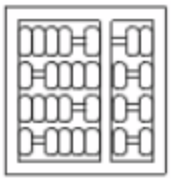


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

From Ontology Charts to Web Ontologies: Heuristics and Transformation Rules

Júlio Cesar dos Reis
Rodrigo Bonacin
Maria Cecília Calani Baranauskas

Technical Report - IC-11-02 - Relatório Técnico

January - 2011 - Janeiro

From Ontology Charts to Web Ontologies: Heuristics and Transformation Rules

Júlio Cesar dos Reis[1,2], Rodrigo Bonacin[2], M. Cecília C. Baranauskas[1]

[1] Departament of Information System at Institute of Computing
University of Campinas – UNICAMP
13083-970, Campinas, SP, Brazil

[2] Center for Information Technology Renato Archer - Rodovia Dom Pedro I, km 143,6,
13069-901, Campinas, SP, Brazil

{*julio.reis and rodrigo.bonacin*}*@cti.gov.br, cecilia@ic.unicamp.br*

Abstract

The evolution of the Web depends on novel techniques and methodologies that can handle and better represent the meanings of the huge amount of information available nowadays. Recent proposals in literature have explored new approaches based on Semiotics. The 'Semiotic Web ontology' (SWO) is an attempt to model the information in a computer-tractable and more adequate way, and, at the same time to be compatible with the Semantic Web (SW) standards. This work presents an assisted process for building SWOs. The process includes heuristics and transformation rules for deriving an initial Web ontology described in Web Ontology Language from Ontology Charts produced by the Semantic Analysis Method. Moreover, the whole process is discussed; results of the application of the process to a real context show the potential of the approach and the value of the proposed heuristics and implemented rules to create more representative Web ontologies.

Keywords: Organizational Semiotics, Semantic Analysis Method, Web Ontology Language, OWL, Semantic Web, Semiotic Web Ontology.

1 Introduction

One of the biggest challenges in the contemporary Web evolution is related to the treatment of the information meaning. There is a growing need for methods, techniques and tools that can better represent the semantic aspects of the information available in Web systems. The first initiatives taken by Berners-Lee *et al.* [1] already aimed to create a Web that also takes into account the meanings of information and not just its displaying. Nevertheless, recent studies such as Tanasescu & Streibel [19], Reis *et al.* [13] among others point out that there

are still various limitations and problems regarding technologies from the Semantic Web (SW) initiative. In the context of Computer Science, "Web ontology" can be understood as a specification of a conceptualization which provides descriptions about knowledge [7].

Even with the advent of the Web ontologies there are still no tools to assist in the organization of the information in a way suitable for human mental operations in an individual or societal way [4]. In traditional Web ontologies the concepts that are represented need to be organized in a way to produce a "knowledge tree". This tree should be able to translate a subject, representing it as accurately as possible. However, by establishing a hierarchy between concepts, it is difficult to accurately represent different contexts, which means that the ontology needs to be associated to a well-defined domain. Thus, to facilitate the work for the computer, the organization within an ontology is formally made, creating a specific relation of words.

In order to overcome such limitations and problems some approaches have been described in literature such as Gärdenfors [5]. More recently Reis *et al.* [13] proposed a new approach for the design of Web ontologies which uses the Semantic Analysis Method (SAM) [10] to better treat the representation of the information in a computer-tractable way. Reis *et al.* [13] propose the concept "Semiotic Web ontology" (SWO) which combines SAM concepts with technologies of the SW to describe computationally tractable ontologies using Web Ontology Language (OWL) [20]. According to Reis *et al.* [13], this approach enables to incorporate in SW ontologies concerns and possible representations arising from the Ontology in a Semiotic perspective, building more representative Web ontologies.

Nevertheless, in order to create the SWO heuristics and transformation rules are necessary to construct a computer-interpretable ontology from an Ontology Chart (OC) produced by SAM. Therefore, this work presents a new procedure composed by a set of heuristics and steps to build an initial model of a Web ontology described in OWL. Assuming that the Semiotic approach contributes with improvements in business modeling it is plausible to have both: the Organizational Semiotic (OS) [15] methods with a different and valuable view of the social context on one hand, and an ontology described in OWL that interoperates with what already exists on the other hand. As semantic refers to meaning, and meanings are socially created by humans, we expect to create a more faithful computer ontology considering an information system as a more abstract conceptual model that can capture the behaviour of the involved agents.

Among the main benefits are the visibility of the transformation process, and the production of an initial design model closely related to the real world concepts. Some relations between the models are mapped, and one model supports the construction of the other providing benefits presented in the different visions. Thus, in order to use the outcome of the SAM (*i.e.* OC) with languages that describe Web ontologies, it is necessary to create a procedure that makes explicit the relationships between them, and makes possible the construction of one diagram from the other.

Consequently, the main objective in this work is to show the heuristics and transformation rules created to construct SWO. This is made by an assisted process with human intervention in SONAR tool [14]. This report describes the process including the heuristics and the rules implemented in SONAR based on these heuristics, and illustrate the

application of such process in a real context situation with content of a Web system. This technical report is organized as follows: Section 2 presents the Theoretical and Methodogical Background; Section 3 describes the heuristics and transformation rules developed to create Web ontologies aided by SAM; Section 4 illustrates the process using data from a real Web system; Section 5 presents a discussion and Section 6 concludes.

2 Theoretical and Methodological Background

This section firstly presents an overview of the main concepts of the Semantic Analysis Method (SAM) that are important to understand the proposed solution; then some characteristics and properties of Web Ontology Language (OWL) are also presented.

2.1 The Semantic Analysis Method

In opposition to the objectivism, which presupposes that there exists a world independent of the observer, an objective reality composed by a structure of pre-existent entities, the SAM is based on the subjectivist paradigm [10], which understands reality as a social construction based on the behaviour of agents participating on it. The two basic axioms upon which the OS methods are founded can be expressed as: there is not knowledge without a knower; and there is not comprehension without action.

OS adopts a subjectivist philosophical stance and an agent-in-action ontology. This philosophical position states that, for all practical purposes, nothing exists without a perceiving agent or without the agent engaging in actions. That is to say, each thing depends for its existence upon the existence of its antecedents. Words and expressions we use are names for invariant patterns in the flux of actions and events which the agents experience. The classical distinction between entity, attribute and relationship disappears to be replaced by the concepts of agents, affordances (the actions or attributes of agents) and norms (for the socially defined patterns of behaviour) related to their antecedents to indicate the ontological dependency [16].

The concepts of the Semantic Analysis are represented by means of this agent-in-action ontology. Ontology in OS represents a business domain which can be described by the concepts, the ontological dependencies between the concepts, and the norms detailing the constraints at both universal and instance level [11]. The ontology in which the SAM is based assumes that the only thing that we can know is our own behaviour in our own environment. This ontology assumes that the world known to a particular agent comprises only the actions he can perform in his environment [10]. According to Stamper *et al.* [16] an agent-in-action ontology can bring benefits since it enables us to handle different meanings for the same term, and the resulting conceptual model of an information system has not only a sound philosophical basis, but also retains semantic richness.

SAM comes from the MEASUR (Methods for Eliciting, Analyzing and Specifying Users' Requirements) [17] and is a method to elicit, analyse and represent the nature of reality by affordances an agent perceives to exist. In SAM "the world" is socially

constructed by the actions of agents, on the basis of what is offered by the physical world itself. The SAM assists users or problem-owners in eliciting and representing their requirements and meanings in a formal and precise model. In SAM, the analyst in the role of facilitator specifies the required system functions in an Ontology Chart (OC) - a graphic representation of a conceptual model. The OC maps the vocabulary and the temporal relationships between the percepts that those words represent [16] and describes a view of responsible agents in the focal domain and their pattern of behavior named affordances [10]. OC can be constructed following practical rules as described by [16, 21]. Some basic concepts of SAM adopted in this work are based on Liu [10] and are described as follows:

– **Affordance**, the concept introduced by Gibson [6] can be used to express the invariant repertories of behaviour of an organism made available by some combined structure of the organism and its environment. In SAM [17] the concept introduced by Gibson was extended by Stamper to include invariants of behavior in the social world; affordances are social constructs in a certain social context [10]. The social world acts as the environment that is constantly affecting the agents' behaviour at the same time that it is affected by the agents' actions.

– **Agent** is a special kind of affordance, which can be defined as something that has responsible behaviour. Agents are affordances that can take responsibility both for their own actions and the actions of others. An agent can be an individual person, a cultural group, a language community, a society, etc. (an employee, a department, an organization, etc.);

– **Ontological dependency** is formed when an affordance is only possible if certain other affordances are available. We say that the affordance "A" is ontological dependent on the affordance "B" to mean that "A" exists only when "B" does; *e.g.*: for a person to be able to stumble, he/she must first walk; for two people to divorce, they need to be married; thus there exist an ontological dependency between to stumble and to walk, and also between divorce and marriage.

– **Determiners** are properties which are variants of quality and quantity that differentiate one instance from another. Determiner are attributes that enable one to describe an agent or an affordance;

– **Specialization**, agents and affordances can be placed in generic-specific structures according to whether or not they possess shared or different properties;

– **Whole-part**, an agent or affordance can be part of other agent or affordance. The part also owns all the ontological dependencies from the whole;

– **Role-Name,** an agent can have a specific role depending on an affordance it has.

The SAM concepts are represented by means of an OC.

Table 1: Graphical representation of some SAM concepts

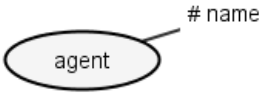
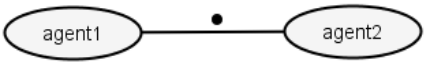
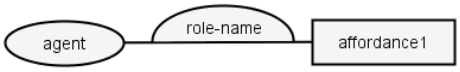
Representation	Concept
	**Affordance**
	**Agent**
	**Determiner**
	**Specialization** can be used in agents, affordances and in role-name
	**Ontological dependency** between an affordance and an agent. It means that the affordance exists while the agent exists
	**Ontological dependency** between two affordances. It means that the affordance2 exists while the affordance1 exists
	**Whole-part relation** – the agent2 is part of the agent1. It can be used between affordances, agents and role-name
	**Role-Name** – The agent that has an affordance1 is named as a role-name

Table 1 shows this graphical representation, in which agents are represented by circles, affordances by rectangles, ontological dependencies by lines from left to right, role-names by half-curves and whole-part by lines with a black dot. Figure 1 presents an example of OC.

The SAM provides a different and independent view of the organizational model. The SAM addresses issues that are not represented in any Web ontology and it provides a different way of thinking about representing meanings if compared with traditional Web ontologies as described in the next section.

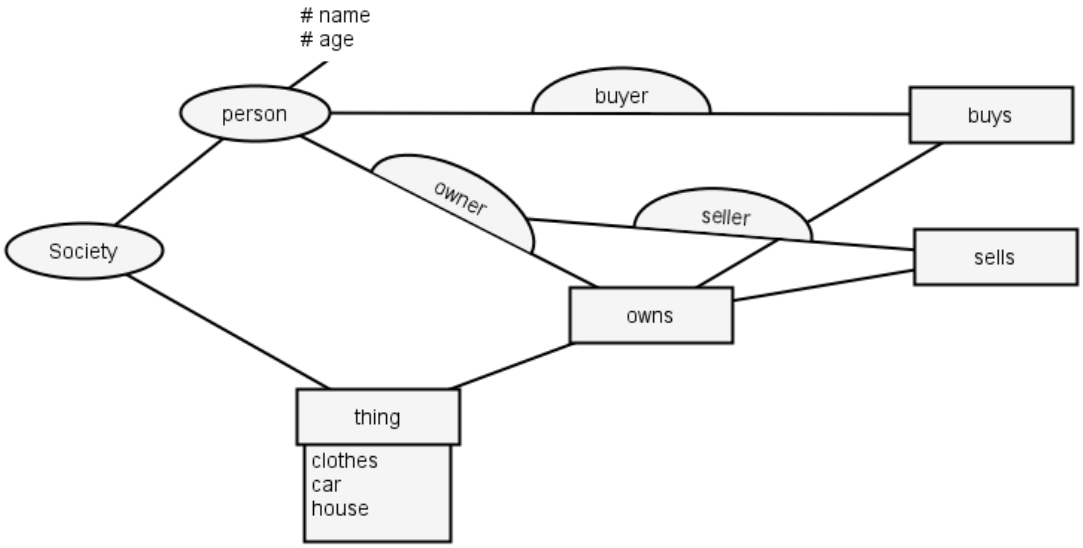


Figure 1: An example of Ontology Chart adapted from [10]

2.2 Web Ontology

Ontology has been an important concept in Philosophy, and later Library Sciences; nowadays it has become relevant to the Computer Science and in particular, the Artificial Intelligence (AI) community. In AI, an ontology is used to formally specify the concepts and relationships that characterize a certain body of knowledge (domain). The formal nature of ontologies makes them amenable to machine-readability and provides a well-defined semantics for the defined terms. This allows computer programs to manipulate, transform and draw inferences from information represented using the ontology [9]. According to Studer *et al.* [18] ontology is a shared and common understanding of some domain that can be communicated between people and computers; it is a formal specification that should be readable and understandable by machines. It is worth mentioning that the concept of Ontology from SAM is different from the one in the Web ontology domain.

The term ontology in Computer Science is often used to refer to the semantic understanding (a conceptual framework of knowledge) shared by individuals participating in a given knowledge domain. A semantic ontology may exist as an informal conceptual framework of types of concepts and their relations named and defined in natural language. Alternatively, it could be constructed as a formal semantics taking into account the domain, with the types of concepts and their relationships defined systematically in a logical language. Indeed, within the Web environment, ontology is not simply a conceptual framework, but a concrete syntactic structure that tries to model the semantics of a domain [8]. According to Noy and McGuinness [12], an ontology along with a number of different instances of its classes constitutes a knowledge base. The classes are the focus of most ontologies. Classes describe the concepts in the domain. For instance, a class of wines represents all wines; specific wines are instances of this class. The Bordeaux wine is an

instance of a class of wines. A class may have sub-classes that represent concepts that are more specific than super-classes; *e.g.* it is possible to divide the class of all wines into red, white and rosé wines. Alternatively, the class of all wines can be divided into sparkling wines and non-sparkling wines.

Web ontology is usually described by computational languages based on logic for knowledge representation and inference. These languages for ontology description are designed specifically to define ontologies. According to the SW architecture proposed by Berners-Lee *et al.* [1], the ontology description languages are related to other Web languages such as Extensible Markup Language (XML), Resource Description Framework (RDF) and RDF Schema (RDFS). In order to address interoperability problems and define a universal paradigm for web-based exchange of ontological information, the World Wide Web Consortium (W3C) created the Web Ontology Language (OWL) which became a W3C Recommendation in February of 2004 [20]. OWL is currently defined by a set of recommendations of the W3C. Using OWL as a common language, knowledge experts and application developers can create, modify, link and import ontologies in a distributed environment. OWL is an important piece of the future vision of the Web – the SW*;* Kalyanpur *et al.* [9] present a brief OWL history.

Statistics presented by Cardoso [3] show that OWL is the most common approach for modeling ontologies in software. OWL has three sub-languages with increasing expressivity: OWL Lite, OWL DL and OWL Full; each with a different intended audience based on scope and complexity of the application domain. The goal of OWL Lite is to provide a language that is viewed by tool builders as easy enough and useful enough to support, thereby acting as an entry ontology language for SW application developers; it supports classification hierarchy with simple constraints (*e.g.* it only permits cardinality values of 0 or 1), quick migration path for thesauri and other taxonomies, and lower formal complexity than OWL DL.

OWL DL supports maximum expressiveness with computational completeness. It means that all conclusions are guaranteed to be computable, finishing in finite time (decibility). It corresponds to Description Logics (DL) whereas OWL Full provides more freedom in domain modeling at the cost of a higher learning curve, with syntactic freedom of RDF with no computational guarantees. OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF [20].

According to Kalyanpur *et al.* [9] OWL is built on top of RDF, which is itself built upon the XML syntax. RDF (including RDFS – the RDF schema language) and OWL provide the capability of creating Classes, Properties, and Instances. Classes (or concepts) are general categories that can be arranged in hierarchies. Each class defines a group of individuals that share some properties. Instances (or individuals) are specific objects, and classes are used to define what types an object has. Properties (or relationships) are attributes of instances. Properties are defined generally and then used in instances to either specify data values or link to other instances. There is a specific syntax for creating classes, data and object properties, restrictions and instances. The OWL syntax can be viewed in details in W3C [20].

3 Building OWL ontologies Informed by SAM

In order to create a SWO, first the SAM must be applied in the context under study and an OC is created that will be the source model for the transformation from OC to OWL code. The SAM can be used following the orientations and phases as described by Liu [10]. According to Reis *et al.* [13] the OC is important to identify the possible agents in the context and their patterns of behaviour (affordances) including their existential relationships, and then pass these to a Web ontology described in OWL.

In order to accomplish that, a set of specific heuristics and rules were created to derive an initial OWL ontology. The proposed heuristics are based on the basic principles proposed by Liu [10], and also by the work made by Bonacin *et al.* [2]. They have proposed a set of heuristics to construct system design Unified Model Language (UML) diagrams from OC; those heuristics were adapted to our purpose since there are differences between UML and OWL. The main differences from UML to OWL are that OWL does not have some concepts from UML as methods and composition. It is important to mention that applying the heuristics and transformation rules does not guarantee an equivalent ontology in OWL; instead it represents some support to the analyst during the modeling process.

Based on these previous works, a procedure was refined and generalized in order to construct a sequence of steps into a group of heuristics that are computationally implemented by transformation rules that build ontologies described in OWL, from the results of the SAM. The produced model (OWL file) can be a valuable starting point for the design of a complete and consistent OWL ontology. The SAM addresses issues that are not represented in the traditional ontology in Computer Science, and moreover it provides a different way of thinking about the meanings when compared with the paradigm in which the traditional Web ontologies are situated. Consequently the transformation from OC to Web ontologies in OWL is not a trivial task. The following section presents it.

3.1 Heuristics to Construct Web ontology from Ontology Chart

Observing the OC is possible to note that it represents mainly the affordances and the existential relation between then. The affordances can be an agent and the relations between affordances can be as a whole-part, a specialization or an ontological dependence. Bonacin *et al.* [2] organized and divided the explanation of their heuristics based on the affordances relation, since this relation can be between affordances that are verbs or nouns. They have identified 10 different cases of relation between the affordances. In this work a different approach for explanation purpose is used; the heuristics were divided following the concepts from SAM such as: affordances, agents, determiners, role-name, whole-part, specialization and ontological dependence. Figure 2 shows an example of an OC used to apply and exemplify the approach with the proposed heuristics and transformation rules. Following the heuristics with examples using the Figure 2 are presented. It mainly involves discovering the relationship between the classes through object properties in order to construct a first version of the OWL ontology.

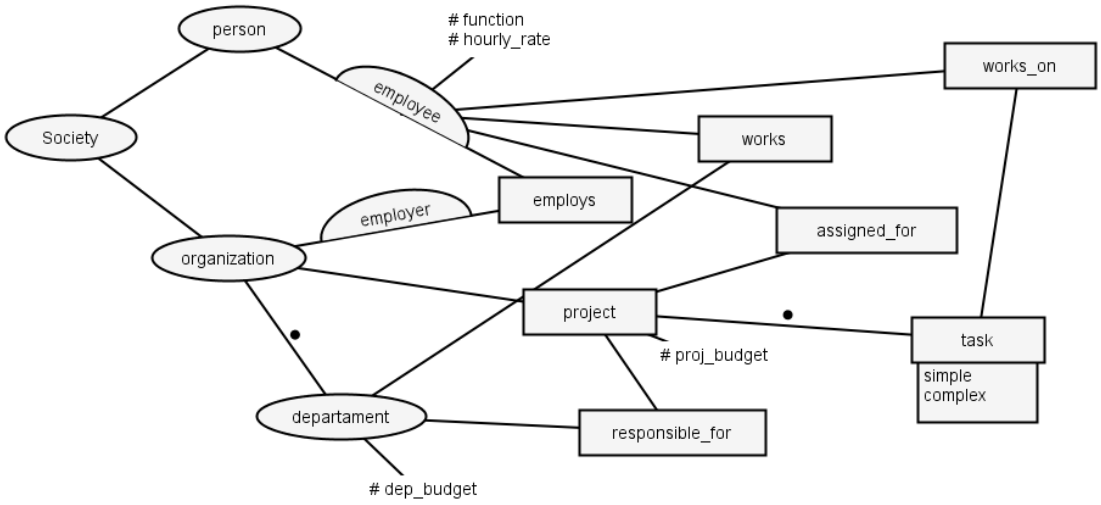


Figure 2: Ontology Chart for project management [10:79]

Affordances – During the SAM the world is mainly modeled by the identification of social constructions called affordances, while in the computational ontologies the world is modeled by the identification of classes and individuals presented in the world. The presence of an affordance in the OC suggests a class to be modeled into the OWL ontology. For instance, considering the Figure 2, a "*project*" by the SAM perspective is an affordance of the society, and by the perspective of conventional ontologies it can be a class with attributes. If the affordance named "*project*" was represented in the OC, this suggests that there is a class in the context, and probably is possible to refer it using the name "*project*". Based on the procedure of extracting name of classes from nouns, thus name of affordances that are noun can be mapped as class in OWL (*i.e.* affordances that suggests entities will be classes in OWL). However neither all affordances are nouns, for example the affordance "*employ*" which is a verb. According to Bonacin *et al.* [2] the affordances can be mapped as classes in UML or methods in classes depending on a heuristic process; for the modeling context using OWL, since there is not methods, the affordances that are verbs (*i.e.* suggests action) will be mapped as object properties in OWL. Thereby the affordance "*employ*" will not be a class in OWL, but it will be an object property named "*employ*". The details of this property, as how to set its domain and range will be better explained when considering the computational implementation of this heuristic through a transformation rule.

Agents – All the agents represented in the OC will be mapped as classes in OWL and as sub-classes of a class called "*Agent*". This is carried in order to identify the agents into the OWL ontology, so all agents from the OC would inherit the possible characteristics and properties from the class "*Agent*". Thus, the agents like "*person*", "*organization*", "*departament*" presented in Figure 2 will be classes into the OWL ontology.

Determiners – Since determiners are "like" attributes, this is directly mapped as data properties in OWL which is connected to the right appropriated class. As in the example of the Figure 2, the determiner "*function*" will be a data property in OWL and its domain will be

set with the class "*employee*"; while the determiner "*dep_budget"* will be mapped to the agent "*department*", then this data property domain will be the class "*departament*".

Role-Name – Considering the heuristics from OC to UML, the role-name can be mapped as classes or method according to the heuristic process. However, the role-name is always connected to an agent; and it is also an entity name. Hence for the context from OC to OWL, role-name will be mapped as sub-class of the class in OWL that represents the agent of the left side of the role-name. For instance, considering the Figure 2, the role-name "*employee*" will be mapped to a class in OWL, and this class will be a sub-class of the class named "*person*"; the same idea is used in the role-name "*employer*". The relation between the role-name with the affordance of its right side will be better explained in the transformation rule.

Whole-Part – In the heuristics for UML, the whole-part is mapped as a composition. Since there is not the composition concept in OWL, it is necessary to reconsider this heuristic. Based on Bonacin *et al.* [2], there are two situations of whole-part relation. First is when both the source affordance and the target affordance are nouns. In this situation the affordances or agents are mapped as classes; then an object property called "*partOf"* can be created, and the target class will be a restriction of this source class. For example, in the Figure 2, the class agent "*departament*" mapped as a class in OWL will be part of the agent "*organization*" that also was mapped as a class into OWL. The second situation, both affordances would be verbs, so based in the affordance heuristic described, both of these affordances would be object properties in OWL. Therefore the target affordance will be mapped as sub-property of the source affordance that is also an object property. Moreover, since there is not part without the whole, when there is a whole-part relationship is also necessary to have an ontological dependence between the affordance of whole and part. The implementation details in applying this heuristic are presented in the respective transformation rule.

Specialization – The specialization can be used in agents, affordances or role-name; the specialization relation between the generic and the more specific type can be between nouns and also between verbs, as in whole-part. When the more generic affordance type is an action and it is mapped as an object property, then the more specific affordances will be mapped as a sub-property of the object property in OWL that represents the more generic affordance. Nevertheless, when the more generic affordance type is an entity (*i.e.* an OWL class), and consequently is mapped as a class in OWL, the more specific affordances will be mapped as classes in OWL and it will be sub-classes of the class more generic. The situation when the more specific affordances are verbs is an exception and it will be explored in the discussion section.

Ontological Dependence – This relation between affordances is the most common and used in the OC modeling. When an object cannot exist without other, an association between classes can be modeled in OWL. For example, the ontological dependence that exists between the affordances "*society*" and "*person*" in Figure 2 suggests an association between then in OWL. In the context of UML it is a relationship between two classes. In OWL, independent of the type of the affordance (*e.g.* verb or noun) both in the source or target affordance will have an object property between them. For that, creating an object

property called as "*depends_on*", the source affordance can be mapped as the range of this property, and the target affordance of the relation can be mapped as domain of this property. Considering the Figure 2, the affordance "*project*" is ontologically dependent of the affordance "*organization*"; thus the transformation will create an object property stating that "*project*" depends on "*organization*". There is a temporal relation between the ontological dependence of two affordances; so an affordance that depends on other just will exist while the other exists. Just using the object property as proposed in this heuristic is not going to be enough to fully represent the concept of ontological dependence of SAM into OWL. During the discussion section it will be discussed in details. Instantiating all this heuristics as computer procedures will be explained in the following transformation rule section.

3.2 Transformation Rules with Human Intervention

In order to computationally accomplish the ontology design as stated by Reis *et al.* [13], first is necessary to model the OC. For that, we may use the SONAR tool [14] - an OC modeling software tool. A computer-tractable SWO is obtained through new functionalities of SONAR that was extended with transformation rules based on the heuristics proposed in this work (section 3.1).

In the proposed assisted process, after modeling the OC, the user in the role of an analyst may transform it into OWL ontology by using wizards on the SONAR's "Tool" menu; in the option "Export OWL ontology with heuristics". In order to accomplish the whole process it is necessary to consider two important steps using the wizards: first, the analyst will need to specify which affordances will be mapped to OWL classes. Figure 3 shows an example of the SONAR's wizard where the analyst specifies which affordances will be mapped into OWL classes.

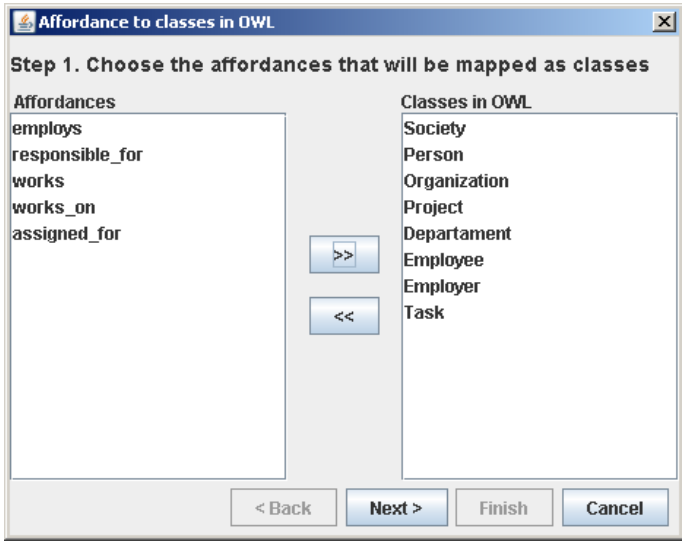


Figure 3: SONAR's interface to choose the affordances that will be OWL classes

Following, in the next step the analyst must specify which affordances will be mapped as object properties in OWL, and to associate these with appropriated OWL classes (affordances that were already mapped to classes) that will be connected (contain) to each object property selected (see Figure 4). This is important, since it is necessary to know which class will have priority on an object property (*i.e.* which affordance will have priority on other). For example, as illustrated by Figure 2, thinking in OWL the analyst must ask: Who employs? The "*employee*" or the "*employer*". Thus using the wizard as shown in Figure 4 the analyst can set that the "*employer*" employs (*i.e.* the class "*employer*" in OWL will be in the domain of the object property "*employ*"). It is also necessary to observe that in Figure 4 the class "*employee*" will be connected to the object properties: "*assigned_for*", "*works_on*" and "*works*". Using this step the necessary information will be available to the procedure to determine the appropriated transformation rules that will be applied, and consequently derive an initial version of the SWO. After the second step (Figure 4) the analyst will need to choose the path that the OWL file will be recorded.

Next we present how each OC concept is mapped to OWL code, considering the proposed heuristics and the information provided in the steps of the wizard; parts of OWL code are also illustrated and exemplified regarding the application of the implemented transformation rules in SONAR tool relative to the OC of Figure 2.

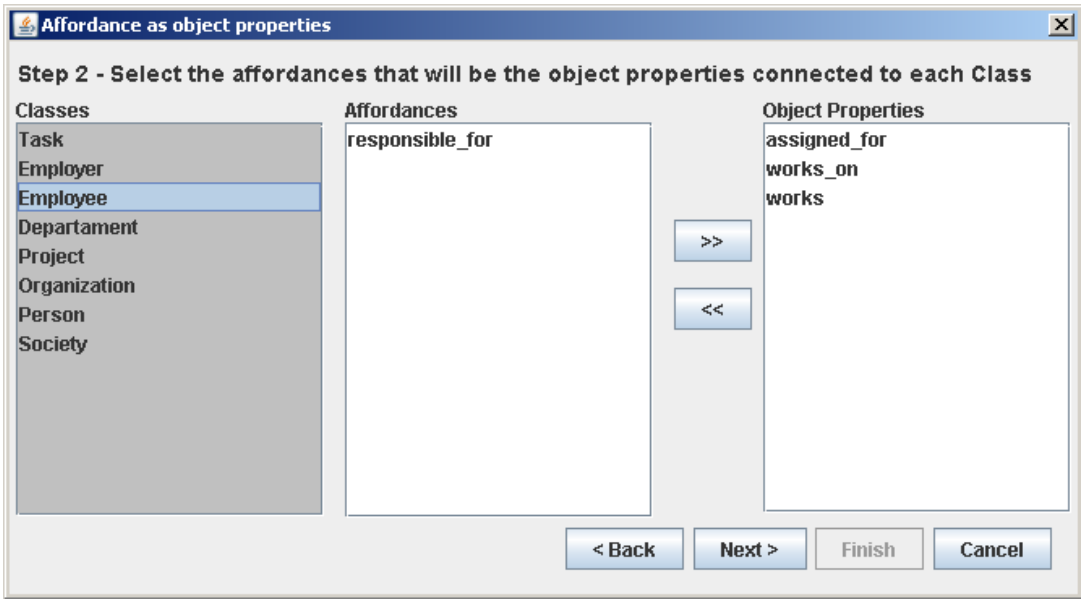


Figure 4: SONAR's interface to choose the affordances that will be mapped as object properties in OWL

Transformation Rules

Affordances – According to the affordance heuristic for mapping the affordances, they can be an OWL class or an object property. Using the wizards illustrated by Figure 3 and

Figure 4 the tool has the information that describes which affordances are mapped to classes; then the rule transformation creates an OWL class for each affordance set as class, as for example to the affordance "*task*"; see the following OWL code generated for this affordance.

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

However, since an affordance is set as an object property, the rule creates an object property for each affordance. The domain of each object property created is the class defined at the wizard (Figure 4) by the analyst. Thus if the class "*employee*" set the object property "*works*", then the "*employee*" is the domain of the object property. See the following code.

```
<owl:ObjectProperty rdf:ID="works">
    <rdfs:domain rdf:resource="#employee"/>
    <rdfs:range rdf:resource="#departament"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="works_Inv">
    <rdfs:domain rdf:resource="#departament"/>
    <rdfs:range rdf:resource="#employee"/>
    <owl:inverseOf rdf:resource="#works"/>
</owl:ObjectProperty>
```

The rule also creates the object property inverse of each object property mapped. With that it is possible to have the passive construction; the object property "*works_Inv*" is the inverse of the resource "*works*", thus the information of other perspective is known, as in this example, the ontology also represents who works in the "*department*" – the "*employee*" . Considering the fragment of code just presented, the range is the resource "*department*". In order to set the appropriate range, the affordance transformation rule gets the first affordance that is not an object property from the first ontological dependence connected to the considered object property. If the source affordance was set as an object property, it cannot be set as the range, then the rule considers the source affordance from the next ontological dependence. When the source element is a set of affordances as specialization, the range will be set as the most generic affordance, since in current SONAR version there is not how to set a specific affordance of a specialization. And, when there is a role-name in the ontological dependence, the rule considers the role-name; for example in the next OWL code; in this case, the domain of the object property "*employs*" is the "*employer*", and since the other ontological dependence is different from the one between "*employer*" and "*employs*", it is the one that contain the role-name "*employee*" that will be set as the range of the object property "*employs".*

```
<owl:ObjectProperty rdf:ID="employs">
    <rdfs:domain rdf:resource="#employer "/>
```

```
    <rdfs:range rdf:resource="# employee "/>
 </owl:ObjectProperty>
```

Agents – In the "agent heuristic" a class called "*Agent*" is created in OWL, and all agents from the OC form a sub-class of that. See the following OWL code example showing the classes "*Agent*" and "*person*" as a sub-class of "*Agent*"; the same happens for the class "*organization"*.

```
<owl:Class rdf:ID="Agent"/>
<owl:Class rdf:ID="person">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="Agent"/>
    </rdfs:subClassOf>
 </owl:Class>
```

Determiners – In order to accomplish the "determiner heuristic" for each determiner of the OC an *owl:DatatypeProperty* is created, which has the name of the determiner. The domain of each data property created is the OWL class of the affordance to which the determiner is connected, and the range is a *string* resource. The OWL code example shows the property "*proj_budget"* and its domain "*project*"; the same happens to the property "*function*" which has the domain "*employee*".

```
<owl:DatatypeProperty rdf:ID="proj_budget">
    <rdfs:domain rdf:resource="#project"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
 <owl:DatatypeProperty rdf:ID="function">
    <rdfs:domain rdf:resource="#employee"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>
```

Role-Name – In the "role-name heuristic", classes in OWL are created for each role-name as a sub-class of the agent of the left side of the role-name. According to the following OWL code illustrated, the role-name "*employee*" is an OWL class and it is sub-class of "*person*" class; the same happens with the role-name "*employer*" which is sub-class of "*organization"*.

```
<owl:Class rdf:ID="employee">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
 </owl:Class>
<owl:Class rdf:ID="employer">
```

```
    <rdfs:subClassOf>
        <owl:Class rdf:ID="organization"/>
    </rdfs:subClassOf>
</owl:Class>
```

Regarding the relation of the role-name with the affordance at the right side, considering the ontological dependence to which the role-name belongs, whether the right side affordance of the role-name was an object property, then the role-name once mapped as an OWL class will be in the domain of this object property. In this situation the rule transformation verifies if the class role-name is already "owner" of the object property (*i.e.* whether the class role-name is already in the domain of this object property), thus this relation was already done in OWL; if not, it is necessary to set the class role-name as domain of the object property. In the following OWL code example the affordance "*employee*" was already in the domain of the object property "*assigned_for*".

```
<owl:ObjectProperty rdf:ID="assigned_for">
    <rdfs:domain rdf:resource="#employee"/>
    <rdfs:range rdf:resource="#project"/>
</owl:ObjectProperty>
```

On the contrary, if the right side affordance (target of the ontological dependence) is not an object property, the transformation rule creates a relation between these two classes, *i.e.* between the role-name and the class of the right side; then an object property "*depends_on*" is created, and the domain is set as the OWL class that represents the right side affordance from the ontological dependence, and the range as the role-name. There is not any situation as this in Figure 2.

Whole-Part – In the UML heuristics, the whole-part is mapped as a composition. Since there is not the composition concept in OWL it is necessary to reconsider this heuristic. Based on Bonacin *et al.* [2] there are two situations of whole-part relation. First, when both the source affordance and the target affordance are nouns. In this situation the affordances were mapped as classes using the wizard of Figure 3; to represent this, an object property called "*partOf*" was created, and the target class is set as a restriction of the source class. For example, in the Figure 2, the agent "*departament*" mapped as an OWL class is part of the agent "*organization*" that also was mapped as a class into the OWL ontology.

In order to accomplish the "whole-part heuristic", specific object properties were created. One is the object property "*hasPart*" that is inverse of the transitive property "*partOf*". The following OWL code illustrates the rule.

```
<owl:ObjectProperty rdf:ID="hasPart">
    <owl:inverseOf>
        <owl:TransitiveProperty rdf:ID="partOf"/>
    </owl:inverseOf>
</owl:ObjectProperty>
```

```
<owl:TransitiveProperty rdf:ID="partOf">
    <owl:inverseOf rdf:resource="#hasPart"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
```

The following object properties "*hasPart_directly*" and "*partOf_directly*" which respectively are sub-property of "*hasPart*" and "*partOf*" were created in order to enable inference regarding these properties.

```
<owl:ObjectProperty rdf:ID="hasPart_directly">
    <rdfs:subPropertyOf>
            <owl:ObjectProperty rdf:ID="hasPart"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf>
            <owl:TransitiveProperty rdf:ID="partOf_directly"/>
    </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="partOf_directly">
    <rdfs:subPropertyOf>
            <owl:ObjectProperty rdf:ID="partOf"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf rdf:resource="#hasPart_directly"/>
</owl:ObjectProperty>
```

According to the "whole-part heuristic", since the affordances (source and target) were known and set as classes using the wizard (Figure 3), the target class will be a restriction of the source class. The OWL class "*departament*" is part of the class "*organization*". In order to represent it using the created object properties ("*partOf*" and "*hasPart*"), the class "*organization*" is set as restricted by a sub-class on the property "*hasPart*", in which all values from the "*organization*" come from the class "*departament*". The next OWL code describes the representation of the "*organization*" class.

```
<owl:Class rdf:ID="organization">
    <rdfs:subClassOf>
            <owl:Restriction>
                    <owl:onProperty>
                            <owl:ObjectProperty rdf:ID="hasPart"/>
                    </owl:onProperty>
                    <owl:allValuesFrom>
                            <owl:Class rdf:ID="departament"/>
                    </owl:allValuesFrom>
            </owl:Restriction>
```

```
    </rdfs:subClassOf>
    <rdfs:subClassOf>
            <owl:Class rdf:ID="Agent"/>
    </rdfs:subClassOf>
</owl:Class>
```

The "*task*" class is also part of the "*project*" class, thus the same idea is followed as in the "*organization*" to represent the "*project*" class in OWL; the following OWL code illustrates it.

```
<owl:Class rdf:ID="project">
    <rdfs:subClassOf>
            <owl:Restriction>
                    <owl:onProperty>
                            <owl:ObjectProperty rdf:ID="hasPart"/>
                    </owl:onProperty>
                    <owl:allValuesFrom>
                            <owl:Class rdf:ID="task"/>
                    </owl:allValuesFrom>
            </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

In both situations ("*departament*"- "*organization*" and *"task"* - "*project*") an object property of ontological dependence between the affordances whole and part is also created. The "ontological dependence rule transformation" will be better explained later in this section. As presented by the "whole-part heuristic", the part is ontologically dependent on the whole, thus "*departament*" depends on "*organization*" and *"task"* depends on "*project*". In order to handle it, this transformation rule also creates object properties that represents ontological dependences as described by the OWL example code that illustrates these object properties.

```
<owl:TransitiveProperty rdf:ID="departament_dependsOn_organization">
    <rdfs:domain rdf:resource="#departament"/>
    <rdfs:range rdf:resource="#organization"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:ObjectProperty>
<owl:TransitiveProperty rdf:ID="task_dependsOn_project">
    <rdfs:domain rdf:resource="#task"/>
    <rdfs:range rdf:resource="#project"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:ObjectProperty>
```

In the other situation, when both affordances (whole and part) are set as object properties in OWL using the wizard (Figure 4), the target object property that represents the part is mapped as a sub-property of the source object property that represents the whole. This situation does not appear in Figure 2 since the whole-part affordances of the relations in the OC were set as OWL classes. However, this rule transformation foresees this situation and is able to handle it. Regarding the ontological dependence representation of such situation, there are limitations as described in the discussion section.

Specialization – According to the "specialization heuristic" and Figure 2, since the affordance "*task*" was set as an OWL class in the wizard, the more specific types of "t*ask*" must also be set as classes and sub-classes of *"task"*. The OWL code below shows this situation with the affordances "*simple*" and "*complex*".

```
<owl:Class rdf:ID="simple">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="task"/>
    </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="complex">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="task"/>
    </rdfs:subClassOf>
 </owl:Class>
```

In Figure 2 there is not any situation in which the more generic affordance was set as an object property, but it could happen, and the rule transformation foresees that such situation may happen. Imagine that the affordance "*works*" that was set as object property could have more specific kinds of work such as: "*write*" or "*paint*" for example. Then the affordances "*write*" and "*paint*" would be sub-properties of the object property "*works*". The OWL code would be as:

```
<owl:ObjectProperty rdf:ID="write">
<rdfs:subPropertyOf>
            <owl:ObjectProperty rdf:ID="works"/>
    </rdfs:subPropertyOf>
 </owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="paint">
<rdfs:subPropertyOf>
            <owl:ObjectProperty rdf:ID="works"/>
    </rdfs:subPropertyOf>
 </owl:ObjectProperty>
```

Ontological Dependence – In this rule transformation, transitive properties in OWL are created (*i.e.* object property) for all ontological dependences of the OC. Since that both

source and target affordances are classes in OWL, the property is created setting the domain and range; the domain is set with the target affordance, while the range is set with the source affordance. Please consider the two next OWL code examples took from the transformation applied to the OC from Figure 2 regarding the affordance "*project*" depending on the affordance "*organization*", and the affordance "*person*" depending on the affordance "*society*".

```
<owl:TransitiveProperty rdf:ID="project_dependsOn_organization ">
    <rdfs:domain rdf:resource="# project "/>
    <rdfs:range rdf:resource="# organization"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="person_dependsOn_society">
    <rdfs:domain rdf:resource="#person"/>
    <rdfs:range rdf:resource="#society"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
```

If one of the affordances (source or target) were not set as a class in OWL it cannot be set as domain or range of the object property. In this situation the rule transformation seeks for the class that is in the domain of this object property. Then, the object property will be replaced by this class. Such situations may lead to wrong existential relationships as will be argued in the discussion section. Moreover, when the source affordance of the ontological dependence is a specialization, the relation is made using the more generic affordance since it is not possible to connect an ontological dependence to a specific specialized affordance in SONAR.

After applying all these rules an OWL file containing all the OWL codes generated from the transformations is created. The OWL ontology created can be loaded in ontology software editors such as Protégé[1]. Figure 5 shows a hierarchical visualization of the OWL classes created applying the proposed heuristics and transformation rules in the OC of Figure 2. The relations created by the object properties and the data properties or individuals are not present in the Figure 5.

[1] http://protege.stanford.edu/

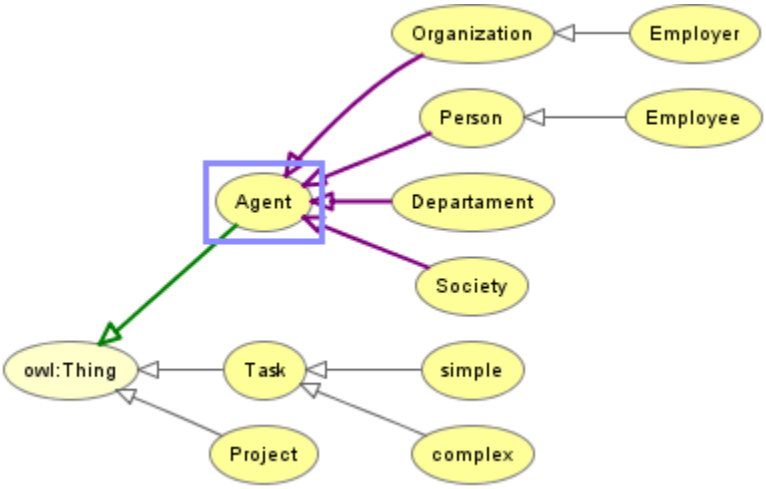


Figure 5: Visualization of the OWL classes created by the transformation rules in SONAR

In the next section the application of this assisted process is illustrated using OC modeling from a real case study.

4 Illustrating the Process in a Case Study

In order to exemplify the proposed transformation process in a real context, the SAM was applied on announcements created by users of an Inclusive Social Network Service (ISN) [22] in an attempt to model meanings from real contents. According to Neris *et al.* [22] ISNs are social networks mediated by computational systems in which each person can integrate a group and interact in order to produce elements that can be shared. ISNs have the objective to be accessible to the widest variety of users, including those less familiar with technology or with low literacy levels. In this work the considered ISN is the '*VilanaRede*'[2].

VilanaRede' users interact with the system and among them through the system by creating announcements of products, services, events and ideas. These users are mostly Brazilian people in process of digital literacy. The content domain of the whole set of announcements available in the ISN *VilanaRede* is wide-ranging; *i.e.* there are announcements about various contexts. In order to apply the SAM and to create the OC, two main contexts were chosen resulting in two OCs. All announcements from such contexts were collected and the OC was built based on them. The first OC modeled is relative to the context of

[2] www.vilanarede.org.br

cooking and meal ordering. The other one is in the domain of physical exercises and health promotion. In the sequence, the proposed transformation rules are applied in both diagrams and the results are presented to evaluate the proposal in a practical and real life context.

Transformation A

Figure 6 shows the outcome of SAM applied to the announcements of the ISN regarding the context of cooking and meal ordering. The OC represents the main agents and affordances and their relationship and ontological dependences in this context. In this OC "*meal*" ontologically depends on "*ingredient*" and on "*person*" in the role-name of "*cook*" that cooks. The "*person*" in the role of a "*cook*" has the affordance "*cooks*" that depends on a "*cook*" to follow a "*recipe*". The "*person*" in the role of "*seller*" sells the "*meal*" that someone in the role of "*cook*" cooks. The "*recipe*" does not ontologically depends on the "*ingredient*" to exist since the "*recipe*" continues to exist even when the ingredients do not exist anymore, different from the "*meal*" that depends on the ingredients to exist. The affordance "*order*" depends on a person in the role of "*buyer"* to exist and on a "*meal*" that was cooked. Besides, the affordance "*sells"* ontologically depends on "*person*" in the role of "*seller*" and also of "*meal*". There are various specific kinds of "*meal*".

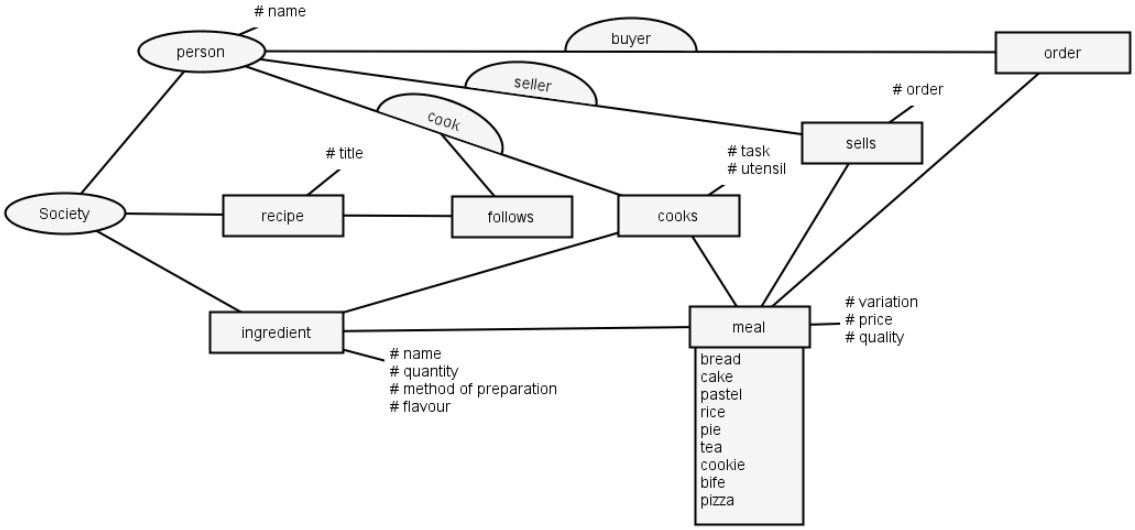


Figure 6: The OC modeled from the ISN announcements about cooking and meal ordering

After modeling the OC through SONAR, the OWL ontology is exported based on the proposed heuristics and transformation rules. The transformation process applied to the OC illustrated by Figure 7 is called Transformation A. The first step is to choose the affordances that will be mapped to OWL classes. Figure 7 shows this step in SONAR's interface. The affordances in the left side of this interface are the object properties.

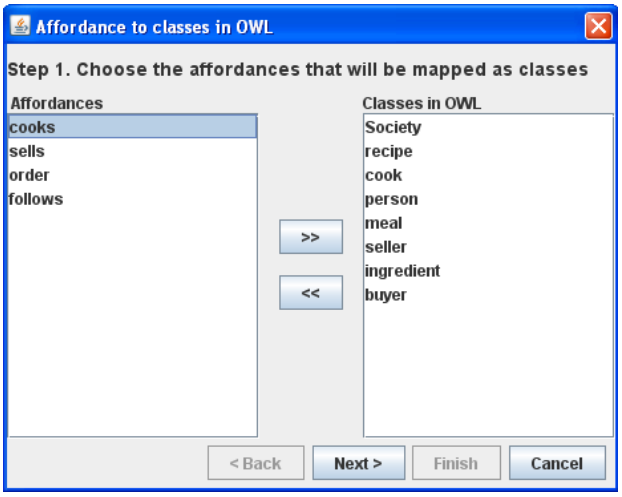


Figure 7: Choosing the affordances mapped to OWL classes by the Transformation A

In the next step the analyst has to select the affordances that will be the object properties in OWL connected to each class. Figure 8 shows this step in SONAR's interface. In this interface, for the OC of Figure 6, the class "*buyer*" was chosen to be in the domain of the object property "*order*". The affordance "*seller*" mapped to OWL class is the domain for the affordance "*sells*" that was mapped as object property, while the class "*cooks*" for the affordances "*cooks*" and *"follows"* that were mapped as object properties.

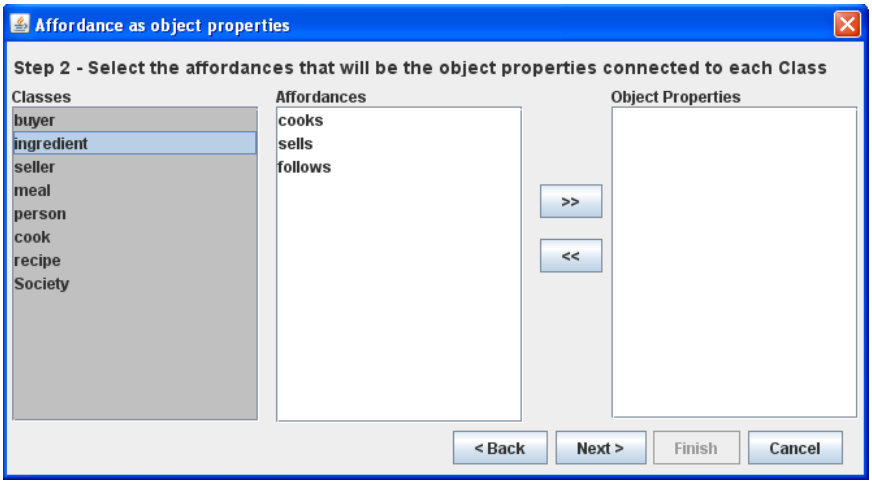


Figure 8: Choosing the affordances mapped to object properties in OWL for the Transformation A

Some important relations that were mapped from OC to OWL by the proposed heuristics and transformation rules regarding the OC (Figure 6) are presented as follows. In Appendix A the OWL code of the Transformation A is fully described. First, as "*buyer*",

"*seller*" and "*cook*" are role-names of the agent *"person",* they are mapped as sub-classes of *"person"*. Observe the correspondent OWL code:

```
<owl:Class rdf:ID="person">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="agent"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="cook">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="seller">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="buyer">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
</owl:Class>
```

Once the affordance "*meal"* has various specific affordances all of them were mapped as sub-classes of "*meal"*. The following OWL code shows some more specific affordances:

```
<owl:Class rdf:ID="bread">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="cake">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
```

The object properties are also successfully created according to the rule transformations. Regarding the object property "*cooks"* the domain is *"cook"* and the range is "*ingredient"* since "*cooks"* depends on the affordances "*ingredient"* and *"cook"* to exist. The same happens for the affordance "*follows"* since *"cook"* follows a "*recipe"*. The next OWL code illustrates it.

```
<owl:ObjectProperty rdf:ID="cooks">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#ingredient"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="follows">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#recipe"/>
</owl:ObjectProperty>
```

The same situation happens for "*sells"* regarding *"seller"* and "*meal*", and also for "*order"* regarding *"buyer"* and "*meal*".

```
<owl:ObjectProperty rdf:ID="sells">
    <rdfs:domain rdf:resource="#seller"/>
    <rdfs:range rdf:resource="#meal"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="order">
    <rdfs:domain rdf:resource="#buyer"/>
    <rdfs:range rdf:resource="#meal"/>
</owl:ObjectProperty>
```

All the object transitive properties related to ontological dependences are also created. Some examples are highlighted. The concept of "*recipe"* depends on *"Society"* to exist as well as the concept of "*meal"* depends on *"ingredient"*.

```
<owl:TransitiveProperty rdf:ID="recipe_dependsOn_Society">
    <rdfs:domain rdf:resource="#recipe"/>
    <rdfs:range rdf:resource="#Society"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="cooks_dependsOn_person">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="meal_dependsOn_ingredient">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="#ingredient"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
```

The determiners are also successfully mapped by the rules. The OWL code examples about the created data properties in OWL are based on the modeled determiners in the OC

(Figure 6). Regarding the determiners of affordances mapped as object properties, it is not possible to create them. The discussion section explains about it.

Figure **9** shows visually the OWL classes hierarchy created by the transformation at SONAR.

```
<owl:DatatypeProperty rdf:ID="variation">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="quantity">
    <rdfs:domain rdf:resource="#ingredient"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

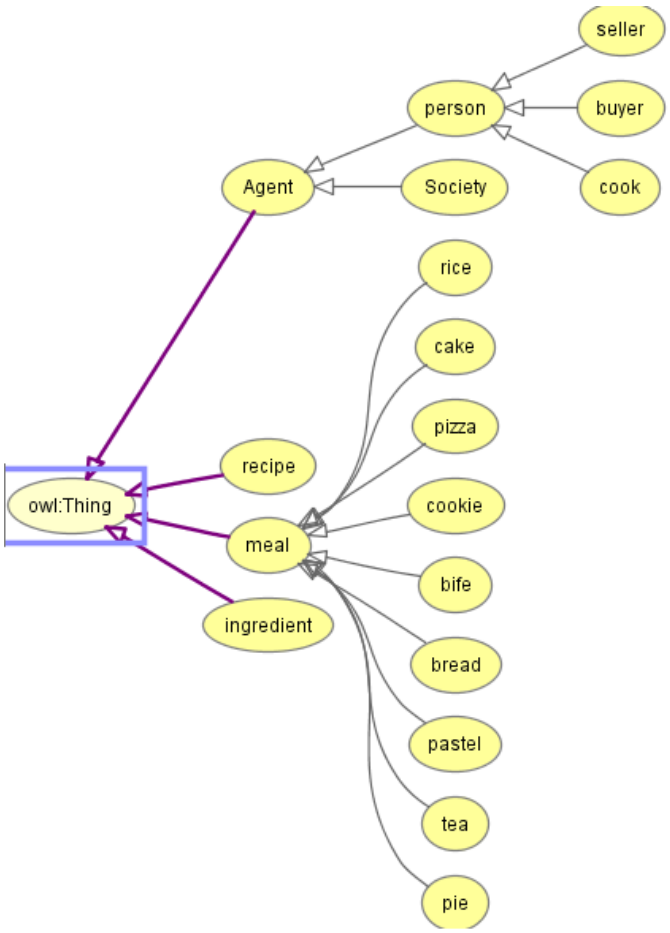


Figure 9: The OWL classes created in the Transformation A

Transformation B

In order to better test and exemplify the proposed transformation rules in real life situation, the transformation process was applied to another and more elaborated OC. The called Transformation B was applied to an OC built using a collection of announcements from *VilanaRede* in the context of physical activities and health promotion. Figure 10 presents the OC modeled. A brief reading of the diagram can be done pointing out the most important relations.

The *"habit"* is an affordance of *"person"*; "*habit"* and *"disease"* together affords "*prevents*". "*Person*" in the role of a *"physical educator" "guides"* "*physical activity*". *"Physical exercise"* is part of a *"physical activity"*. Besides, "*person*" in the role of *"health community agent"* and *"habit"* together affords *"promotes"*. According to the shared semantics in the ISN, *"improvement in health"* is ontologically dependent on the affordance *"promotes"* and *"physical exercise"*; while "*well-being"* depends on *"health"* and *"habit"*.

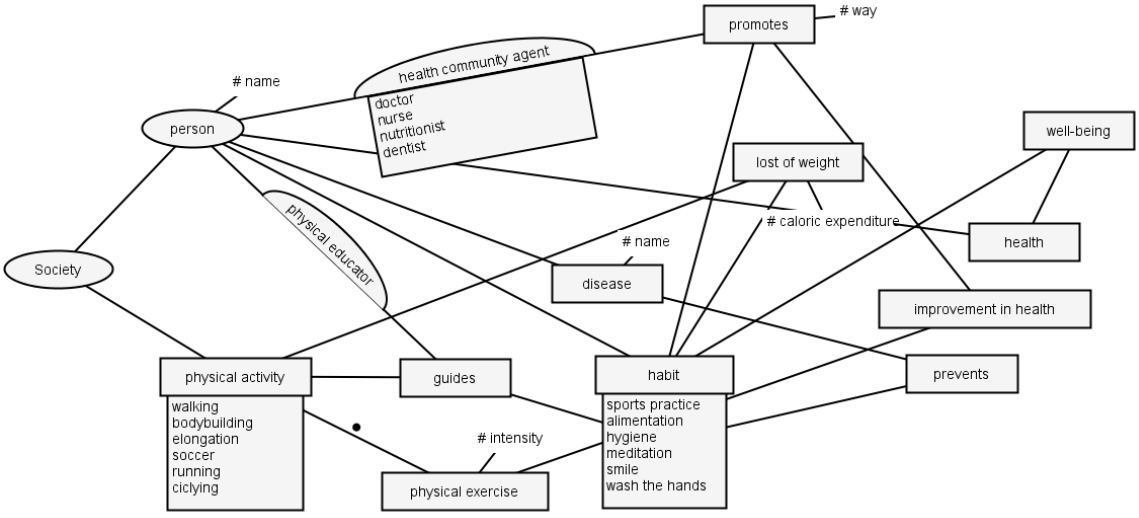


Figure 10: The OC modeled from ISN announcements about physical activities and health

As in the Transformation A, after modeling the OC, the analyst must use the SONAR's wizard steps to choose the affordances that are mapped to OWL classes or object properties. Figure 11 and Figure 12 show these steps. After that, the Web ontology described in OWL is created automatically by SONAR. In Appendix B the OWL code of the Transformation B is fully described.

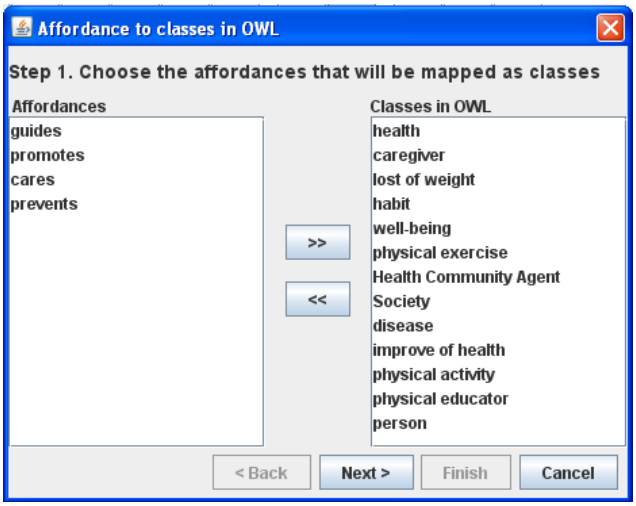


Figure 11: Choosing the affordances mapped as OWL classes for the Transformation B

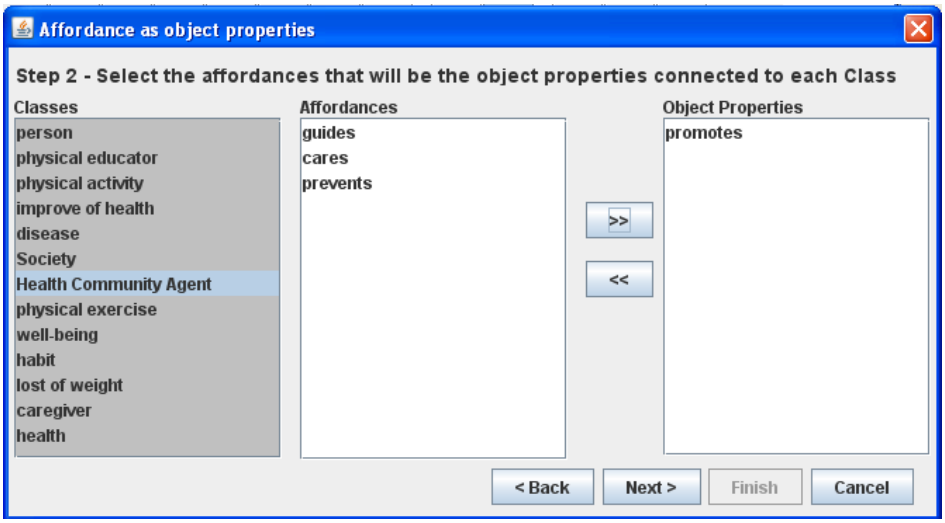


Figure 12: Choosing the affordances mapped as object properties in OWL for the Transformation B

With the OC illustrated in Figure 10 and the transformation rules, "*person*" is a sub-class of "*agent*" as well as *"physical educator"* and *"health community agent"* are sub-classes of "*person*". Observe the following generated OWL code:

```
<owl:Class rdf:ID="person">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="Agent"/>
```

```
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="physical educator">
    <rdfs:subClassOf>
          <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="health community agent ">
    <rdfs:subClassOf>
          <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
  </owl:Class>
```

"Physical exercise" is part of the "*physical activity*", thus in OWL the *"physical exercise"* is modeled as a restriction of the "*physical activity*" according to the "whole-part transformation".

```
<owl:Class rdf:ID="physical activity">
    <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty>
                      <owl:ObjectProperty rdf:ID="hasPart"/>
                </owl:onProperty>
                <owl:allValuesFrom>
                      <owl:Class rdf:ID="physical exercise"/>
                </owl:allValuesFrom>
          </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

The specific relations are also mapped to OWL. Observe the following OWL code exemplifying a specific affordance of "*physical activity"* and of *"habit"* in Transformation B. The same happens for the role-name *"health community agent"*. "N*urse"* in the context of the ISN is a kind of *"health community agent"*.

```
<owl:Class rdf:ID="walking">
    <rdfs:subClassOf>
          <owl:Class rdf:ID="physical activity"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="sports practice">
    <rdfs:subClassOf>
          <owl:Class rdf:ID="habit"/>
```

```
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="nurse">
    <rdfs:subClassOf>
          <owl:Class rdf:ID="health community agent"/>
    </rdfs:subClassOf>
  </owl:Class>
```

Regarding the affordances set as object properties such as: "*guides*", "*prevents*", "*promotes*", "*cares*" they are also successfully described in OWL. The next OWL code example presents the object properties "*guides*" and "*prevents*" showing that not just role-names or agents can be in the domain, but also any affordances mapped as OWL classes, in this case the class "*habit*".

```
<owl:ObjectProperty rdf:ID="guides">
    <rdfs:domain rdf:resource="#physical educator"/>
    <rdfs:range rdf:resource="#physical activity"/>
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="prevents">
    <rdfs:domain rdf:resource="#habit"/>
    <rdfs:range rdf:resource="#disease"/>
  </owl:ObjectProperty>
```

Some examples of transitive properties of Transformation B to model the ontological dependences are illustrated next. It means that the concept of "*health*" just exists while the concept of *"person"* exists, as well as the concept of "*well-being*" is ontologically dependent on "*health*" as modeled by the object property.

```
<owl:TransitiveProperty rdf:ID="improve of health_dependsOn_physical exercise">
    <rdfs:domain rdf:resource="#improve of health"/>
    <rdfs:range rdf:resource="#physical exercise"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:ID="health_dependsOn_person">
    <rdfs:domain rdf:resource="#health"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:TransitiveProperty>
  <owl:TransitiveProperty rdf:ID="well-being_dependsOn_health">
    <rdfs:domain rdf:resource="#well-being"/>
    <rdfs:range rdf:resource="#health"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
```

</owl:TransitiveProperty>

As an example of determiner, the "*intensity"* is shown as a determiner of *"physical exercise"*. The hierarchy of OWL classes created in the Transformation B by SONAR is visually illustrated by Figure 13.

```
<owl:DatatypeProperty rdf:ID="intensity">
    <rdfs:domain rdf:resource="#physical exercise"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
```

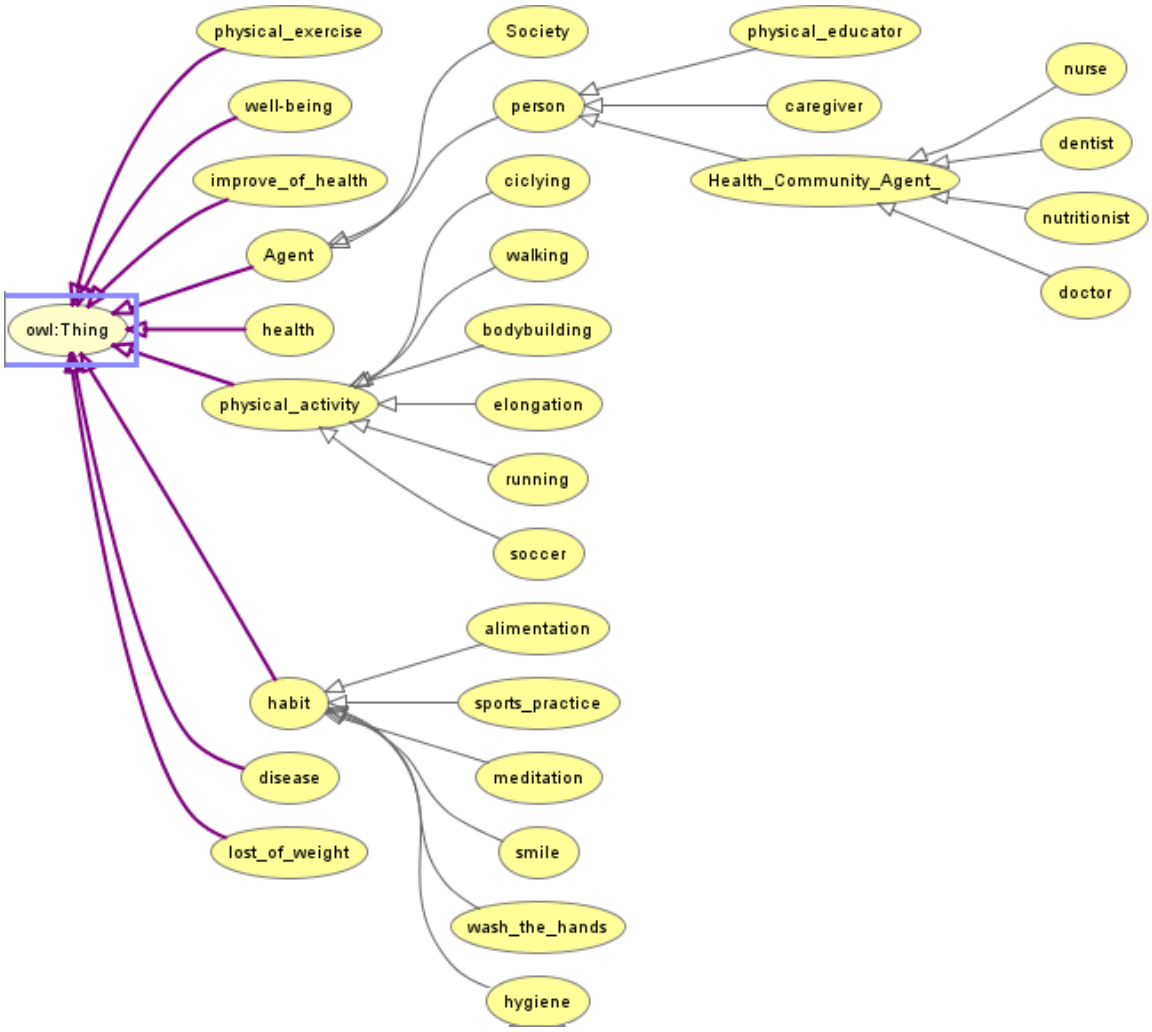


Figure 13: The OWL classes created in the Transformation B

5 Discussion

The first and necessary topic to be discussed refers to the different visions of the approaches we have worked with. The SAM follows the subjectivist paradigm while the traditional Web ontology approach follows the objectivist paradigm. Whereas objectivism assumes a single reality and explains differences of ideas as aberrations, subjectivism treats different ideas of individuals as starting point for a shared reality; subjectivism emphasizes the abilities of individuals, their freedom to choose courses of action and the moral responsibility for their choice, as well as the uncertainty, novelty and strife they bring about [10:24].

Concerning OC and Web ontologies, the differences between them become evident already in the root of both models; in OC the root of everything is a certain *Society,* while in OWL the root of everything is the class *Thing*. By using SAM the model exposes a few basic assumptions underpinned by the ontological principles. There is a root agent in the chart which functions as the ultimate antecedent for the whole problem domain under study. This root agent is the social community in which all its members share the same fundamental concepts and culture. In SAM, Ontology is not an absolute knowledge rather it is relative to a determined *Society*. Thus regarding the modeling in OWL there is not *Society* without *Thing* (the *Society* is a *Thing*), and in the SAM perspective there is not *Thing* without *Society* (every affordance depends on the *Society* existence). Moreover, in SAM the focus is in the agents including their patterns of behaviour (affordances) and their ontological dependencies while in OWL the focus is the concepts (classes) and their relationships. There is a paradigm difference in modeling information in these two perspectives.

However, according to Reis *et al.* [13] identifying the agents of the context, and understanding and modeling the invariants of behaviour of these human agents are key points for more accurate and flexible computer-interpretable ontology models. It is not intended to create an OC in OWL or to substitute the OC at the conceptual or business level; OWL models and OC do not replace each other. Furthermore, it is not expected that the OC and OWL ontologies model the same thing, since they represent different concepts and visions. There are different concepts in OWL and in SAM, consequently the semantic models (formed from these different concepts) exhibit different representations of reality. Nevertheless, following the proposed heuristics of this work some properties such as agent-affordance relationship are emphasized during the transformation rules while other aspects from the OC may not be fully transcribed to OWL.

In this context it is important to verify whether the heuristics really enable to map properties from one model to other correctly. It is necessary to check whether the OWL ontology created has inconsistencies from two points of view. First by conceptually analyzing whether the existential and temporal dependences between the affordances in the OC are kept, observing whether the agent-affordance relationship still exists in the OWL ontology. The heuristics clearly map each entity from one model to the other. The agents and role-names create a well defined hierarchy of classes. Affordances set as classes are clearly mapped to OWL classes, while the affordances set as object properties (*i.e.* represent the patterns of behaviour as properties) tie the relationship between the agents and role-

names with their affordances (action sense) to concepts (mapped as classes). The specifics of affordances are successfully mapped to sub-classes ensuring the relationship with the more generic affordance. Determiners have an immediate mapping to data properties. Whole-part relationships are set as restriction of classes preserving the relation between whole and part mapped in the OC. Ontological dependences are also represented as object properties connecting the affordances that depend on each other to exist. Limitations regarding the temporal representation of the ontological dependences will be soon explained in this section.

Moreover, it is important to examine whether the OWL ontology is consistent from the SW point of view. For that, it is necessary to observe whether all the relationship of the model logically make sense. Once the rules implementations have followed the OWL DL restrictions in order to assure computational completeness it may lead to consistencies. However it still may not avoid modeling problems. In order to verify it a more formal and precise study using Venn diagrams could be conducted analyzing the created ontology. For that, instances should be created. Besides, it is also possible to use a semantic reasoner as Pellet[3] available in ontology editor software tools to test the generated ontology and check its consistency. From the presented results it is possible to see the potential of the proposal to different SW initiatives. For instance, the SW ontology created by SONAR could support tasks such as semantic search and also could be integrated to other ontologies in the Web environment since it is described in OWL.

In addition to the clear potential of the solution it is possible to point out some limitations as well. First, the concept of ontological dependence is a hard problem to deal with in OWL; the solution treats part of this representation since the ontological dependence is a temporal concept, thus representing it as object properties is not enough. Still concerning ontological dependence, not all ontological relations modeled in the OC are possible to be fully transcribed to OWL following the heuristics. The situations that involve affordances mapped to object properties are more complicated to deal with. For instance, following the heuristics and using OWL DL, it is not possible to represent the ontological dependences between two affordances that were set as object properties, since both will not be mapped to OWL classes; a third object property should be necessary to set the domain and range with such objects that are not classes. In OWL DL, in order to retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time) the domain and range can be just set as a class or a set of classes.

There are also other situations of limitation. For instance, since the right affordance of an ontological dependence may be set to object property and the left affordance may be an

[3] http://clarkparsia.com/pellet

agent, an specialization (*i.e.* a specific affordance) or a noun affordance, thus using an approach in which the owner of the right affordance set as object property is considered as a possible domain can lead to wrong ontological relationships. The two following OWL codified situations that could be found in the Transformation A and B illustrate this situation.

```
<owl:TransitiveProperty rdf:ID="cooks_dependsOn_ingredient">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#ingredient"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>

<owl:TransitiveProperty rdf:ID="guides_dependsOn_physical activity">
    <rdfs:domain rdf:resource="#physical educator"/>
    <rdfs:range rdf:resource="#physical activity"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
```

The first property is about the ontological dependence between the affordances "c*ooks"* and *"ingredient"*. Since "*cooks"* was set as object property, the possible approach could use the owner of *"cooks"*; in this case "*cook"* would be the range of the property. However it leads to a wrong ontological dependence since "*cook"* does not need *"ingredient"* to exist, different from the affordance *"cooks"* that just exists while the affordance *"ingredient"* exists. The same situation occurs to the second property. The affordance "*guides"* exists just while *"physical activity"* exists, but the *"physical educator"* does not depend on the *"physical activity"*. Then some ontological dependences as these are not possible to be covered by the transformation.

Other ontological dependence situations, even not being mapped, do not lead to wrong statements. The next OWL code illustrates that in these cases it is not possible (due to the limitations already explained) to represent that "*meal"* depends on *"cooks",* but it is not wrong the representation that *"meal"* depends on *"cook"*. The transformation rules consider this situation (*i.e.* such object properties are created) but not the first one.

```
<owl:TransitiveProperty rdf:ID="meal_dependsOn_cooks">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="#cook"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>

<owl:TransitiveProperty rdf:ID="cooks_dependsOn_person">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
```

The "ontological dependence transformation rule" is not capable of dealing with the transitivity of the ontological dependence (*e.g.* from the SAM perspective if we eliminate the *Society* concept, then all other concepts that depend on it should not exist anymore). In the presented solution, just the more immediate ontological dependence is represented. In order to support and treat transitivity dependence, in the future it could be handled by Semantic Web Rule Language (SWRL)[4] or other extensions of the OWL. Therefore, the original model could be extended to better represent the temporal relations; for instance, it should be possible to create rules that represent situations such as: an individual of the "o*rganization*" class can only be instantiated whether it is associated to an individual of the "*society*" class, and if this individual (of the "*Society*" class) is deleted, the other also must be.

In OC usually have determiners connected to an affordance, besides the possibility to be connected to agents. Since some affordances can be set to object property by the assisted process, there are problems in representing it in OWL, once it is not possible in OWL DL to have a data property connected to an object property. Data properties are just connected to OWL classes. Other limitation is that in OC it is possible to have a determiner of a determiner. In order to represent that in OWL, it should be possible to set the domain of a data property as other data property, and this is not possible considering OWL DL definition. Both the domain and range must be set as a class or a set of classes. Moreover, Bonacin *et al.* [2] stated the possibility for the case of specialization in that the more generic affordances could be nouns and the more specific affordances could be verbs. This situation was not considered by the "specialization transformation rule" in this work, since there is not a way to an object property (the more specific affordances - verbs) to be set as sub-classes of a class in OWL (the more generic affordance - noun).

The proposed heuristics and the transformation rules implemented in SONAR in this work enable to create ontology described in OWL from OC; nevertheless such limitations still need to be treated. Regarding the implemented solution, it could also be possible to use meta-models with transformation rules described as for example using Atlas Transformation Language (ATL)[5] technology. In general, there is a growing need for solutions that deal with semantic aspects in Web systems. The considered OS approach may deal with the shortcomings of conventional SW ontologies, and this work contributes to a computational implementation in this direction. Such solution aligned to Reis *et al.* [13] proposal represents an effort toward a more human-representative Web Ontology.

---

[4] http://www.w3.org/Submission/SWRL/
[5] http://www.eclipse.org/atl/

6 Conclusion

The Web needs new approaches to better handling the complex meanings of the information in its environment. The evolution of the Web depends on more adequate methods and computer-tractable solutions that can represent the knowledge presented in Web applications content. In this direction, based on previous work which proposes to represent the knowledge based on Organizational Semiotics methods, this work presented a solution to a computer-interpretable "Semiotic Web ontology". An assisted process for building "Semiotic Web ontology" was proposed in an attempt to deal with the shortcomings of conventional SW ontologies.

For that, heuristics to support the creation of a Web ontology described in Ontology Web Language (OWL) from the outcomes of the Semantic Analysis Method (SAM) were presented. Based on these heuristics, transformation rules assisted by wizards were implemented in the SONAR case software tool in order to model the Ontology Chart and to derive an initial version of a Web ontology. The solution was tested with the modeling of real case context showing the potential of the proposed heuristics, and the usefulness of the implemented rules. The heuristics and transformation rules support the assisted process and materialize the Semiotic-based approach ideas for building Web ontologies. The solution brings opportunities to improve the semantic models used in the existing Semantic Web applications and initiatives, and also enhance the chances to "interoperate" with what already exists since OWL is a W3C standard.

Next steps involve refining the implemented transformation rules, generating representations using SWRL as described in the discussion section; new tests with more complex OCs are also important to improve the solution. Moreover, practical experiments that can illustrate the application of this approach, including the use of the "Semiotic Web ontology" in a search mechanism are planned to be executed. Further work can also involve the inclusion of the OS Norm Analysis Method in the process.

Acknowledgments

This work was funded by Microsoft Research - FAPESP Institute for IT Research (proc. n. 2007/54564-1) and by CNPq/CTI (680.041/2006-0). The authors also thank colleagues from CTI, IC (UNICAMP), NIED, InterHAD and Casa Brasil for insightful discussion.

References

1. BERNERS-LEE, T., HENDLER, J., LASSILA, O., 2001. **The Semantic Web**, Scientific American.
2. BONACIN, R., BARANAUSKAS, M. C. C., LIU, K., 2004. **From Ontology Charts to Class Diagrams: semantic analysis aiding systems design**. In Proceedings of the 6th International Conference on Enterprise Information Systems, ICEIS 2004, Porto, Portugal. v. 1. pp. 389-395.

3. CARDOSO, J., 2007. **The Semantic Web Vision: Where Are We?** Intelligent Systems, IEEE. Volume 22, Issue 5, pp: 84 – 88.
4. CARVALHO, M. L. B., 2005. **Web semântica e semiótica: ontologias e aplicação a permacultura**. Master Dissertation - Centro Universitário Eurípides da Marília (in Portuguese)
5. GÄRDENFORS, P., 2004. **How to Make the Semantic Web More Semantic.** In Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS), pp. 17-34.
6. GIBSON, J.J., 1977. **The Theory of Affordances**. In Perceiving, Acting, and Knowing. Eds. Robert Shaw and John Bransford.
7. GRUBER, T. R., 1993. **A translation approach to portable ontologies**. Knowledge Acquisition, v.5, n.2.
8. JACOB, ELIN K., 2005. **Ontologies and the Semantic Web**. Bulletin of the American Society for Information Science and Technology. Volume 29 - Issue 4, pp. 19 – 22.
9. KALYANPUR, A.; GOLBECK, J.; BANERJEE, J.; HENDLER, J., 2004. **Owl: Capturing semantic information using a standardized web ontology language**, Multilingual Computing & Technology Magazine, Vol. 15, issue 7.
10. LIU, K., 2000. **Semiotics in information systems engineering**. Cambridge University Press.
11. LIU, K., SUN, L., FU, Y., 2008. **Ontological Modeling of Content Management and Provision**. In Information and Software Technology, V.50 N.11, pp.1155-1164.
12. NOY, N. F. & MCGUINNESS, D. L., 2001. **Ontology Development 101: A Guide to Creating Your First Ontology**. Disponível em: <http://protege.stanford.edu/publications/ontology_development/ontology101.pdf> Accessed August 2010.
13. REIS, J. C., BONACIN, R. AND BARANAUSKAS, M.C.C., 2010. **A Semiotic-based Approach to the design of Web Ontologies.** In Proceedings of the 12th International Conference on Informatics and Semiotics in Organisations – ICISO 2010. Reading, UK. pp. 60-67.
14. SANTOS, T. M.; BONACIN, R.; BARANAUSKAS, M. C. C.; RODRIGUES, M. A., 2008. **A Model Driven Architecture Tool Based on Semantic Analysis Method.** In Proc. of the 10th International Conference on Enterprise Information Systems – ICEIS 2008. Barcelona, Spain, v. 2. pp. 305-310.
15. STAMPER, R. K.; ALTHANS, K.; BACKHOUSE, J., 1988. **Measur: Method For Eliciting, Analysing and Specifying User Requirements**. In: Computerized Assistance During the Information Systems Life Cycle. North-Holland, pp. 67-115.
16. STAMPER, R., LIU, K; M. HAFKAMP, Y., ADES, 2000. **Understanding the Role of Signs and Norms in Organisations: A semiotic approach to information systems design**. Journal of Behaviour and Information Technology, 19(1): 15-27.
17. STAMPER, R.K., 1993. **Social Norms in requirements analysis - an outline of MEASUR**. In Jirotka M, Goguen J, Bickerton M. (eds) Requirements Engineering, Technical and Social Aspects. New York.

18. STUDER, R. *et al.,* 1998. **Knowledge engineering: principles and methods**. Data & Knowledge Engineering, v.25, n. 1-2.
19. TANASESCU, V. & STREIBEL, O., 2007. **Extreme Tagging: Emergent Semantics through the Tagging of Tags**. In Proc. of the International Workshop on Emergent Semantics and Ontology Evolution *(ESOE2007) at ISWC/ASWC*, Busan, South Korea, pp. 84-85.
20. WORLD WIDE WEB CONSORTIUM (W3C), 2004. **OWL-Web Ontology Language**, Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features>, Accessed August 2010.
21. SALTER, A. **Semantic Modelling and a Semantic Normal Form**. SOCTR/01/01. School of Computing. Staffordshire University.
22. NERIS, V. P. A.; ALMEIDA, L. D. ; MIRANDA, L. C. ; HAYASHI, E. ; BARANAUSKAS, M. C. C., 2009. **Towards a Socially-constructed Meaning for Inclusive Social Network Systems.** In Proceedings of the International Conference on Informatics and Semiotics in Organisations. Beijing, China. pp. 247-254.

Appendix A: Transformation A

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/Ontological_Diagram_0.owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://www.owl-ontologies.com/Ontological_Diagram_0.owl">
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="recipe"/>
 <owl:Class rdf:ID="meal"/>
 <owl:Class rdf:ID="ingredient"/>
 <owl:Class rdf:ID="Society">
    <rdfs:subClassOf>
          <owl:Class rdf:ID="Agent"/>
    </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="Agent"/>
 <owl:Class rdf:ID="person">
```

```
    <rdfs:subClassOf>
        <owl:Class rdf:ID="Agent"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="cook">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="seller">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="buyer">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="person"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="bread">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="cake">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="pastel">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="rice">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="pie">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:ID="tea">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="cookie">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="bife">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="pizza">
    <rdfs:subClassOf>
            <owl:Class rdf:ID="meal"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="cooks">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#ingredient"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="cooks_Inv">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="#cook"/>
    <owl:inverseOf rdf:resource="#cooks"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="sells">
    <rdfs:domain rdf:resource="#seller"/>
    <rdfs:range rdf:resource="#meal"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="sells_Inv">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="#seller"/>
    <owl:inverseOf rdf:resource="#sells"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="order">
    <rdfs:domain rdf:resource="#buyer"/>
    <rdfs:range rdf:resource="#meal"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="order_Inv">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="#buyer"/>
    <owl:inverseOf rdf:resource="#order"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="follows">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#recipe"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="follows_Inv">
    <rdfs:domain rdf:resource="#recipe"/>
    <rdfs:range rdf:resource="#cook"/>
    <owl:inverseOf rdf:resource="#follows"/>
</owl:ObjectProperty>
<owl:TransitiveProperty rdf:ID="person_dependsOn_Society">
    <rdfs:domain rdf:resource="#person"/>
    <rdfs:range rdf:resource="#Society"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="cooks_dependsOn_person">
    <rdfs:domain rdf:resource="#cook"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="sells_dependsOn_person">
    <rdfs:domain rdf:resource="#seller"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="meal_dependsOn_ingredient">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="#ingredient"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="order_dependsOn_person">
    <rdfs:domain rdf:resource="#buyer"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="recipe_dependsOn_Society">
    <rdfs:domain rdf:resource="#recipe"/>
    <rdfs:range rdf:resource="#Society"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
```

```
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="ingredient_dependsOn_Society">
   <rdfs:domain rdf:resource="#ingredient"/>
   <rdfs:range rdf:resource="#Society"/>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="meal_dependsOn_cooks">
   <rdfs:domain rdf:resource="#meal"/>
   <rdfs:range rdf:resource="#cook"/>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:ObjectProperty rdf:ID="hasPart">
   <owl:inverseOf>
         <owl:TransitiveProperty rdf:ID="partOf"/>
   </owl:inverseOf>
</owl:ObjectProperty>
<owl:TransitiveProperty rdf:ID="partOf">
   <owl:inverseOf rdf:resource="#hasPart"/>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:ObjectProperty rdf:ID="hasPart_directly">
   <rdfs:subPropertyOf>
         <owl:ObjectProperty rdf:ID="hasPart"/>
   </rdfs:subPropertyOf>
   <owl:inverseOf>
         <owl:TransitiveProperty rdf:ID="partOf_directly"/>
   </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="partOf_directly">
   <rdfs:subPropertyOf>
         <owl:ObjectProperty rdf:ID="partOf"/>
   </rdfs:subPropertyOf>
   <owl:inverseOf rdf:resource="#hasPart_directly"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="name">
   <rdfs:domain rdf:resource="#person"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="variation">
   <rdfs:domain rdf:resource="#meal"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="price">
```

```
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="quality">
    <rdfs:domain rdf:resource="#meal"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="quantity">
    <rdfs:domain rdf:resource="#ingredient"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="method of preparation">
    <rdfs:domain rdf:resource="#ingredient"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="flavour">
    <rdfs:domain rdf:resource="#ingredient"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="title">
    <rdfs:domain rdf:resource="#recipe"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
</rdf:RDF>
```

Appendix B: Transformation B

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.owl-ontologies.com/Ontological_Diagram_4.owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://www.owl-ontologies.com/Ontological_Diagram_4.owl">
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="physical activity">
    <rdfs:subClassOf>
```

```
            <owl:Restriction>
                <owl:onProperty>
                    <owl:ObjectProperty rdf:ID="hasPart"/>
                </owl:onProperty>
                <owl:allValuesFrom>
                    <owl:Class rdf:ID="physical exercise"/>
                </owl:allValuesFrom>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="improve of health"/>
    <owl:Class rdf:ID="disease"/>
    <owl:Class rdf:ID="physical exercise"/>
    <owl:Class rdf:ID="well-being"/>
    <owl:Class rdf:ID="habit"/>
    <owl:Class rdf:ID="lost of weight"/>
    <owl:Class rdf:ID="health"/>
    <owl:Class rdf:ID="person">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="Agent"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Agent"/>
    <owl:Class rdf:ID="Society">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="Agent"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="physical educator">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="person"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="Health Community Agent ">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="person"/>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="caregiver">
        <rdfs:subClassOf>
            <owl:Class rdf:ID="person"/>
        </rdfs:subClassOf>
    </owl:Class>
```

```
<owl:Class rdf:ID="walking">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="physical activity"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="bodybuilding">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="physical activity"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="elongation">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="physical activity"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="soccer">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="physical activity"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="running">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="physical activity"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ciclying">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="physical activity"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="doctor">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="health community agent"/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="nurse">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="Health Community Agent "/>
   </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="nutritionist">
   <rdfs:subClassOf>
         <owl:Class rdf:ID="health community agent"/>
```

```
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="dentist">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="health community agent"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="sports practice">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="habit"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="alimentation">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="habit"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="hygiene">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="habit"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="meditation">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="habit"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="smile">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="habit"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="wash the hands">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="habit"/>
    </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="guides">
    <rdfs:domain rdf:resource="#physical educator"/>
    <rdfs:range rdf:resource="#physical activity"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="guides_Inv">
    <rdfs:domain rdf:resource="#physical activity"/>
```

```
    <rdfs:range rdf:resource="#physical educator"/>
    <owl:inverseOf rdf:resource="#guides"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="promotes">
    <rdfs:domain rdf:resource="#health community agent"/>
    <rdfs:range rdf:resource="#habit"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="promotes_Inv">
    <rdfs:domain rdf:resource="#habit"/>
    <rdfs:range rdf:resource="#health community agent"/>
    <owl:inverseOf rdf:resource="#promotes"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="cares">
    <rdfs:domain rdf:resource="#caregiver"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="prevents">
    <rdfs:domain rdf:resource="#habit"/>
    <rdfs:range rdf:resource="#disease"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="prevents_Inv">
    <rdfs:domain rdf:resource="#disease"/>
    <rdfs:range rdf:resource="#habit"/>
    <owl:inverseOf rdf:resource="#prevents"/>
</owl:ObjectProperty>
<owl:TransitiveProperty rdf:ID="guides_dependsOn_person">
    <rdfs:domain rdf:resource="#physical educator"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="person_dependsOn_Society">
    <rdfs:domain rdf:resource="#person"/>
    <rdfs:range rdf:resource="#Society"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="physical activity_dependsOn_Society">
    <rdfs:domain rdf:resource="#physical activity"/>
    <rdfs:range rdf:resource="#Society"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="promotes_dependsOn_person">
    <rdfs:domain rdf:resource="#health community agent"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
```

```
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="improve of health_dependsOn_promotes">
    <rdfs:domain rdf:resource="#improve of health"/>
    <rdfs:range rdf:resource="#health community agent"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="improve of health_dependsOn_physical exercise">
    <rdfs:domain rdf:resource="#improve of health"/>
    <rdfs:range rdf:resource="#physical exercise"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="habit_dependsOn_person">
    <rdfs:domain rdf:resource="#habit"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="well-being_dependsOn_habit">
    <rdfs:domain rdf:resource="#well-being"/>
    <rdfs:range rdf:resource="#habit"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="lost of weight_dependsOn_habit">
    <rdfs:domain rdf:resource="#lost of weight"/>
    <rdfs:range rdf:resource="#habit"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="cares_dependsOn_person">
    <rdfs:domain rdf:resource="#caregiver"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="health_dependsOn_person">
    <rdfs:domain rdf:resource="#health"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="well-being_dependsOn_health">
    <rdfs:domain rdf:resource="#well-being"/>
    <rdfs:range rdf:resource="#health"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="lost of weight_dependsOn_physical activity">
    <rdfs:domain rdf:resource="#lost of weight"/>
```

```
    <rdfs:range rdf:resource="#physical activity"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="disease_dependsOn_person">
    <rdfs:domain rdf:resource="#disease"/>
    <rdfs:range rdf:resource="#person"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:TransitiveProperty rdf:ID="physical exercise_ dependsOn_ physical activity">
    <rdfs:domain rdf:resource="#physical exercise"/>
    <rdfs:range rdf:resource="#physical activity"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:ObjectProperty rdf:ID="hasPart">
    <owl:inverseOf>
            <owl:TransitiveProperty rdf:ID="partOf"/>
    </owl:inverseOf>
</owl:ObjectProperty>
<owl:TransitiveProperty rdf:ID="partOf">
    <owl:inverseOf rdf:resource="#hasPart"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
<owl:ObjectProperty rdf:ID="hasPart_directly">
    <rdfs:subPropertyOf>
            <owl:ObjectProperty rdf:ID="hasPart"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf>
            <owl:TransitiveProperty rdf:ID="partOf_directly"/>
    </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="partOf_directly">
    <rdfs:subPropertyOf>
            <owl:ObjectProperty rdf:ID="partOf"/>
    </rdfs:subPropertyOf>
    <owl:inverseOf rdf:resource="#hasPart_directly"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#person"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="intensity">
    <rdfs:domain rdf:resource="#physical exercise"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="caloric expenditure">
        <rdfs:domain rdf:resource="#lost of weight"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>
</rdf:RDF>
```