfs:comment>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#hasMaxValueFloat">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#QuantitativeValueFloat"/>
    <rdfs:subPropertyOf rdf:resource="#hasMaxValue"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property captures the upper limit of a QuantitativeValueFloat
instance.</rdfs:comment>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="validThrough">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property specifies the end of the validity of the Offering.
The point in time must be in GMT.</rdfs:comment>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Offering"/>
          <owl:Class rdf:about="#UnitPriceSpecification"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="closes">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The closing hour of the LocationOfSalesOrServiceProvisioning on the
given DayOfWeek given in the local time valid at the Location.</rdfs:comment>
    <rdfs:domain rdf:resource="#OpeningHoursSpecification"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#time"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="eligibleRegions">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property specifies the geo-political region or regions for which the
offer is valid using the two-character version of ISO 3166-1 (ISO 3166-1
alpha-2) for regions or ISO 3166-2 , which breaks down the countries from ISO
3166-1 into administrative subdivisions.

```

SEBIS Technical Report

Important: Do NOT use 3-letter ISO 3166-1 codes!</rdfs:comment>

```
<rdfs:domain>
  <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Offering"/>
      <owl:Class rdf:about="#DeliveryChargeSpecification"/>
    </owl:unionOf>
  </owl:Class>
</rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="valueAddedTaxIncluded">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdfs:domain rdf:resource="#PriceSpecification"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This property specifies whether the applicable value-added tax (VAT) is
    included in the prices of the PriceSpecification or not. It determines this
    feature for all types of PriceSpecifications, i.e. UnitPriceSpecifications,
    DeliveryChargeSpecifications, and PaymentChargeSpecifications.
  </rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="opens">
  <rdfs:domain rdf:resource="#OpeningHoursSpecification"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#time"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >The opening hour of the LocationOfSalesOrServiceProvisioning on the given
    DayOfWeek given in the local time valid at the Location.</rdfs:comment>
  </owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="legalName">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >The legal name of the business entity.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#BusinessEntity"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasValueInteger">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This subproperty specifies that the upper and lower limit of the given
    QuantitativeValueInteger are identical and have the respective integer value.
    It is a shortcut for such cases where a quantitative property is (at least
    practically) a single value and not an interval.</rdfs:comment>
  <rdfs:subPropertyOf rdf:resource="#hasMaxValueInteger"/>
  <rdfs:subPropertyOf rdf:resource="#hasMinValueInteger"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasUnitOfMeasurement">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >The unit of measurement for a QuantitativeValue, a Unit Price
    Specification, or a TypeAndQuantityNode given using the UN/CEFACT Common Code
    (3 characters).</rdfs:comment>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#QuantitativeValue"/>
        <owl:Class rdf:about="#UnitPriceSpecification"/>
        <owl:Class rdf:about="#TypeAndQuantityNode"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="validFrom">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    >This property specifies the beginning of the validity of the Offering.
    The point in time must be in GMT.</rdfs:comment>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Offering"/>
        <owl:Class rdf:about="#UnitPriceSpecification"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
```

```

    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="amountOfThisGood">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#TypeAndQuantityNode"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property specifies the quantity of the goods included in the
    Offering via this TypeAndQuantityNode. The quantity is given in the
    UnitOfMeasurement attached to the TypeAndQuantityNode.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="datatypeProductOrServiceProperty">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property is the super property for all pure datatype properties that
    can be used to describe a product and services instance, or via the instances
    placeholders, of a set of instances of mass-produces commodities.

```

Only such eClassOWL properties that are no quantitative properties and that have no predefined QualitativeValue instances are subproperties of this property. In practice, this refers to a few integer properties for which the integer value represents qualitative aspects, for string datatypes (as long as no predefined values exist), and for boolean datatype properties.</rdfs:comment>

```

  <rdfs:domain rdf:resource="#ProductOrService"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="description">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#ProductOrService"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >A short textual description of the product or service. This can be easily
    extracted by search engines and other applications.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="#hasMinValue">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This property captures the lower limit of a QuantitativeValue
    instance.</rdfs:comment>
  <rdfs:domain rdf:resource="#QuantitativeValue"/>
</owl:DatatypeProperty>
<BusinessFunction rdf:ID="Maintain">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This Business Function indicates that the Business Entity offers to carry
    out typical maintenance tasks for the specified Product. Maintenance tasks are
    actions that undo or compensate for wear or other deterioration caused by
    regular usage, in order to restore the originally intended function of the
    product, or to prevent outage or malfunction. In other words, the Business
    Entity is usually able and willing to maintain an object x if x is an instance
    of the given class of product.</rdfs:comment>
</BusinessFunction>
<DeliveryModeParcelService rdf:ID="FederalExpress">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Delivery via the parcel service Federal Express.</rdfs:comment>
</DeliveryModeParcelService>
<BusinessFunction rdf:ID="LeaseOut">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This Business Function indicates that the Business Entity offers to
    temporarily grant the right to use the specified Product.</rdfs:comment>
</BusinessFunction>
<DayOfWeek rdf:ID="Saturday">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Saturday as a day of the week.</rdfs:comment>
</DayOfWeek>
<PaymentMethod rdf:ID="ByBankTransferInAdvance">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Payment by bank transfer in advance, i.e.m the offering Business Entity
    will inform the buying party about their bank account details and will
    deliver the goods upon receipt of the due amount.</rdfs:comment>
</PaymentMethod>
<PaymentMethodCreditCard rdf:ID="DinersClub">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"

```

SEBIS Technical Report

```
>Payment by credit cards issued by the Diner's Club
network.</rdfs:comment>
</PaymentMethodCreditCard>
<BusinessFunction rdf:ID="Dispose">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This Business Function indicates that the Business Entity offers to
    accept the specified Product for proper disposal, recycling, or any other kind
    of allowed usages, freeing the current owner from all rights and
    obligations.</rdfs:comment>
</BusinessFunction>
<WarrantyScope rdf:ID="Labor-BringIn">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >In case of a defect or malfunction, the buying party has the right to
    transport the good to a service location determined by the the selling
    Business Entity and will be charged only for parts and materials needed to fix
    the problem. Labor will be covered by the selling Business Entity or one of
    its partnering Business Entities.
```

Note: This is just a rough classification for filtering offers. It is up to the buying party to check the exact scope and terms and conditions of the Warranty Promise.</rdfs:comment>

```
</WarrantyScope>
<DayOfWeek rdf:ID="Friday">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Friday as a day of the week.</rdfs:comment>
</DayOfWeek>
<PaymentMethod rdf:ID="Cash">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Payment by cash upon delivery or pickup.</rdfs:comment>
</PaymentMethod>
<DayOfWeek rdf:ID="Wednesday">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Wednesday as a day of the week.</rdfs:comment>
</DayOfWeek>
<DayOfWeek rdf:ID="Tuesday">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Tuesday as a day of the week.</rdfs:comment>
</DayOfWeek>
<DayOfWeek rdf:ID="Thursday">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Thursday as a day of the week.</rdfs:comment>
</DayOfWeek>
<BusinessFunction rdf:ID="Buy">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This Business Function indicates that the Business Entity is in general
    interested in purchasing the specified Product.</rdfs:comment>
</BusinessFunction>
<DayOfWeek rdf:ID="Monday">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Monday as a day of the week.</rdfs:comment>
</DayOfWeek>
<DeliveryModeParcelService rdf:ID="DHL">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Delivery via the parcel service DHL.</rdfs:comment>
</DeliveryModeParcelService>
<BusinessEntityType rdf:ID="PublicInstitution">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The BusinessEntityType representing such agents that are part of the
    administration or owned by the public.</rdfs:comment>
</BusinessEntityType>
<PaymentMethod rdf:ID="ByInvoice">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Payment by bank transfer after delivery, i.e. the offering Business
    Entity will deliver first, inform the buying party about the due amount and
    their bank account details, and expect payment shortly after
    delivery.</rdfs:comment>
</PaymentMethod>
<BusinessFunction rdf:ID="Repair">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This Business Function indicates that the Business Entity offers to try
    to evaluate the chances for repairing, and, if positive, repair the specified
```

SEBIS Technical Report

Product. Repairing means actions that restore the originally intended function of a product that suffers from outage or malfunction.

In other words, the Business Entity is usually able and willing to repair an object *x* if *x* is an instance of the given class of product.

```
</BusinessFunction>
<DeliveryMethod rdf:ID="DeliveryModeDirectDownload">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Delivery of the goods via direct download from the Internet, i.e., the
    offering Business Entity provides the buying party with details on how to
    retrieve the goods online. Connection fees and other costs of using the
    infrastructure are to be carried by the buying party.</rdfs:comment>
</DeliveryMethod>
<DeliveryModeParcelService rdf:ID="UPS">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Delivery via the parcel service UPS.</rdfs:comment>
</DeliveryModeParcelService>
<PaymentMethodCreditCard rdf:ID="MasterCard">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Payment by credit cards issued by the MasterCard network.</rdfs:comment>
</PaymentMethodCreditCard>
<BusinessFunction rdf:ID="Sell">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This Business Function indicates that the Business Entity offers to
    transfer permanently all property rights on the specified
    Product.</rdfs:comment>
</BusinessFunction>
<PaymentMethodCreditCard rdf:ID="VISA">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Payment by credit cards issued by the VISA network.</rdfs:comment>
</PaymentMethodCreditCard>
<PaymentMethodCreditCard rdf:ID="AmericanExpress">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Payment by credit cards issued by the American Express
    network.</rdfs:comment>
</PaymentMethodCreditCard>
<BusinessFunction rdf:ID="ProvideService">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This Business Function indicates that the Business Entity offers to
    provide the type of Service.
```

Note: Maintain and Repair are also types of Services. However, eClassOWL and other ontologies provide classes for tangible products as well as for types of services. The business function Provide Service is to be used with such goods that are Services, while Maintain and Repair can be used with goods for which the class of product exists in the ontology, but not the respective type of service.

Example: Car maintenance could be expressed both as *Provide Service Car Maintenance* or *Maintain Cars*. Since existing ontologies for goods often tangle products and services, it seems beneficial to include Provide Service as a business function.

```
</BusinessFunction>
<BusinessEntityType rdf:ID="Business">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The BusinessEntityType representing such agents that are themselves
    offering commercial services or products on the market. Usually, businesses
    are characterized that they are officially registered with the public
    administration and strive for profits by their activities.</rdfs:comment>
</BusinessEntityType>
<DayOfWeek rdf:ID="Sunday">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Sunday as a day of the week.</rdfs:comment>
</DayOfWeek>
<BusinessEntityType rdf:ID="Reseller">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The BusinessEntityType representing such agents that are purchasing the
    scope of products included in the Offering for resale on the market. Resellers
    are also businesses, i.e. they are officially registered with the public
    administration and strive for profits by their activities.</rdfs:comment>
</BusinessEntityType>
<WarrantyScope rdf:ID="PartsAndLabor-PickUp">
```

SEBIS Technical Report

```
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>In case of a defect or malfunction, the buying party has the request from
the selling Business Entity to pick-up the good from its current location to a
suitable service location, where the functionality of the good will be
restored. All labor, parts, and materials needed to fix the problem will be
covered by the selling Business Entity or one of its partnering Business
Entities.</rdfs:comment>
</WarrantyScope>
<DeliveryMethod rdf:ID="DeliveryModeMail">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Delivery via regular mail service (private or public postal
services).</rdfs:comment>
</DeliveryMethod>
<BusinessEntityType rdf:ID="Enduser">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >The BusinessEntityType representing such agents that are purchasing the
good or service for private consumption, in particular not for resale or for
usage within an industrial enterprise. By default, a BusinessEntity is an
Enduser.</rdfs:comment>
</BusinessEntityType>
<WarrantyScope rdf:ID="PartsAndLabor-BringIn">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >In case of a defect or malfunction, the buying party has the right to
transport the good to a service location determined by the the selling
Business Entity and will not be be charged for labor, parts, and materials
needed to fix the problem. All those costs will be covered by the selling
Business Entity or one of its partnering Business Entities.

Note: This is just a rough classification for filtering offers. It is up to
the buying party to check the exact scope and terms and conditions of the
Warranty Promise.</rdfs:comment>
</WarrantyScope>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)
http://protege.stanford.edu -->
```



```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:ex="http://www.domain2.com#"
  xmlns="http://www.heppnetz.de/ontologies/goodrelations/examples#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:gr="http://www.heppnetz.de/ontologies/goodrelations/v1#"
  xml:base="http://www.heppnetz.de/ontologies/goodrelations/examples">
  <owl:Ontology rdf:about="">
    <owl:imports
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1"/>
    <owl:imports
rdf:resource="http://protege.stanford.edu/plugins/owl/protege"/>
  </owl:Ontology>
  <owl:Class rdf:ID="Piano">
    <rdfs:subClassOf
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrService"/>
  </owl:Class>
  <owl:Class rdf:ID="Cellphone">
    <rdfs:subClassOf
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrService"/>
  </owl:Class>
  <owl:Class rdf:ID="Battery">
    <rdfs:subClassOf
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrService"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="hasWeight">
    <rdfs:subPropertyOf
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#quantitativeProductOrServiceProperty"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasTalkTime">
    <rdfs:subPropertyOf
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#quantitativeProductOrServiceProperty"/>
  </owl:ObjectProperty>
  <gr:BusinessEntity rdf:ID="PeterMiller">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      >Peter Miller</rdfs:comment>
    <rdfs:seeAlso rdf:resource="http://www.peter-millers-shop.com"/>
    <gr:legalName rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      >Peter Miller's Shop</gr:legalName>
  </gr:BusinessEntity>
  <Cellphone rdf:ID="AnonymousCellphoneInstancesOfTypeSony_sl234">
    <rdf:type
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrServiceSomeInstancesPlaceholder"/>
    <gr:hasMakeAndModel>
      <Cellphone rdf:ID="SonyCellPhoneModel_sl234">
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          >Sony cellphone model sl234</rdfs:comment>
        <rdfs:seeAlso rdf:resource="http://www.sony.com/cellphones/sl234"/>
      <rdf:type
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrServiceModel"/>
        <hasTalkTime>
          <gr:QuantitativeValueInteger rdf:ID="QuantitativeValueInteger_9">
            <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              ></gr:hasUnitOfMeasurement>

```

```

    <gr:hasMaxValueInteger
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</gr:hasMaxValueInteger>
    <gr:hasMinValueInteger
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</gr:hasMinValueInteger>
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The node representing a time duration of 120
minutes.</rdfs:comment>
    </gr:QuantitativeValueInteger>
  </hasTalkTime>
  <hasWeight>
    <gr:QuantitativeValueFloat rdf:ID="QuantitativeValueFloat_10">
    <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></gr:hasUnitOfMeasurement>
    <gr:hasMaxValueFloat
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >100.0</gr:hasMaxValueFloat>
    <gr:hasMinValueFloat
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >100.0</gr:hasMinValueFloat>
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >The value node representing a weight of 100 grams</rdfs:comment>
    </gr:QuantitativeValueFloat>
  </hasWeight>
</Cellphone>
</gr:hasMakeAndModel>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >This node represents all of the anonymous cellphone instances of Sony
s1234 which Peter Miller is willing to try to repair.</rdfs:comment>
</Cellphone>
<gr:Offering rdf:ID="MillersOfferingEbay">
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-12-01T00:00:00Z</gr:validFrom>
  <gr:hasBusinessFunction
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Sell"/>
  <gr:validThrough rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
  >2007-12-31T23:59:59Z</gr:validThrough>
  <rdfs:seeAlso rdf:resource="http://www.ebay.com/auction1234/">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Peter Miller's Offering to sell his cellphone on eBay.</rdfs:comment>
  <gr:includesObject>
    <gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_PeterEbay">
      <gr:typeOfGood>
        <gr:ActualProductOrServiceInstance rdf:ID="PetersUsedCellphone">
          <gr:hasMakeAndModel rdf:resource="#SonyCellPhoneModel_s1234"/>
          <rdfs:type rdf:resource="#Cellphone"/>
        </gr:ActualProductOrServiceInstance>
      </gr:typeOfGood>
      <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >C62</gr:hasUnitOfMeasurement>
      <gr:amountOfThisGood
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >1.0</gr:amountOfThisGood>
    </gr:TypeAndQuantityNode>
  </gr:includesObject>
</gr:Offering>
  <gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_Amazon1">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >This node represents that the Amazon offering includes two
batteries.</rdfs:comment>
    <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >C62</gr:hasUnitOfMeasurement>
    <gr:typeOfGood>
      <gr:ProductOrServicesSomeInstancesPlaceholder
rdf:ID="CellPhoneBattery_InstancePlaceholder">

```

SEBIS Technical Report

```

    <rdf:type rdf:resource="#Battery"/>
  </gr:ProductOrServicesSomeInstancesPlaceholder>
</gr:typeOfGood>
<gr:amountOfThisGood rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>2.0</gr:amountOfThisGood>
</gr:TypeAndQuantityNode>
<gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_Sony">
  <gr:typeOfGood>
    <Cellphone rdf:ID="Cellphone_15">
      <gr:hasMakeAndModel rdf:resource="#SonyCellPhoneModel_s1234"/>
      <rdf:type
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ProductOrServ
icesSomeInstancesPlaceholder"/>
    </Cellphone>
  </gr:typeOfGood>
  <gr:amountOfThisGood rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>1.0</gr:amountOfThisGood>
  <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>C62</gr:hasUnitOfMeasurement>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>The node representing the fact that the general Sony offer refers to one
cellphone. C62 is the common code for "unit".</rdfs:comment>
</gr:TypeAndQuantityNode>
<gr:BusinessEntity rdf:ID="Amazon">
  <gr:legalName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Amazon Inc.</gr:legalName>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Amazon</rdfs:comment>
  <rdfs:seeAlso rdf:resource="http://www.amazon.com"/>
</gr:BusinessEntity>
<gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_MillersRepair">
  <gr:amountOfThisGood rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>1.0</gr:amountOfThisGood>
  <gr:typeOfGood
rdf:resource="#AnonymousCellphoneInstancesOfTypeSony_s1234"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>The node that represents that Peter Miller's offer to repair Sony s1234
cellphones refers to 1 cellphone. This node may become significant in
combination with Unit Price Specifications.</rdfs:comment>
  <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>C62</gr:hasUnitOfMeasurement>
</gr:TypeAndQuantityNode>
<gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_Amazon2">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>This instance reflects that the Amazon offering includes 1 unit (Code 62)
of the instances placeholder Cellphone_3.</rdfs:comment>
  <gr:amountOfThisGood rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>1.0</gr:amountOfThisGood>
  <gr:typeOfGood>
    <gr:ProductOrServicesSomeInstancesPlaceholder rdf:ID="Cellphone_3">
      <gr:hasMakeAndModel rdf:resource="#SonyCellPhoneModel_s1234"/>
      <rdf:type rdf:resource="#Cellphone"/>
    </gr:ProductOrServicesSomeInstancesPlaceholder>
  </gr:typeOfGood>
  <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>C62</gr:hasUnitOfMeasurement>
</gr:TypeAndQuantityNode>
<gr:Offering rdf:ID="SonyOffering_s1234_phones">
  <gr:validThrough rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-12-31T23:59:59Z</gr:validThrough>
  <rdfs:seeAlso rdf:resource="http://www.sony.com/cellphones/s1234"/>
  <gr:hasBusinessFunction
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Sell"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>The general Sony offer to sell s1234s</rdfs:comment>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_Sony"/>
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-01-01T00:00:00Z</gr:validFrom>

```

```

</gr:Offering>
<gr:Offering rdf:ID="MillersOfferToRepair">
  <gr:hasBusinessFunction
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Repair"/>
  <rdfs:seeAlso rdf:resource="http://www.peter-millers-shop.com/service/" />
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-12-01T00:00:00Z</gr:validFrom>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Peter Miller's Offering to repair Sony s1234 cellphones.</rdfs:comment>
  <gr:validThrough rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-12-31T23:59:59Z</gr:validThrough>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_MillersRepair"/>
</gr:Offering>
<gr:Offering rdf:ID="AmazonOfferingABundle">
  <gr:validThrough rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-12-31T23:59:59Z</gr:validThrough>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Amazon is offering a bundle, composed of s1234 phones and two
batteries.</rdfs:comment>
  <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
>2007-01-01T00:00:00Z</gr:validFrom>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_Amazon1"/>
  <gr:includesObject rdf:resource="#TypeAndQuantityNode_Amazon2"/>
  <rdfs:seeAlso rdf:resource="http://www.amazon.com/cellphones/" />
  <gr:hasBusinessFunction
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Sell"/>
</gr:Offering>
<gr:BusinessEntity rdf:ID="Sony">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Sony AG</rdfs:comment>
  <gr:legalName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Sony AG</gr:legalName>
  <rdfs:seeAlso rdf:resource="http://www.sony.com"/>
</gr:BusinessEntity>
<gr:Offering rdf:ID="MillersOfferingToRepairAnyCellPhone">
  <gr:hasBusinessFunction
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Repair"/>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Peter Miller promises to repair cellphones that weigh between 10 and 120
grams.</rdfs:comment>
  <gr:includesObject>
    <gr:TypeAndQuantityNode rdf:ID="TypeAndQuantityNode_Miller-
CellPhoneRepair">
      <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>This node represents that Peter Miller's offer to repair refers to
one unit of a cellphone.</rdfs:comment>
      <gr:typeOfGood>
        <gr:ProductOrServicesSomeInstancesPlaceholder rdf:ID="Cellphone_10-
AnyCellphoneThatWeighs10-120grams">
          <hasWeight>
            <gr:QuantitativeValueFloat rdf:ID="QuantitativeValueFloat_11-
WeightBetween10_and_120Gram">
              <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>This value object represents weights between 10 and 120
grams.</rdfs:comment>
              <gr:hasMaxValueFloat
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>120.0</gr:hasMaxValueFloat>
              <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>GRM</gr:hasUnitOfMeasurement>
              <gr:hasMinValueFloat
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>10.0</gr:hasMinValueFloat>
            </gr:QuantitativeValueFloat>
          </hasWeight>
          <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>This entity represents anonymous instances of cellphones that
weigh between 10 and 120 grams.</rdfs:comment>

```

SEBIS Technical Report

```
<rdf:type
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/examples#Cellpho
ne"/>
    </gr:ProductOrServicesSomeInstancesPlaceholder>
    </gr:typeOfGood>
    <gr:amountOfThisGood
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.0</gr:amountOfThisGood>
    <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >C62</gr:hasUnitOfMeasurement>
    </gr:TypeAndQuantityNode>
    </gr:includesObject>
    </gr:Offering>
    <rdf:Description
rdf:about="http://www.heppnetz.de/ontologies/goodrelations/examples#MillersOff
erToRepair">
        <gr:acceptedPaymentMethods
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ByBankTransfe
rInAdvance"/>
        <gr:hasWarrantyPromise>
        <gr:WarrantyPromise rdf:ID="PeterMillersWarrantyPromise">
            <gr:hasWarrantyScope
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#PartsAndLabor
-BringIn"/>
                <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >Peter Miller grants a warranty on parts and labor for 1 month,
customer bring-in.</rdfs:comment>
                <gr:durationOfWarrantyInMonths
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
                >0</gr:durationOfWarrantyInMonths>
            </gr:WarrantyPromise>
        </gr:hasWarrantyPromise>
    </rdf:Description>
    <rdf:Description
rdf:about="http://www.heppnetz.de/ontologies/goodrelations/examples#PeterMille
r">
        <gr:offers>
        <rdf:Description
rdf:about="http://www.heppnetz.de/ontologies/goodrelations/examples#MillersOff
eringEbay">
            <gr:hasWarrantyPromise rdf:resource="#PeterMillersWarrantyPromise"/>
            <gr:hasPriceSpecification>
            <gr:UnitPriceSpecification
rdf:ID="UnitPriceSpecification_MillersCellPhone">
                <gr:hasCurrency
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >EUR</gr:hasCurrency>
                <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >C62</gr:hasUnitOfMeasurement>
                <gr:valueAddedTaxIncluded
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
                >true</gr:valueAddedTaxIncluded>
                <gr:hasCurrencyValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
                >80.0</gr:hasCurrencyValue>
                <gr:validThrough
rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
                >2007-12-31T23:59:59Z</gr:validThrough>
                <gr:validFrom
rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
                >2007-12-01T00:00:00Z</gr:validFrom>
                <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >The price specification for Peter Miller's fix-price
auction.</rdfs:comment>
            </gr:UnitPriceSpecification>
        </gr:hasPriceSpecification>
```

SEBIS Technical Report

```
<gr:acceptedPaymentMethods
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#ByBankTransfe
rInAdvance"/>
  <gr:eligibleRegions
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >IT</gr:eligibleRegions>
  <gr:eligibleRegions
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >AT</gr:eligibleRegions>
  </rdf:Description>
</gr:offers>
<gr:offers
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/examples#Millers
OfferToRepair"/>
  </rdf:Description>
  <rdf:Description
rdf:about="http://www.heppnetz.de/ontologies/goodrelations/examples#AmazonOffe
ringABundle">
    <gr:availableDeliveryMethods
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#DeliveryModeM
ail"/>
      <gr:eligibleRegions rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >AT</gr:eligibleRegions>
      <gr:eligibleRegions rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >CH</gr:eligibleRegions>
      <gr:acceptedPaymentMethods
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#MasterCard"/>
      <gr:hasPriceSpecification>
        <gr:DeliveryChargeSpecification rdf:ID="DeliveryChargeSpecification_3-
Amazon-8EUR">
          <gr:appliesToDeliveryMethod
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#DHL"/>
          <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >The specification of shipment charges for the Amazon
offering.</rdfs:comment>
          <gr:valueAddedTaxIncluded
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</gr:valueAddedTaxIncluded>
          <gr:eligibleRegions
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >CH</gr:eligibleRegions>
          <gr:eligibleRegions
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >DE</gr:eligibleRegions>
          <gr:eligibleRegions
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >AT</gr:eligibleRegions>
          <gr:hasCurrency rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >EUR</gr:hasCurrency>
          <gr:hasCurrencyValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
          >8.0</gr:hasCurrencyValue>
          </gr:DeliveryChargeSpecification>
          </gr:hasPriceSpecification>
          <gr:hasWarrantyPromise>
            <gr:WarrantyPromise rdf:ID="WarrantyPromise_6-Amazon-36months">
              <gr:durationOfWarrantyInMonths
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
              >0</gr:durationOfWarrantyInMonths>
              <gr:hasWarrantyScope
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Labor-
BringIn"/>
                <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >Amazon grants a warranty of 36 months covering labor
only.</rdfs:comment>
                </gr:WarrantyPromise>
              </gr:hasWarrantyPromise>
            <gr:availableDeliveryMethods
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#DHL"/>
            <gr:acceptedPaymentMethods
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#VISA"/>
```

```

    <gr:hasWarrantyPromise>
      <gr:WarrantyPromise rdf:ID="WarrantyPromise_5-Amazon-12months">
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Amazon grants a warranty of 12 months parts and labor, pick-
up.</rdfs:comment>
        <gr:durationOfWarrantyInMonths
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >12</gr:durationOfWarrantyInMonths>
        <gr:hasWarrantyScope
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#PartsAndLabor-PickUp"/>
      </gr:WarrantyPromise>
    </gr:hasWarrantyPromise>
    <gr:eligibleRegions rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >DE</gr:eligibleRegions>
    <gr:hasPriceSpecification>
      <gr:UnitPriceSpecification rdf:ID="UnitPriceSpecification_Amazon99">
        <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >The price specification that one unit of the bundle costs 99 Euros
including VAT.</rdfs:comment>
        <gr:validThrough
rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
          >2007-12-31T23:59:59Z</gr:validThrough>
        <gr:validFrom rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"
          >2007-12-01T00:00:00Z</gr:validFrom>
        <gr:hasCurrencyValue
rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
          >99.0</gr:hasCurrencyValue>
        <gr:hasCurrency rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >EUR</gr:hasCurrency>
        <gr:valueAddedTaxIncluded
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >true</gr:valueAddedTaxIncluded>
        <gr:hasUnitOfMeasurement
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >C62</gr:hasUnitOfMeasurement>
      </gr:UnitPriceSpecification>
    </gr:hasPriceSpecification>
  </rdf:Description>
</rdf:Description>
rdf:about="http://www.heppnetz.de/ontologies/goodrelations/examples#Sony">
  <gr:offers>
    <rdf:Description
rdf:about="http://www.heppnetz.de/ontologies/goodrelations/examples#SonyOffering_sl234_phones">
      <gr:hasWarrantyPromise>
        <gr:WarrantyPromise rdf:ID="WarrantyPromise_Sony_24_months">
          <gr:hasWarrantyScope
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#PartsAndLabor-BringIn"/>
        </gr:WarrantyPromise>
        <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >Sony grants a warranty of 24 months parts and labor, customer
bring-in.</rdfs:comment>
        <gr:durationOfWarrantyInMonths
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >0</gr:durationOfWarrantyInMonths>
      </gr:WarrantyPromise>
    </gr:hasWarrantyPromise>
    <gr:eligibleCustomerTypes
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/v1#Reseller"/>
  </rdf:Description>
</gr:offers>
</rdf:Description>
rdf:about="http://www.heppnetz.de/ontologies/goodrelations/examples#Amazon">
  <gr:offers>
    <rdf:Description
rdf:resource="http://www.heppnetz.de/ontologies/goodrelations/examples#AmazonOfferingABundle"/>
  </rdf:Description>

```

</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)
http://protege.stanford.edu -->

14 APPENDIX E: PRICING MODEL BY KELKAR, LEUKEL, AND SCHMITZ (2002)

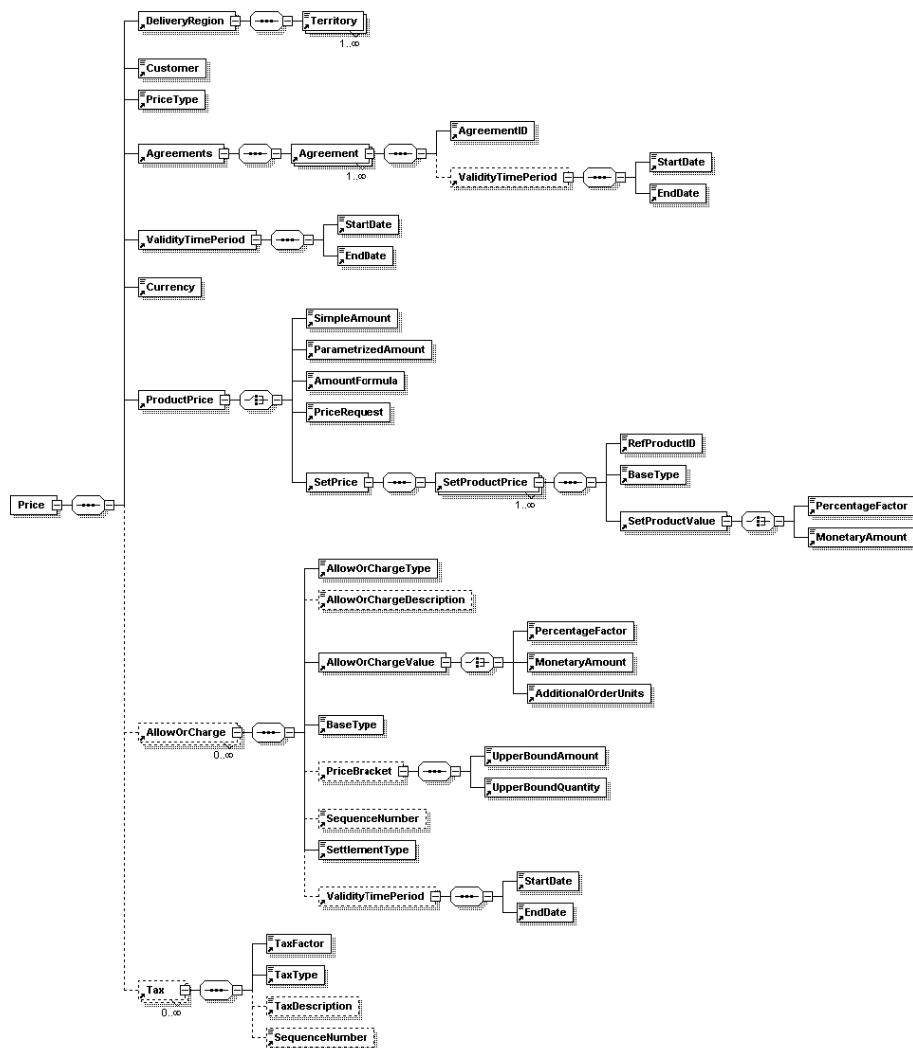


Figure 2. Derived Price Model.

Figure 8. Price model by Kelkar, Leukel, and Schmitz (2002), p. 7

Taken from (Kelkar, Leukel, & Schmitz, 2002), p. 372