# WOP 2014
## 5th Workshop on Ontology and Semantic Web Patterns

Co-located with ISWC2014
Riva del Garda, Italy - October 19th 2014

**Edited By:**
**Victor de Boer**, VU University Amsterdam, NL
**Aldo Gangemi**, Université Paris 13, FR
**Krzysztof Janowicz**, University of California, USA
**Agnieszka Ławrynowicz**, Poznan University of Technology, PL

# Preface

The 5th edition of the Workshop on Ontology and Semantic Web Patterns (WOP2014) was be the very first in Europe, in which traditionally the design pattern community for Semantic Web and Linked Data had been very strong. The aim of the workshop was twofold: (i) providing an arena for proposing and discussing good practices, patterns, pattern-based ontologies, systems etc., and (ii) broadening the pattern community that is developing its own language for discussing and describing relevant problems and their solutions.

WOP2014 was a full-day workshop, co-located with the 13th International Semantic Web Conference, that included an invited talk, paper presentations and posters. The invited talk was given by Valentina Presutti and was entitled "Fueling the future with Semantic Web Patterns". Altogether, WOP2014 received 10 research paper submissions and 2 pattern paper submissions. From among these submissions, the Program Committee selected 6 research papers and 2 pattern papers for the presentation at the workshop. The poster session included 4 posters (2 of them presented pattern papers, and the remaining 2 presented patterns that were described within research papers).

This year's Workshop on Ontology and Semantic Web Patterns offered a fast-track submission of selected papers to the Semantic Web journal's special issue on ontology design patterns. Authors of best papers were invited to submit a revised and extended version of their work to the journal. Based on the reviewer scores, the Organization Committee decided to invite two papers. To ensure objectivity, this decision was verified with the members of the Steering Committee.

We thank the Program Committe and the Steering Commitee for their hard work, and the authors for submitting their papers and for addressing the reviewers comments. We thank our invited speaker, Valentina Presutti, for the interesting and inspiring talk. We also thank the organisers of the 13th International Semantic Web Conference for hosting WOP2014 at ISWC2014.

Further information about the Workshop on Ontology and Semantic Web Patterns can be found at: `http://ontologydesignpatterns.org/wiki/WOP:2014`.

**November 2014**

WOP Chairs

Victor de Boer
Aldo Gangemi
Krzysztof Janowicz
Agnieszka Ławrynowicz

# Fueling the future with Semantic Web Patterns

Valentina Presutti[1,2]

[1] Semantic Technology Laboratory of the National Research Council (CNR), Rome, Italy
[2] University Paris 13 - CNRS, Paris, France

**Abstract.** I will claim that Semantic Web Patterns can drive the next technological breakthrough: they can be key for providing intelligent applications with sophisticated ways of interpreting data. I will picture scenarios of a possible not so far future in order to support my claim. I will argue that current Semantic Web Patterns are not sufficient for addressing the envisioned requirements, and I will suggest a research direction for fixing the problem, which includes the hybridization of existing computer science pattern-based approaches, and human computing.

# A pattern-based ontology for describing publishing workflows

Aldo Gangemi[1,2], Silvio Peroni[1,3],
David Shotton[4], and Fabio Vitali[3]

[1] STLab-ISTC, Consiglio Nazionale delle Ricerche (Italy)
aldo.gangemi@cnr.it
[2] Laboratoire d'Informatique de Paris Nord, Université Paris 13 (France)
[3] Department of Computer Science and Engineering, University of Bologna (Italy)
silvio.peroni@unibo.it, fabio.vitali@unibo.it
[4] Oxford e-Research Centre, University of Oxford (UK)
david.shotton@oerc.ox.ac.uk

**Abstract.** In this paper we introduce the *Publishing Workflow Ontology* (*PWO*), i.e., an OWL 2 DL ontology for the description of generic workflows that is particularly suitable for formalising typical publishing processes such as the publication of articles in journals. We support the presentation with a discussion of all the ontology design patterns that have been reused for modelling the main characteristics of workflows.

**Keywords:** PWO, ODP, publishing process, workflow description

## 1 Introduction

Keeping track of publication processes is a crucial task for publishers. This activity allows them to produce statistics on their goods (e.g., books, authors, editors) and to understand whether and how their production changes over time. Organisers of particular events, such as academic conferences, have similar needs. Tracking the number of submissions in the current edition of a conference, the number of accepted papers, the review process, etc., are important statistics that can be used to improve the review process in future editions of the conference.

Some communities have started to publish data, e.g., the Semantic Web Dog Food[5] and the Semantic Web Journal[6], which describe those scholarly data as RDF statements in the Linked Data, in order to allow software agents and applications to check and reason on them, and to infer new information. However, the description of processes, for instance the peer-review process or the publishing process, is something that is not currently handled – although sources of related raw data exist (e.g., EasyChair metadata). Furthermore, having these types of data publicly available would increase the transparency of the aforementioned processes and allow their use for statistical analysis. Of course, a model

[5] Semantic Web Dog Food: http://data.semanticweb.org.
[6] Semantic Web Journal: http://semantic-web-journal.com.

for describing these data is needed. Moreover, the model should be easy to integrate and adapt according to the needs and constraints of different domains (publishing, academic conferences, research funding, etc.).

In this paper we introduce the *Publishing Workflow Ontology* (*PWO*), that we developed in order to accommodate the aforementioned requirements. This ontology is one of the *Semantic Publishing and Referencing* (*SPAR*) *Ontologies*[7] (which have been created for the description of different aspects of the publishing domain), and allows one to describe the logical steps in a workflow, as for example the process of publication of a document. Each step may involve one or more events that take place at a particular phase of the workflow (e.g., authors are writing the article, the article is under review, a reviewer suggests to revise the article, the article is in printing, the article has been published, etc.). This ontology has been developed in order to allow its use with other SPAR Ontologies as well as other models and existing data.

The rest of the paper is organised as follows. In Section 2 we discuss some related works on workflows within the Semantic Web domain. In Section 3 we provide the definitions of workflow we have used as starting point for modelling our ontology, and discuss the use of some existing ontology design patterns for addressing the modelling issues related to the main characteristics of workflows. In Section 4 we introduce PWO, describing how it extends the aforementioned patterns in order to handle the main components of workflows, and we support the discussion by means of a real example of publication process of an article of the Semantic Web Journal. Finally, in Section 5 we conclude the paper sketching out some future works.

## 2 Workflows and the Semantic Web

In the last years the Semantic Web community have started on working and proposing models for the formalisation and description of generic workflows, and have shown several applications of these models/theories within the publishing domain. Maybe the first huge-impact project on these topic has been Workflow 4ever (STREP FP7-ICT-2007-6 270192)[8] [8]. This project addresses challenges related to the preservation of scientific experiments through the definition of models and ontologies for describing scientific experiments, to the collection of best practices for the creation and management of *Research Objects*[9] [2], and to the analysis and management of decay in scientific workflows.

As already stated, one of the outcomes of the project has been the proposal for workflow-centric Research Objects [1], i.e., an OWL ontology[10] for linking together scientific workflows, the provenance of their executions, interconnections between workflows and related resources (e.g., datasets, publications, etc.), and social aspects related to such scientific experiments.

[7] SPAR Ontologies website: http://purl.org/spar.
[8] Workflow 4ever project homepage: http://www.wf4ever-project.org.
[9] Research Object website: http://www.researchobject.org.
[10] Research Object OWL ontology: http://purl.org/wf4ever/ro.

Another interesting proposal for describing workflows is the work done by Garijo and Gil [7]. In this work, they describe a framework to publish *computational* workflows, which includes the specification a particular OWL ontology, i.e., the *Open Provenance Model for Workflows* (*OPMW*)[11], for the description of workflow traces and their templates. Along the lines of the aforementioned work, the same authors recently published the *Ontology for Provenance and Plans* (*P-Plan*)[12]. P-Plan is an OWL 2 DL ontology that extends the *Provenance Ontology* [12] in order to represent the plans that guided the execution of scientific processes, describing how such plans are composed and their correspondence to provenance records that describe the execution itself.

Finally, among the other proposals for describing workflows, it worths mentioning the OWL ontology proposed by Sebastian *et al.* [18] for describing generic workflows, which reuses existing ontologies such as the *Change and Annotations Ontology* (*ChAO*) [13], and the *SCUFL2 Core ontology*[13] that has been used to describe workflows in *Taverna*[14], an open source and domain-independent Workflow Management System [19].

## 3 Foundational material: design patterns

In order to design an ontology for modelling (publishing) workflows, we have to understand what are the minimal characteristics that such ontology should address and if we can reuse some existing modelling solutions. Oxford Dictionaries defines workflow as follows:

> "The sequence of industrial, administrative, or other processes through which a piece of work passes from initiation to completion."[15]

From this definition it is possible to identify some important characteristics of any workflow, i.e., the fact that it *involves* a *sequence* of processes that allow to initiate and then complete a piece of work during a specifiable *time interval*. The definition of the SearchCIO website is still more specific:

> "Workflow is a term used to describe the tasks, procedural steps, organizations or people involved, required input and output information, and tools needed for each step in a business process."[16]

From this definition we can spot other crucial aspects. First of all, its structural organisation in procedural steps, each of them *describes* tasks performed by organisations and people, and each step *requires* some input information and tools in order to produce an output. Using these two definition as input, we

[11] Open Provenance Model for Workflows: http://www.opmw.org/ontology/.
[12] Ontology for Provenance and Plans: http://purl.org/net/p-plan#.
[13] http://ns.taverna.org.uk/2010/scufl2
[14] http://www.taverna.org.uk
[15] http://www.oxforddictionaries.com/definition/english/workflow
[16] http://searchcio.techtarget.com/definition/workflow

can identify some well-known ontological patterns that already address, from an abstract point of view, some of the aspects related of workflows.

**Participation.** The *participation pattern*[17] is a simple pattern that allows us to describe processes, events, or states (through the class *Event*), and to specify the various objects (through the class *Object*) that participate in these events.

This pattern seems to be very useful to define workflows as events involving people, organisations, places, and other objects as participants, as well as to link workflows and related activities to the expected steps .

**Sequence.** The *sequence pattern*[18] is another pattern that can be used between tasks, processes or time intervals, in order to define sequences of such objects through direct (i.e., *directlyFollows* and *directlyPrecedes*) and transitive relations (i.e., *follows* and *precedes*). It is, of course, very useful to describe the logical organisation of the various steps of a workflow.

**Control flow and plan execution.** The *control flow* pattern[19] is an OWL representation of some of the constructs defined in the Workflow Patterns[20] by Wil van der Alst (cf. [17]). Either action or control (e.g., branching, concurrency, looping) tasks are represented and related by means of the *sequence pattern. Tasks* are distinct from *activities*, which are supposed to be executed based on the task structure. This link is made in the context of the *basic plan description*[21] and the *basic plan execution*[22] patterns, which reuse the foundational *descriptions and situations pattern* to relate task compositions (*plans*) to organised activities (*plan executions*). A comprehensive presentation is provided in [6].

These patterns are of course, very useful to describe the kinds of steps (the term used here for *tasks*) in a workflow and in general in publishing workflows. The action and control tasks from the control flow pattern are not specialised in the publishing workflow pattern, because they are expected to work as they are (by typing the steps according to their workflow semantics) when the need for control flows emerges in a planned workflow.

**Time-indexed situation.** The *time-indexed situation pattern*[23] allows the description of a situation (i.e., the class *TimeIndexedSituation*) – i.e., a view on a set of entities linked to it through the property *isSettingFor* – that is explicitly indexed at some time specifiable through the property *atTime* linking a time interval (i.e., an instance of the class *TimeInterval*).

This pattern can be used to describe steps from an abstract point of view as kinds of situations representing the settings for all the events and input/output material needed or produced by these steps. Notice that time-indexed situation combines perfectly with plan execution in order to provide a temporal ordering to activities organised into a plan.

[17] http://www.ontologydesignpatterns.org/cp/owl/participation.owl
[18] http://www.ontologydesignpatterns.org/cp/owl/sequence.owl
[19] http://www.ontologydesignpatterns.org/cp/owl/controlflow.owl
[20] The Workflow Patterns page is: http://www.workflowpatterns.com.
[21] http://www.ontologydesignpatterns.org/cp/owl/basicplandescription.owl
[22] http://www.ontologydesignpatterns.org/cp/owl/basicplanexecution.owl
[23] http://www.ontologydesignpatterns.org/cp/owl/timeindexedsituation.owl

**Error Ontology.** The *Error Ontology*[24] is a unit test that produces an inconsistent model if a particular (and incorrect) situation happens. It works by means of a data property, *error:hasError*, that denies its usage for any resource, as shown as below (in Manchester Syntax [9]):

```
DataProperty: error:hasError
  Domain: error:hasError exactly 0    Range: xsd:string
```

A resource that has an error makes the ontology inconsistent, since its domain is "all those resources that do not have any *error:hasError* assertion".

This model is very useful in our context in order to define constraints on the input/output objects needed by the steps of a workflow. For instance, we could use it to deny the use of a certain object as input of a step if it will be produced only as output of one of the following steps.

## 4 PWO: the Publishing Workflow Ontology

In order to accommodate workflow requirements, we developed the *Publishing Workflow Ontology*[25] (*PWO*), which is entirely based on the ontology patterns introduced in Section 3. This ontology allows one to describe the logical steps in a workflow, as for example the process of publication of a document. Each step may involve one or more events (or actions) that take place to a particular phase of the workflow (e.g., authors are writing the article, the article is under review, a reviewer suggests to revise the article, the article is in printing, the article has been published, etc.).

As shown in Fig. 1, PWO is based on two main classes, which are:

- class *pwo:Workflow*. It represents a sequence of connected tasks (i.e., steps) undertaken by the agents; it is a subclass of *plan:PlanExecution*[26];
- class *pwo:Step*. It is an atomic unit of a workflow, subclass of *taskrole:Task*; it is characterised by a (required) starting time and an ending time, and it is associated with one or more events (activities) that are executed within the step. A workflow step usually involves some input information, material or energy needed to complete the step, and some output information, material or energy produced by that step. In the case of a publishing workflow, a step typically results in the creation of a publication entity, usually by the modification of another pre-existing publication entity, e.g., the creation of an edited paper from a rough draft, or of an HTML representation from an XML document.

[24] http://www.essepuntato.it/2009/10/error

[25] http://purl.org/spar/pwo

[26] Note that in PWO we are not using explicitly the separation between workflow definition and workflow execution, since PWO has been thought as an ontology to provide a retrospective description of running workflows. Even if this is a simplification of the whole approach described by the imported patterns, we decided to include both patterns for workflow definition and execution in order to handle even workflow definitions in case we may need it (even if we have not yet explored this use of PWO properly).

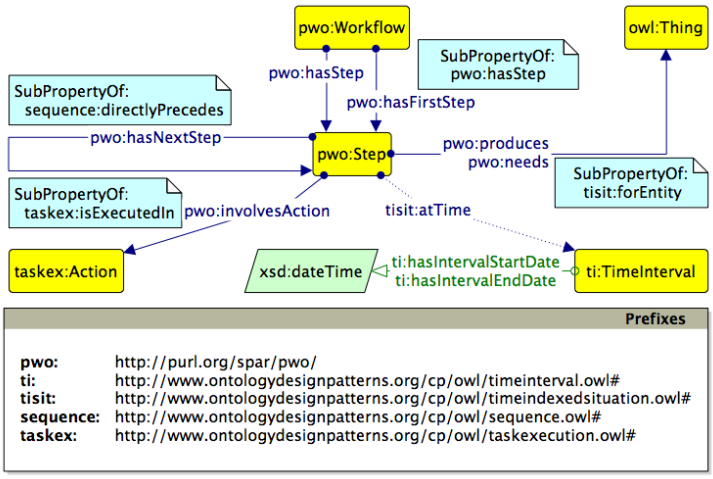


**Fig. 1.** Graffoo representation [5] of the Publishing Workflow Ontology (PWO).

PWO was implemented according to the aforementioned ontology patterns. As shown in Table 1, such patterns have been used as follows:

- *plan execution* to describe workflows as plans, and their executions;
- *time-indexed situation* to describe workflow steps as entities that involve a duration and that are characterised by events and objects (needed for and produced by the step);
- *sequence* to define the order in which steps appear within a workflow;
- *control flow* to describe the specialization and nature of steps at planning time;
- *participation* to describe events (and eventually agents involved) taking part in the activities carried out according to the steps.

In addition, by means of the Error Ontology, we can generate an inconsistency every time the steps of a workflow are not arranged in a correct temporal order. In particular, an error is raised when a step requires (property *pwo:needs*) to use a particular object that will be produced (property *pwo:produces*) as consequence of another sequent step. The following excerpt shows the implementation of this constraint through a SWRL rule [10]:

```
Step(?step1) , Step(?step2) , needs(?step1,?resource) ,
produces(?step2,?resource) , sequence:precedes(?step1,?step2)
  -> error:hasError(?step1,"A step cannot need a resource that will be
       produced by a following step"^^xsd:string)
```

In the next subsections we show how to describe the process of publication of a journal article step by step. In particular we introduce how PWO can be used in combination with existing data of the *Semantic Web Journal*[27] [11] and other SPAR ontologies, such as PSO [15], C4O [4], FaBiO and CiTO [14].

[27] Semantic Web Journal data: http://semantic-web-journal.com/sejp.

**Table 1.** A summary of all the entities of PWO and their relations with the original pattern-based entities.

| PWO entity | Pattern entity | Description |
|---|---|---|
| Workflow | Plan (plan execution) | The class of particular situation types describing a real-life work, composed by a sequence of steps |
| Step | Task (task role, via control flow) | The class describing specific tasks that form the workflow and that are done within particular time intervals |
| hasStep | definesTask (basic plan description) | The relation linking a workflow to a component step |
| hasFirstStep | definesTask (basic plan description) | A sub-property of *hasStep* which identifies the starting step of a workflow |
| hasNextStep | directlyPrecedes (sequence) | An object property linking a step in a workflow with the step that directly follows it |
| hasPreviousStep | directlyFollows (sequence) | An object property linking a step in a workflow with the step that directly precedes it |
| involvesAction | isExecutedIn (task execution) | The object property linking a step in a workflow to an activity done in the context of that step |
| needs | forEntity (time-indexed situation) | The object property linking a workflow step to anything required to undertake that step |
| produces | forEntity (time-indexed situation) | The object property linking a workflow step to the thing that the step produces, creates or results in |

### 4.1 A typical publishing workflow of a journal article

From a pure publisher's perspective, the first step of any workflow that brings to a new journal publication starts with a formal submission of a manuscript performed by someone, hereinafter the *author*. This activity expresses, at the same time, interest on the topics of the journal and may acknowledge, indirectly, the quality of the journal itself – since authors (usually) would like to publish articles in a venue that they consider respectful and qualitatively worth for different reasons (e.g., quality of reviews, journal impact factor, definite timing of the publishing process). Then, in the next step, i.e., the reviewing phase, the person (designated by the publisher) in charge of the quality of submitted material, hereinafter the *editor*, invites other people (hereinafter the *reviewers*) for assessing the quality of the submitted manuscript. The opinions returned by the reviewers to the editor are the fundamental input that the editor will use to decide upon the fate of the manuscript during the next step, i.e., the decision phase. Finally, if the manuscript have been considered worth of publication in the present form, the editor will acknowledge the author of the acceptance of his/her work – and the next steps of the workflow will be in charge of the publisher itself. Otherwise, if the article is not ready for being published, the editor either may ask for its rejection, thus finishing the workflow, or (s)he can return a list of issues to be addressed to the author in order to deserve publication. In this latter case, the revision phase will start and the author will revise the paper according to reviewers' comments and editor's suggestions, and thus the workflow will continue with a new submission phase.

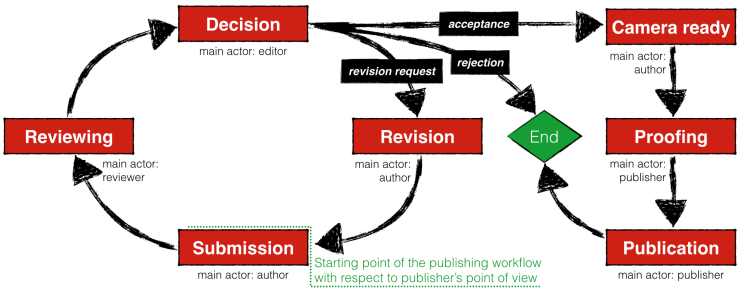


**Fig. 2.** A diagram describing the typical publishing workflow of a journal article – note that it does not take into account any withdrawing action by the author, nor any comment made by users on publisher's website before/after article publication.

The whole publishing workflow we have described (summarised in Fig. 2) can be formally represented by means of PWO. In the following excerpt (in Turtle [16]) we create an instance of the class *pwo:Workflow* as composed by a definite (but not specified, in this example) number of steps[28]:

```
:workflow a pwo:Workflow ;
  pwo:hasFirstStep :step-one ;
  pwo:hasStep :step-two , :step-three , :step-four, ... .
```

In the next sections we show how to describe the first four steps of such workflow by taking into account real publication data available in the Semantic Web Journal Linked Data repository concerning [3].

### 4.2 Submission

The first step of the workflow concerned the submission of a manuscript by one of its authors, in this case Paolo Ciccarese. Thus, the manuscript received the status of "submitted" and it was made available to the journal editor and the reviewers for the next step of the workflow. In order to describe all these aspects concerning the first step, we use several entities defined in the ontology patterns imported by PWO, as well as a number of other entities from another SPAR ontology, i.e., the *Publishing Status Ontology* (*PSO*)[29] [15]. This is an ontology for describing the status held by a document or other publication entity at each of the various stages in the publishing process. In addition, existing entities of the Semantic Web Journal Linked Data repository (e.g., people and manuscripts) are reused in order to demonstrate the flexibility of PWO in working with other existing models and data, as shown as follows:

```
:step-one a pwo:Step ; # Submission step
  pwo:involvesAction :submission-action ; tisit:atTime [ a ti:TimeInterval ;
    ti:hasIntervalStartDate "2013-01-21T10:08:28"^^xsd:dateTime ;
    ti:hasIntervalEndDate "2013-01-21T10:08:28"^^xsd:dateTime ] ;
```

[28] Prefixes available at http://www.essepuntato.it/2014/wop/prefixes.ttl.
[29] http://purl.org/spar/pso

```
  pwo:needs swj-node:432 ; pwo:produces :submitted-status ;
  pwo:hasNextStep :step-two .
# The event in which one of the authors submits the manuscript
:submission-action a taskex:Action ;
  dcterms:description "Paolo Ciccarese submits the paper" ;
  part:hasParticipant swj:paolo-ciccarese , swj-node:432 .
# The new status 'submitted' associated to the paper after the submission
:submitted-status a pso:StatusInTime ;  pso:isStatusHeldBy swj-node:432 ;
  pso:isAcquiredAsConsequenceOf :submission-action ;
  pso:withStatus pso:submitted ; tvc:atTime [ a ti:TimeInterval ;
    ti:hasIntervalStartDate "2013-01-21T10:08:28"^^xsd:dateTime ] .
```

### 4.3 Reviewing

The step regarding the reviewing phase began with the activity of the editor, Giancarlo Guizzardi, of looking for appropriate reviewers for the paper. Once found, the reviewers were provided with the manuscript, reviewed it, and wrote down their comments that were finally sent back to the editor. In order to describe all the aspects concerning the second step, we use several entities defined in additional SPAR ontologies, i.e., the *Citation Counting and Context Characterisation Ontology* (*C4O*)[30] [4] the *Citation Typing Ontology* (*CiTO*)[31] [14], in order to express the content of reviews and to explicitly link those to the manuscript they reviewed. In the following excerpt we introduce the formalisation in PWO of the second step of the workflow:

```
:step-two a pwo:Step ; pwo:hasNextStep :step-three ; # Reviewing step
  pwo:involvesAction :choosing-reviewers-action ,
    :reviewing-action , :reviews-notification-sending-action ;
  tisit:atTime [ a ti:TimeInterval ;
    ti:hasIntervalStartDate "2013-02-18T17:04:32"^^xsd:dateTime ;
    ti:hasIntervalEndDate "2013-04-01T05:53:24"^^xsd:dateTime ] ;
  # The review process can start only when a manuscript has been submitted
  pwo:needs swj-node:432 , :submitted-status ;
  pwo:produces :review-1, :review-2, :under-review-status, :reviewed-status.
:choosing-reviewers-action a taskex:Action ;
  dcterms:description "The editor, Giancarlo Guizzardi, chooses Csaba Veres
      and Fernando Naufel do Amaral as reviewers of the manuscript" ;
  part:hasParticipant swj:csaba-veres , swj:fernando-naufel-do-amaral ,
    swj:giancarlo-guizzardi , swj-node:432 .
:reviewing-action a taskex:Action ;
  dcterms:description "Reviewers review the manuscript" ;
  part:hasParticipant
    swj:csaba-veres , swj:fernando-naufel-do-amaral , swj-node:432 .
:reviews-notification-sending-action a taskex:Action ;
  dcterms:description "The reviews are sent to the editor" ;
  part:hasParticipant swj:csaba-veres , swj:fernando-naufel-do-amaral ,
    :review-1 , :review-2 , swj:giancarlo-guizzardi .
:review-1 a fabio:Comment ; # Review 1 by Csaba Veres
  frbr:realizationOf [ a fabio:Review ] ;
  cito:reviews swj-node:432 ; frbr:realizer swj:csaba-veres ;
  c4o:hasContent "The paper addresses a very practical..." .
:review-2 a fabio:Comment ; # Review 2 by Fernando Naufel do Amaral
  frbr:realizationOf [ a fabio:Review ] ; cito:reviews swj-node:432 ;
  frbr:realizer swj:fernando-naufel-do-amaral ;
  c4o:hasContent "The paper presents the Collection Ontology (CO)..." .
# The paper has been assigned to the under-review status for a while
:under-review-status a pso:StatusInTime ; pso:isStatusHeldBy swj-node:432 ;
```

[30] http://purl.org/spar/c4o
[31] http://purl.org/spar/cito

```
  pso:isAcquiredAsConsequenceOf :reviewing-action ;
  pso:isLostAsConsequenceOf :reviews-notification-sending-action ;
  pso:withStatus pso:under-review ; tvc:atTime [a ti:TimeInterval ;
    ti:hasIntervalStartDate "2013-02-26T12:00:07"^^xsd:dateTime ;
    ti:hasIntervalEndDate "2013-04-01T05:53:24"^^xsd:dateTime ] .
# The paper status has changed in 'reviewed' after reviewers' comments
:reviewed-status a pso:StatusInTime ; pso:isStatusHeldBy swj-node:432 ;
  pso:isAcquiredAsConsequenceOf :reviews-notification-sending-action ;
  pso:withStatus pso:reviewed ; tvc:atTime [a ti:TimeInterval ;
    ti:hasIntervalStartDate "2013-04-01T05:53:24"^^xsd:dateTime ] .
```

### 4.4 Decision

During the third step, the editor was responsible for the fate of the paper and provided a decision for it according to reviewers' comments. Once formalised the decision, a decision letter was sent by email to the corresponding author (i.e., Paolo Ciccarese) and the status of the paper changed in then in "minor revision". In the following excerpt we introduce the formalisation in PWO of the third step of the workflow:

```
:step-three a pwo:Step ; pwo:hasNextStep :step-four ; # Notification step
  pwo:involvesAction :decision-action , :notification-action ;
  tisit:atTime [ a ti:TimeInterval ;
    ti:hasIntervalStartDate "2013-04-01T05:53:24"^^xsd:dateTime ;
    ti:hasIntervalEndDate "2013-06-10T17:47:53"^^xsd:dateTime ] ;
  pwo:needs swj-node:432 , :review-1 , :review-2 ;
  pwo:produces :minor-revision-status , :decision-letter .
:decision-action a taskex:Action ;
  dcterms:description "The editor decides for acceptance or not" ;
  part:hasParticipant
    swj:giancarlo-guizzardi , :review-1 , :review-2 ,swj-node:432 .
:notification-action a taskex:Action ;
  dcterms:description "The editor notifies his decision to the corresponding
      author (i.e., Paolo Ciccarese)." ;
  part:hasParticipant swj:giancarlo-guizzardi , :decision-letter ,
    :review-1 , :review-2 , swj:paolo-ciccarese , swj-node:432 .
# The decision letter written by the editor
:decision-letter a fabio:Letter , fabio:Email ;
  frbr:realizationOf [ a fabio:Opinion ] cito:citesAsRelated swj-node:432 ;
  frbr:realizer swj:giancarlo-guizzardi ;
  c4o:hasContent "Dear authors, Thank you for your interest in..." .
# The minor revision status assigned to the paper after editor's decision
:minor-revision-status a pso:StatusInTime ; pso:isStatusHeldBy swj-node:432 ;
  pso:isAcquiredAsConsequenceOf :decision-action ;
  pso:withStatus swj:minorRevision ; tvc:atTime [a ti:TimeInterval ;
    ti:hasIntervalStartDate "2013-06-10T17:47:53"^^xsd:dateTime ] .
```

### 4.5 Revision

During the fourth step, the authors worked in order to revise the content of the previous version of the paper according to reviewers' comments and editor's suggestions. At the end of this step, the main result was the creation of a new version of the paper (i.e., *swj-node:506* in our example) that had to be submitted in the next step. In the following excerpt we introduce the formalisation in PWO of the fourth step of the workflow:

```
:step-four a pwo:Step ; pwo:hasNextStep :step-five ; # Revision step
  pwo:involvesAction :revision-action ; tisit:atTime [ a ti:TimeInterval ;
```

```
    ti:hasIntervalStartDate "2013-06-10T17:47:53"^^xsd:dateTime ;
    ti:hasIntervalEndDate "2013-07-01T05:51:30"^^xsd:dateTime ] ;
  pwo:needs swj-node:432 , :decision-letter , :review-1 , :review-2 ;
  pwo:produces swj-node:506 .
:revision-action a taskex:Action ;
  dcterms:description "The authors revises the paper" ;
  part:hasParticipant swj-node:432 , :decision-letter ,
    :review-1 , :review-2 , swj:silvio-peroni , swj:paolo-ciccarese .
```

## 5 Conclusion

In this paper we introduced the *Publishing Workflow Ontology* (*PWO*), i.e., an OWL 2 DL ontology part of the Semantic Publishing and Referencing (SPAR) Ontologies, which allows the description of publishing workflows in RDF. The whole ontology is entirely based on existing ontology design patterns that allowed us to model the various aspects of workflows in an appropriate and standardised way. We showed a particular use of PWO for describing the first steps of a real publishing workflow concerning the publication of an article of the Semantic Web Journal, i.e., [3], in which we reused entities and data coming from several models and data, e.g., other SPAR ontologies and existing resources from the Semantic Web Journal Linked Dataset.

Although PWO had been thought in principle to describe publishing-related workflows, it has been developed on purpose as an ontology for the description of generic workflows. In future we plan to align it to other workflow-related models, e.g., PROV-O, the Research Object ontology and the other ontologies described in Section 2. In addition, we are currently studying the applicability of PWO in the legal and scientific domains. In particular, we plan to work on its use for describing workflows that concern the process of codification of the laws of the United States legislation and the series of computational or data manipulation steps in scientific applications.

## References

1. Belhajjame, K., Corcho, O., Garijo, D., Zhao, J., Missier, P., Newman, D. R., ... Goble, C. (2012). Workflow-Centric Research Objects: A First Class Citizen in the Scholarly Discourse. In Proceedings of the 2nd Workshop on Semantic Publishing (SePublica 2012). http://ceur-ws.org/Vol-903/paper-01.pdf
2. Belhajjame, K., Zhao, J., Garijo, D., Hettne, K. M., Palma, R., Corcho, O., ... Goble, C. A. (2014). The Research Object Suite of Ontologies: Sharing and Exchanging Research Data and Methods on the Open Web. The Computing Research Repository (CoRR), abs/1401.4307. http://arxiv.org/abs/1401.4307
3. Ciccarese, P., & Peroni, S. (2013). The Collections Ontology: creating and handling collections in OWL 2 DL frameworks. Semantic Web. DOI: 10.3233/SW-130121
4. Di Iorio, A., Nuzzolese, A. G., Peroni, S., Shotton, D., & Vitali, F. (2014). Describing bibliographic references in RDF. In Proceedings of 4th Workshop on Semantic Publishing (SePublica 2014). http://ceur-ws.org/Vol-1155/paper-05.pdf
5. Falco, R., Gangemi, A., Peroni, S., & Vitali, F. (2014). Modelling OWL ontologies with Graffoo. In ESWC 2014 Satellite Events - Revised Selected Papers.

6. Gangemi, A., Borgo, S., Catenacci, C., & Lehmann, J. (2004). Task taxonomies for knowledge content. METOKIS Deliverable D7. http://metokis.salzburgresearch.at/files/deliverables/metokis_d07_task_taxonomies_final.pdf
7. Garijo, D., & Gil, Y. (2011). A new approach for publishing workflows: abstractions, standards, and linked data. In Proceedings of the 6th workshop on Workflows in support of large-scale science (WORKS 2011): 47–56. DOI: 10.1145/2110497.2110504
8. Hettne, K., Soiland-Reyes, S., Klyne, G., Belhajjame, K., Gamble, M., Bechhofer, S., ... Corcho, O. (2012). Workflow forever: semantic web semantic models and tools for preserving and digitally publishing computational experiments. In Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences (SWAT4LS 2011): 36–37. DOI: 10.1145/2166896.2166909
9. Horridge, M., & Patel-Schneider, P. F. (2012). OWL 2 Web Ontology Language: Manchester Syntax (Second Edition). W3C Working Group Note, 11 December 2012. http://www.w3.org/TR/owl2-manchester-syntax/
10. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, 21 May 2004. http://www.w3.org/Submission/SWRL/
11. Hu, Y., Janowicz, K., McKenzie, G., Sengupta, K., & Hitzler, P. (2013). A Linked-Data-Driven and Semantically-Enabled Journal Portal for Scientometrics. In Proceedings of the 12th International Semantic Web Conference (ISWC 2013): 114–129. DOI: 10.1007/978-3-642-41338-4_8
12. Lebo, T., Sahoo, S., & McGuinness, D. (2013). PROV-O: The PROV Ontology. W3C Recommendation, 30 April 2013. http://www.w3.org/TR/prov-o/
13. Noy, N. F., Chugh, A., Liu, W., & Musen, M. A. (2006). A Framework for Ontology Evolution in Collaborative Environments. In Proceedings of the 5th International Semantic Web Conference (ISWC 2006): 544–558. DOI: 10.1007/11926078_39
14. Peroni, S., & Shotton, D. (2012). FaBiO and CiTO: Ontologies for describing bibliographic resources and citations. Web Semantics,17: 33–43. DOI: 10.1016/j.websem.2012.08.001
15. Peroni, S., Shotton, D., & Vitali, F. (2012). Scholarly publishing and linked data: describing roles, statuses, temporal and contextual extents. In Proceedings of the 8th International Conference on Semantic Systems (i-Semantics 2012): 9–16. DOI: 10.1145/2362499.2362502
16. Prud'hommeaux, E., & Carothers, G. (2014). Turtle - Terse RDF Triple Language. W3C Recommendation, 25 February 2014. http://www.w3.org/TR/turtle/
17. Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P., & Mulyar, N. (2006). Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22. http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf
18. Sebastian, A., Noy, N. F., Tudorache, T., & Musen, M. A. (2008). A Generic Ontology for Collaborative Ontology-Development Workflows. In Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008): 318–328. DOI: 10.1007/978-3-540-87696-0_28
19. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., ... Goble, C. (2013). The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Research, 41 (W1): W557–W561. DOI: 10.1093/nar/gkt328

# Building ontologies from textual resources: A pattern based improvement using deep linguistic information

Sami Ghadfi, Nicolas Béchet and Giuseppe Berio

IRISA, UMR 6074, Université de Bretagne-Sud, 56017 Vannes, France
sami.ghadfi@gmail.com, nicolas.bechet@irisa.fr, giuseppe.berio@univ-ubs.fr

**Abstract.** Ontologies are a key component for several applications. Ontologies are often built by hand, but automatizing the process of ontology building has been and is even more recognized as very important for scaling and speeding up this process. However, several difficulties have been identified, some of them are quite fundamental. In this paper, we present our work for overcoming some of the fundamental difficulties. Our work resulted in improvements of an existing ontology building tool (*Text2Onto*). The contribution of our work consists in the creation of a flexible language (DTPL—Dependency Tree Patterns Language) for expressing patterns as syntactic dependency trees to extract semantic relations, and making an existing ontology building tool (*Text2Onto*) able to use them. DTPL allows to exploit deep linguistic information (related to co-reference resolutions, conjunctions, appositions, passive verbal phrases, etc.) provided by deep syntactic analysis of the text, and also (in order to improve the accuracy of patterns) to express the exclusion of some dependency bindings in patterns.

**Keywords:** ontology building, semantic relation extraction, dependency tree patterns, deep linguistic information, Text2Onto, DTPL.

## 1 Introduction

Ontologies are a key component for several applications. Ontologies are often built by hand, but automatizing the process of building ontologies has been and is even more recognized as very important for scaling and speeding up this process. Indeed, humans employ texts for providing information directly or indirectly, through the Web for instance. However, unstructured or semi-structured texts do not provide a well-defined semantic structure to be used by machines for reasoning tasks. Ontologies play therefore the key role for representing more explicitly the knowledge hidden in texts. As a consequence, ontologies can be made available for further applications.

Unfortunately, several difficulties concerning automatic ontology building have been identified, some of them are quite fundamental.

Additional arguments suggesting the need for developing complete "Ontology Building Support Systems" (OBSS) can be mentioned. Despite the fact that humans can recognize *ontology artifacts* from terms and sentences (which is enabled by their knowledge of the domain and the contexts on which terms are put together in sentences, suggesting semantic relations between terms), OBSS can supply the frequent

terms and the contexts in which they appear, and systematically apply rules for suggesting how they are related to ontology artifacts. The magnitude of these terms and contexts makes their identification a task more suitable for machines than humans. In addition, ontologies evolve and these evolutions should be supported by automated systems.

For ontology building, there are two main challenges to be taken into account, which correspond to the basic building blocks of any ontology:

- The extraction of concepts and their possible instances: it is a task in which we further distinguish between the extraction of the concept/instance itself, and naming it;
- The extraction of semantic relations (hierarchical and non-hierarchical): it is a task in which we distinguish between identifying the relation occurrence (for example, identifying the relation occurrence "*lion,animal*"), and then identifying the semantic relation to which it belongs ("*lion,animal*" is an occurrence of a hyponymy relation, the whole relation occurrence can be rewritten as *is-hyponym-of(lion,animal)*).

Even if these two challenges are partially connected (i.e. the extraction of relations may impact on already extracted concepts and instances or may lead to additional concepts and instances), in this paper, we concentrate on the second one, i.e. semantic relation extraction. However, as better explained in Section 2, concept/instance extraction and relation extraction can be treated separately. This is also confirmed by the fact that tools used or usable for concept/instance extraction are developed independently for performing well identified tasks such as terminology extraction (possibly comprising disambiguation) and entity identification.

Semantic relation extraction methods can be categorized into two approaches: Pattern based (mainly employing linguistic patterns), and Clustering based (mainly employing clustering and statistical methods). We consider that linguistic patterns are natural and concrete (because close to what humans (can) apply when they manually build ontologies – by following methodologies and design patterns) for improving the overall ontology building process, thus, we have focused on pattern based approaches for relation extraction for the following detailed reasons:

- Patterns represent frequent contexts in which term-pairs related by a given semantic relation tend to appear—the reason for this observation is the way patterns are constructed; very often, this construction begins by specifying seed examples (term-pairs related by a given semantic relation), then looking for the contexts –in sentences– in which they tend to appear together (these contexts can be sequences or sets of words [1,10], or dependency paths in syntactic dependency trees [12], [11]), and then generalizing/merging the most similar contexts or keeping only the most accurate ones (i.e. contexts relating at least a given number of instance examples)—; for instance, the Hearst pattern " X(NP[1]) such as Y(NP) " [5] induces the relation "Y is-hyponym-of X", where the context in this case is the sequence of words "such as".
- Patterns fall into two categories: 1. Reliable patterns (they possess high precision and low recall), 2. Generic patterns (they possess low precision and high recall). One can use the advantages of one category to overcome the drawbacks of the oth-

[1] NP represents a noun phrase.

er. For instance, in [8], the authors have used reliable patterns as a reference to evaluate the relevance of relation occurrences extracted by generic patterns.
- Any extraction method and technique that does not use predefined patterns takes more processing time, because it needs to identify the (frequent) contexts in which terms related by a given relation do appear (for instance, these contexts can be syntactic dependency links that bind individual words in the text—as used in [2] and [9]—, etc.). These methods are based on the distributional hypothesis [4] and its derivations [15], [6], [7].

However, effective usage of patterns within an OBSS remains an open research question. In Section 2, we present the existing methods for pattern-based relation extraction, and also their inherent difficulties (or limitations) preventing to get acceptable ontologies. Section 3 presents the contributions of the paper, i.e. (I) A method for enhancing OBSS with the ability to use deep linguistic information for relation extraction, (II) Making the generation of relations (including how relations can be named) through patterns very flexible, and (III) Implementing this method within an existing ontology building tool (*Text2Onto*). We finally conclude by summarizing the contributions and presenting perspectives in Section 4.

## 2 Difficulties

Willing to semi-automatically build ontologies (or to support ontology building as best as possible) starting form texts, improvements can be concentrated on:

- Improving the input text by modifying (substituting) the employed terms (e.g. for adopting a more standard terminology) and sentence structures, resolving ambiguities and co-references and so on;
- Improving the quality of the final ontology by performing a quality assessment (e.g. using reasoning, if applicable, similarity (and other) measures) followed by relevant modifications;
- Improving the process of building the ontology by improving the efficiency and effectiveness of the required tasks (i.e. relation extraction, concept/instance extraction, etc.).

In this paper, we focus on the third line of improvements, and more specifically, (as said in the Introduction) on relation extraction, because, as explained in section 2.1 below, concept/instance extraction can be performed independently from relation extraction. Section 2.2 presents the inherent difficulties in using pattern-based approaches for extracting semantic relations and related work.

### 2.1 Reasons for processing relation extraction and concept/instance extraction separately

Although concept/instance extraction and relation extraction are two partially dependent tasks, they can be treated separately. A formal justification can be presented as follows, on the top of a hypothetical ontology Description Logics formalization; whenever a relation (role) R is newly introduced, additional axioms involving existing concepts can be added. Generally speaking, introducing R can result in 3 situations:

- Additional specification for an existing concept C e.g. $C \sqsubseteq \exists R.\top \sqcap \forall R.\top$ or $C \sqcap \exists R.\top \sqcap \forall R.\top \neq \bot$;
- Splitting an existing concept in subconcepts C', C'' such as, for instance, $C \sqsubseteq C' \sqcap C''$, $C' \sqsubseteq \exists R.\top \sqcap \forall R.\top$;
- Creating a new concept C' such that $C' \sqsubseteq \exists R.\top \sqcap \forall R.\top$.

These few arguments should convince the reader that the extraction of relationships can be modularly managed as well. As a consequence, addressing only the difficulties concerning relation extraction is not a limitation; it even contributes in a well-defined modular way to improve concept/instance extraction.

### 2.2 Relation extraction methods using flat patterns and the inherent difficulties

Pattern-based relation extraction methods often concern hyponymy and part-of relations [8]. These methods often use patterns expressed as *flat regular expressions* (Flat patterns), which contain basic syntactic information (like part of speech tags, lemmas, affixes, etc.). The most known and successful example of using flat patterns is Hearst patterns [5], which are used for extracting the *hyponymy relation* (or IS-A/subsumption relation when using the standard ontology terminology). Because of their high precision, Hearst patterns have been used even in clustering-based relation extraction methods: for instance, in [2], Hearst patterns have been used to name the clusters of a hierarchy of terms based on the hyponymy relation (a hyponymy hierarchy is close to an IS-A taxonomy). In [10], a similar approach has been used for naming the clusters of a hyponymy hierarchy.

Another successful use of flat patterns is using reliable patterns to correct the extraction results of less accurate patterns [8].

Java Annotation Patterns Engine (JAPE), a language of the open-source platform General Architecture for Text Engineering (GATE[2]), has been the key language for expressing flat patterns. With JAPE, flat patterns are expressed as transducers (using macros, input and output annotations) to annotate sentences in the text that match the pattern. Transducers are organized in queues corresponding to sentences in which, the results (output annotations or macros) of one phase can be used as inputs by the next one. A relevant usage of JAPE can be found in *Text2Onto* [3] (an ontology building tool), where GATE is used as the key library for preprocessing. *Text2Onto* preprocessing tasks involve some of GATE's components such as the Part Of Speech (POS) tagger, the named entity extractor, and also patterns made by the user.

Using flat patterns has been successful for extracting semantic relations, but such patterns suffer from two major limitations that we point out hereafter.

The absence of deep syntactic information in flat patterns leads to misinterpretations when these patterns are matched to the text. Consider the following sentences: (s1) "*The semantic formalization of knowledge has been achieved by the use of several tools such as ontologies, semantic networks and expert systems*."; (s2) "*Euclid, a great mathematician in his own right, showed to a king that there is no royal road to geometry*.". In these sentences, the comma can play two roles, i.e. a conjunction in (s1), or an introducer of apposition in (s2) (in (s2) the apposition is "great mathematician"). Another example of cases leading to misinterpretations is when the syntactic

[2] A full documentation on GATE can be found at http://gate.ac.uk/documentation.html

structure of the text (having impact on its semantic interpretation) cannot be efficiently and effectively captured by flat patterns. This includes cases like verb phrases expressed in active or passive form, or discontinuity cases (topicalization, etc.).

Flat patterns contain often unnecessary symbols for relation extraction, which often reduce the patterns coverage. It is the syntactic information conveyed by symbols that should be identified: in the example above (sentences (s1) and (s2)), what is interesting is to know whether the comma symbol represents a conjunction or an apposition. Another example is in the Hearst pattern *P*: " <Hypernym>(NP) **including** <Hyponym>(NP) ". The flat pattern *P* can be applied successfully to extract the hyponymy relation instance "specie is-hyponym-of organism" (r1) from the sentence (s3) "*Organisms including species like flies, yeast, monkeys and worms have previously been put on diets and shown to have their life spans extended by 30 to 200%*.". However, if we insert the adjective *diverse* between *including* and *species* in the sentence (s3) (which results in the sentence (s4) *"Organisms including **diverse** species like flies, yeast, monkeys and ..."*), then *P* does not match anymore. However, the semantic relation (r1) should have been extracted from both sentences. Adding an adjective between the word *including* and the hyponym in *P* is not necessary (from the semantic view) to identify the hyponymy relation.

### 2.3 Dependency tree patterns

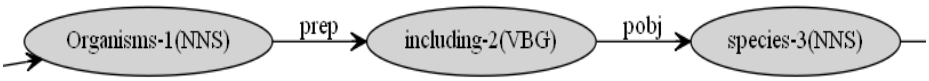


**Fig. 1.** (t3) A sub-tree of the syntactic dependency tree of the sentence (s3)

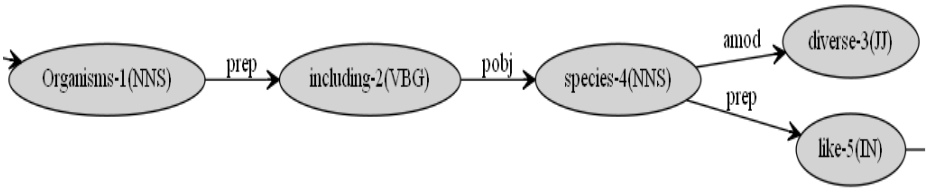


**Fig. 2.** (t4) A sub-tree of the syntactic dependency tree of the sentence (s4)

The limitations of flat patterns mentioned in Section 2.2, can be overcome by using patterns that take into account deep linguistic information, i.e. **syntactic dependency links**[3]. We call these patterns Dependency Tree patterns (DT patterns). For example, the limitation involving the Hearst pattern *P* and sentences (s3) and (s4) mentioned in Section 2.2 can be overcome by the following DT pattern *P2* " <Hypernym>(NP) --prep--> **including(VBG**[4]**)** --pobj--> <Hyponym>(NP) ". A matching between *P2* and

[3] The dependency links (*mwe*, *prep*, *pobj*, *amod*, etc.) mentioned in this paper are described in [13].

[4] VBG is a part of speech tag corresponding to a gerund or the present participle of a verb.

the dependency tree t3/t4 (of the sentence s3/s4) in Figures 1,2 above allows the extraction of the same relation (r1).

Using dependency tree patterns for relation extraction has been proposed in [12] in which the authors presented an algorithm for discovering patterns expressed as dependency paths. Those patterns allowed the authors of [12] to construct (what they called) a "hypernym-only classifier" showing a dramatic improvement compared to previous classifiers: their best logistic regression classifier showed a 132% improvement of average maximum F-score over the Hearst patterns based classifier. In [11] the authors followed a similar approach which they used to compare dependency tree patterns to flat patterns in terms of precision and recall (the patterns they used are for extracting hyponymy relations from Dutch texts) but their result is in contradiction with the work of [12]; the authors of [11] concluded that using deep syntactic information does not produce substantial improvement in the precision and recall of the extracted results. An explanation for this gap can be identified in the section 4.3 (the *error analysis* section) of [11] where one can see that most of the errors are due to the syntactic analyzer.

However, in the works mentioned above, the usage of dependency tree patterns has not been made neither systematic nor user-oriented. Indeed, in those works, there has been no specification of a **formal language** (in the same way that JAPE allows to express flat patterns to be used modularly by extraction tools) for expressing DT patterns that can be used by users for programming and experimenting DT patterns.

The most similar work to ours is [16] which presents a new ontology learning system (*OntoCmaps*) intended to overcome the drawbacks of tools using flat patterns which contain only shallow linguistic information (such as *Text2Onto*). The patterns used in *OntoCmaps* [17] contain deep linguistic information and are expressed in a language syntactically different than ours. Both languages are meant to make patterns use deep linguistic information for extracting knowledge from the text through the usage of regular expressions. Some of the differences between the language used to express patterns in *OntoCmaps* and DTPL (defined in Section 3) is that the later allows to use many POS tags for a node, it also allows to express properties for patterns (as the JAPE language does) that extraction tools could use modularly. Another difference between the two languages is that each pattern expressed in DTPL is meant to extract only **one kind** of relations, the reason is that each time that a pattern identifies more than one kind of relations it indicates nested patterns to differentiate by specifying dependency bindings (each dependency binding consists of a dependency link, the governor and the dependent) that should not exist when a match occurs (by adding the symbol *'!'* to the dependency binding to exclude from the matching, in [18] we present an example for such use); this distinction is needed because the extracted results of one pattern could be erroneous for another one. Another difference is that *OntoCmaps* uses collapsed dependencies [13] while we use uncollapsed ones.

## 3 Improvements in ontology building by using DT patterns

The key contribution of this paper consists in giving OBSS the ability to modularly use **patterns expressed as dependency trees** (Dependency Tree patterns—DT patterns) to take into account deep syntactic information found in texts. This will be achieved by (I) Specifying a formal language for expressing DT patterns to be matched with syntactic dependency trees of sentences, and (II) Creating and integrat-

ing a new algorithm in an ontology building tool (*Text2Onto*) to extract semantic relations by using DT patterns.

### 3.1 DTPL, a language for expressing DT patterns

In this section we are going to define DTPL (Dependency Tree Patterns Language), a language for expressing patterns represented as dependency trees, each DT pattern helps to extract a semantic relation. In order to extract a given relation from a sentence *S*, a DT pattern must be matched with the syntactic dependency tree of *S*.

Dependency trees (both patterns and sentences) comprise nodes and arrows. Table 1 provides the reader with the relevant definitions (the 3rd column concerns patterns— DT patterns— only).

| Tree component | Components of the tree component | Optional or Mandatory |
|---|---|---|
| Node | *NodeValue*: it represents either a non-terminal symbol (output annotation) or a terminal symbol (word) | mandatory |
| | *PosTags*: the part(s) of speech of the symbol represented by this node | mandatory (the wildcard character * can be used, it matches with any POS tag) |
| | *Index*: the index of the symbol in the sentence in which the pattern is to be matched | optional |
| Arrow | *DependencyLink*: the name of the dependency link that exists between the two connected nodes | mandatory |
| | *SourceNode*: the node from which the arrow is departing | mandatory |
| | *ArrivalNode*: the node to which the arrow is aiming at | mandatory |

**Table 1.** Inner components of a dependency tree

In DT patterns, each node must possess only one parent, with one exception for any node linked by the *ref* dependency link (i.e. in a sentence containing a co-reference, the *ref* link binds a relative pronoun with the noun it refers to) with its governor, the reason is that such nodes have more than one parent (for more detail on co-reference links used in this paper we refer the reader to [13]). In other terms, without the occurrences of the *ref* link, a DT pattern must have a tree structure. In [18] the reader can find examples that illustrate how the use of co-reference links in DT patterns allows to extract semantic relations.

In DT patterns, each **node** has to be expressed in the form **NodeValue-Index(PosTags)** (e.g. the nodes " as(IN) ", " as-5(IN) ", " as(*) "), see the example at the end of this subsection. Each **arrow** must be expressed in the form " **DependencyLabel(SourceNode,ArrivalNode);** ". The only imposed constraint is that there must be no spaces between the closed parenthesis " ) " and the " ; " character for expressing each arrow (for instance, in the arrow " mwe(as(IN),such(JJ)); " of the DT pattern (dtp1) at Figure 3, we have *DependencyLink*=mwe, *SourceNode*=as(IN) and *ArrivalNode*=such(JJ)). The output annotation labels (which correspond to non-

terminal symbols) are in the form **<annotationLabel>**. For instance, *<firstHyponym>*, *<domain>*, *<range>* and *<relationName>* are nonterminal symbols in the pattern (dtp1).

DT patterns can possess properties. Each property corresponds to a non-terminal symbol. Each pattern property is defined between two *'#'* characters in the form *#propertyName=regularExpression#*, where *propertyName* is the name of the property, and *regularExpression* is a regular expression combining terminal and non-terminal symbols except pattern properties (for instance, in the pattern (dtp1) it's not allowed to define the *<relationName>* property as follows *#<relationName>=<domain>_to_<range>#* because the non-terminal symbols *<domain>* and *<range>* are also properties of the pattern).

A DT pattern allows to extract a relation (unary, binary, or having any other non-null arity). The idea is that each argument of a relation can be pointed out by a pattern property. For instance, to identify the hyponym and hypernym of a hyponymy relation, one can use the annotations **<hyponym>** and **<hypernym>**. For binary relations (which are quite important because –for instance– any Description Logics formalization of an ontology comprises only binary relations), we can use the properties *<domain>* to represent the Domain of a relation and *<range>* to represent its Range.

The expressions written in a DT pattern are either defining properties (e.g. the 1st three lines in the patterns of Figure 3) or defining dependency links between nodes. The order on which POS tags are mentioned for each node isn't important (for instance, in (dtp2), the nodes *<verb>(VBN|VBZ|VBD)* and *<verb>(VBZ|VBD|VBN)* are the same).

For extracting binary relations for ontologies, DT patterns have to contain the three properties *<relationName>*, *<domain>*, and *<range>*.

| | |
|---|---|
| #**<relationName>**=is-hyponym-of#<br>#**<domain>**=<conjDep>#<br>#**<range>**=<prepositionGov>#<br>mwe(as(IN),such(JJ));<br>pobj(as(IN),<firstHyponym>(NN\|NNS\|NNP));<br>prep(<prepositionGov>(NN\|NNS),as(IN));<br>conj(<firstHyponym>(NN\|NNS\|NNP),<conjDep>(NN\|NNS\|NNP)); | #**<relationName>**=<verb>_<directObject>_<preposition>#<br>#**<domain>**=<depNoun> <subject>#<br>#**<range>**=<prepositionalObject>#<br>nsubj(<verb>(VBN\|VBZ\|VBD),<subject>(NNP\|NN\|NNS));<br>pobj(<preposition>(IN\|TO),<prepositionalObject>(NNP\|NN\|NNS));<br>dobj(<verb>(VBZ\|VBN\|VBD),<directObject>(NNP\|NN\|NNS));<br>prep(<verb>(VBZ\|VBD\|VBN),<preposition>(IN\|TO));<br>nn(<subject>(NNP\|NN\|NNS),<depNoun>(NNP\|NN\|NNS)); |
| (dtp1) DT pattern similar to Hearst's *such as* pattern | (dtp2) DT pattern for extracting semantic relations based on intransitive verb phrases (verb phrases of which the verb is intransitive) containing prepositions |

**Fig. 3.** The DT patterns (dtp1) and (dtp2)

In Figure 3, in the DT pattern (dtp1), the output annotation *<domain>* represents the hyponym, while *<range>* represents the hypernym. In (dtp2), the output annotation *<domain>* represents the subject of the verb annotated by *<verb>*, while *<range>* represents the prepositional object. The tree (tdtp1) in Figure 4 is a way to visualize the DT pattern (dtp1). For visualizing (dtp2) we refer the reader to [18].

The Domain and Range also have to be written as regular expression. For details on the syntax of DTPL we refer the reader to [18].

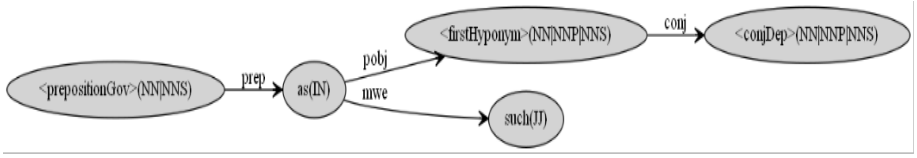


**Fig. 4.** (tdtp1) A visual representation of the DT pattern (dtp1)

Matching (dtp1) with the dependency tree (t5) in Figure 5 (the syntactic dependency tree of the sentence (s5) *"Carmakers such as Maruti, Hyundai, Tata, Toyota, Ford, GM & Mercedes put brakes on price hikes despite margin pressures"*) allows to extract the relations ***is-hyponym-of(mercedes,carmaker)***, ***is-hyponym-of(ford,carmaker)***, ***is-hyponym-of(toyota,carmaker)***, ***is-hyponym-of(tata,carmaker)***, ***is-hyponym-of(hyundai,carmaker)***, ***is-hyponym-of(gm,carmaker)***. While the pattern (dtp2) allows to extract from the tree (t6) in Figure 6 (the syntactic dependency tree of the sentence (s6)*"The Ebola virus causes internal bleeding to its victims"*) the relation ***cause_bleeding_to(ebola virus,victim)*** (r2).

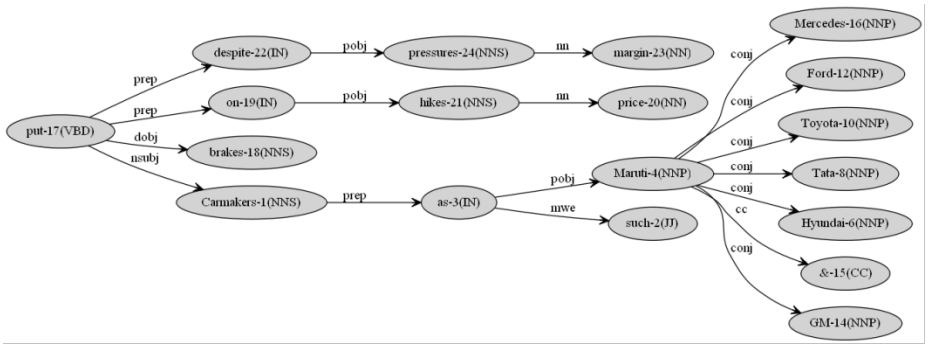


**Fig. 5.** (t5) The syntactic dependency tree of the sentence (s5)

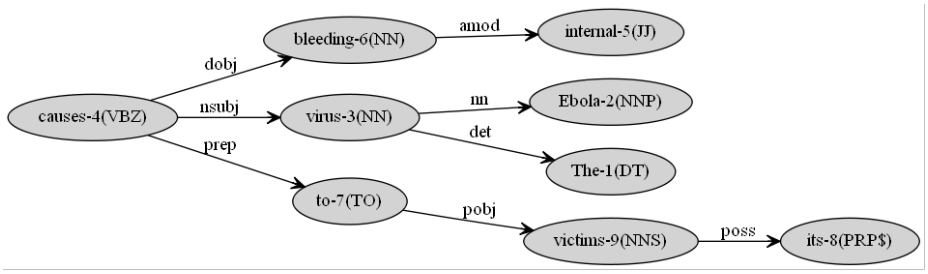


**Fig. 6.** (t6) The syntactic dependency tree of the sentence (s6)

Other examples (including the exploitation of co-reference resolutions, and also examples of expressing the exclusion of some dependency bindings in DT patterns — to improve the accuracy of patterns—) can be found at [18].

### 3.2 Text2Onto enhancement and improvement by introducing DTPL

For relation extraction purposes, *Text2Onto* has been used as target for testing DT patterns expressed by the language DTPL.

*Text2Onto* comprises various algorithms for ontology extraction tasks (such as relation extraction, concept/instance extraction). Given our interest on relation extraction, we will only present *Text2Onto*[5] native algorithm for relation extraction, named *SubcatRelationExtraction*.

| Difference | DTP_BinaryRelationExtraction | SubcatRelationExtraction |
|---|---|---|
| The usage of deep linguistic information | Uses deep linguistic information for extracting semantic relations. | Uses shallow syntactic information for relation extraction. |
| The language used for expressing patterns | Uses patterns expressed in DTPL. | Uses patterns expressed in JAPE. |
| Extracting multiple instances of a relation | Extracts several instances of a relation without restrictions and provides a frequency, indicative of the relevance of a relation instance. | Extracts only one instance of a relation formed by the most frequent element of the Domain and the most frequent element of the Range (which is a source of errors). |
| Allowing more modularity for relation extraction | Allows the explicit naming of the extracted relation (is-a relations, part-of relations, verb phrase based relations, etc.). | • Constrained to extract only relations based on verb phrases (for instance, for transitive verb phrases, it generates relation instances in the form ***Verb(Subject,Object)***).<br>• It does not allow the explicit naming of the relation. |

**Table 2.** Differences between *DTP_BinaryRelationExtraction* and *SubcatRelationExtraction*

*SubcatRelationExtraction* is a pattern-based relation extraction algorithm using flat patterns expressed in JAPE (here, patterns are called JAPE rules) located in *Text2Onto*'s */3rdparty/gate/english* directory. *SubcatRelationExtraction* takes into account the information that JAPE rules possess (like the output annotations *TransitiveVerbPhrase*, *Subject* and *Object*) to generate relations. However, another limitation of *SubcatRelationExtraction* (other than using flat patterns) is the exclusive usage of verb phrases (transitive/intransitive, etc.) for extracting and naming relations. For instance, from the sentence (s6), *SubcatRelationExtraction* extracts (given the **right** JAPE rule) ***cause_to(ebola virus,victim)*** (which is not as meaningful as (r2)). Indeed, despite the fact that numerous semantic relationships can be identified from verb phrases, this is not always the case (as for the hyponymy relation in sentence (s5)).

We implemented the new relation extraction algorithm *DTP_BinaryRelationExtraction*, which uses DT patterns expressed in DTPL for binary relation extraction. *DTP_BinaryRelationExtraction* uses a JAVA library (*TreeMatcher*) which generates relations by using the regular expressions that define the pattern properties <*domain*>, <*range*> and <*relationName*>.

To parse the input texts, *TreeMatcher* uses the *Stanford Full Parser* version 3.3.1 (which can be found at [14]) and its parsing model *englishPCFG.ser.gz*. *TreeMatcher*

[5] *Text2Onto* version 2007-11-09, it can be found at https://code.google.com/p/text2onto/.

is tolerant to DT patterns containing empty lines and multiples space characters as well. Each pattern has to be specified in DTPL in a distinct file (textual file) within *Text2Onto*'s */3rdparty/gate/english* directory. The name of each file containing a DT pattern has to start with "dtp-". For example, the DT pattern (dtp1) can be named "dtp-SuchAsPattern".

DT patterns can be added, removed, modified and used modularly by *Text2Onto* like any flat pattern expressed in JAPE (e.g. the JAPE rules *SubclassOfRelation1*, *SubclassOfRelation2*, etc. of the JAPE file *ontological_relations.jape* in *Text2Onto*'s */3rdparty/gate/english* directory).

*TreeMatcher* allows *DTP_BinaryRelationExtraction* to process relation extraction in four steps: (I) Reading the DT patterns in *Text2Onto*'s */3rdparty/gate/english* directory, (II) Producing syntactic dependency trees from the input corpus by using the *Stanford Full Parser*, (III) Performing matching between dependency trees extracted from the corpus and the DT patterns, each matching can produce many relations (the generation of relations uses the regular expressions attached to the properties *<domain>*, *<range>* and *<relationName>* (see Section 3.1)), and each relation contains the frequency of its occurrence on the corpus, (IV) Producing the result as a list of relations (each relation instance possess an index indicative of its relevance).

Table 2 above summarizes the differences between the algorithm *DTP_BinaryRelationExtraction* and *Text2Onto*'s native algorithm *SubcatRelationExtraction*.

## 4 Conclusion and perspectives

We have presented in this paper a method giving ontology building tools the ability to use deep linguistic information in patterns called DT patterns. Specifically, we have first defined a new language (DTPL) to express these patterns, and, accordingly, enhanced and improved an existing ontology building tool (*Text2Onto*).

The work that we described in this paper is a piece of a bigger scheme aiming at:

- The integration of a new parsing strategy (especially made for relation extraction algorithms) assuring the accuracy of the extracted relations (because of the deep syntactic analysis) while maintaining a reasonable computational cost;
- Introducing two weakly supervised algorithms for pattern discovery, one for DT patterns and the other for flat patterns. The patterns to be learned are for extracting sematic relations (including IS-A and part-of).

Using deep linguistic information needs deep syntactic analysis of the text which takes longer runtime than shallow parsing. This may be overcome by using a strategy for relation extraction which consists in parsing only sentences that contain at least two Terms Representative of the knowledge Domain of the corpus (TRD), the methods for extracting such terms need only shallow parsing. This strategy is expected to enhance the precision of the extracted results, but the gain in computational time depends on how many sentences contain at least two TRDs in the corpus (i.e. if such sentences are too frequent, then there would be no significant gain).

## REFERENCES


1. Sergey Brin. 1998. *"Extracting patterns and relations from the world wide web"*. WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT '98.
2. Sharon A. Caraballo. 1999. *"Automatic acquisition of a hypernym-labeled noun hierarchy from text"*. In Proceedings of ACL-99. pp 120-126, Baltimore, MD.
3. Philipp Cimiano, Johanna Völker. 2005. *"Text2Onto: a framework for ontology learning and data-driven change discovery"*. In Proceedings of the 10th international conference on Natural Language Processing and Information Systems, June 15-17, 2005, Alicante, Spain [doi>10.1007/11428817_21].
4. Zellig S. Harris. 1954. *"Distributional structure"*. Word 10 (23): 146–162.
5. Marti A. Hearst. 1992. *"Automatic Acquisition of Hyponyms from Large Text Corpora"*. In Proceedings of ACL-92. Nantes, France.
6. Maayan Geffet, Ido Dagan. 2005. *"The distributional inclusion hypotheses and lexical entailment"*. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, p.107-114, June 25-30, 2005, Ann Arbor, Michigan [doi>10.3115/1219840.1219854].
7. Maayan Zhitomirsky-Geffet, Ido Dagan, Idan Szpektor, Lili Kotlerman. 2010. *"Directional distributional similarity for lexical inference"*. Natural Language Engineering, 16(04): 359–389.
8. Patrick Pantel, Marco Pennacchiotti. 2006. *"Espresso: leveraging generic patterns for automatically harvesting semantic relations"*. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, p.113-120, July 17-18, 2006, Sydney, Australia.
9. Patrick Pantel, Deepak Ravichandran. 2004. *"Automatically labeling semantic classes"*. In Proceedings of HLT/NAACL-04. pp. 321-328. Boston, MA.
10. Patrick Pantel, Deepak Ravichandran, Eduard H. Hovy. 2004. *"Towards terascale knowledge acquisition"*. In Proceedings of COLING-04. pp. 771-777. Geneva, Switzerland.
11. Erik T. K. Sang, Katja Hofmann. 2009. *"Lexical patterns or dependency patterns: which is better for hypernym extraction?"*. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL '09). Association for Computational Linguistics, Stroudsburg, PA, USA, 174-182.
12. Rion Snow, Daniel Jurafsky, Andrew Y. Ng. 2004. *"Learning syntactic patterns for automatic hypernym discovery"*. In NIPS 2004.
13. Marie-Catherine de Marneffe, Christopher D. Manning. 2008. *"Stanford typed dependencies manual"*: http://nlp.stanford.edu/software/dependencies_manual.pdf
14. *"The Stanford Parser: A statistical parser"*: http://nlp.stanford.edu/software/lex-parser.shtml
15. Julie Weeds, David Weir, Diana McCarthy. 2004. *"Characterizing Measures of Lexical Distributional Similarity"*. In Proceedings of Coling-04. Geneva, Switzerland.
16. Amal Zouaq, Dragan Gasevic, Marek Hatala. 2011. *"Towards open ontology learning and filtering"*. Information Systems, v.36 n.7, p.1064-1081, November, 2011 [doi>10.1016/j.is.2011.03.005].
17. Amal Zouaq, Dragan Gasevic, Marek Hatala. 2012. *"Linguistic Patterns for Information Extraction in OntoCmaps"*. In Proceedings Of the 3rd Workshop on Ontology Patterns - WOP2012, in conjunction with the 11th International Semantic Web Conference, Boston, USA.
18. *"Grammar of the DTPL language, and examples"*: http://people.irisa.fr/Nicolas.Bechet/WOP2014/

# Towards the reuse of standardized thesauri into ontologies[1]

Elena Cardillo[1], Antonietta Folino[2], Roberto Trunfio[2], Roberto Guarasci[2]

[1] Institute for Informatics and Telematics, Italian National Research Council, Pisa, Italy
[2] Laboratorio di Documentazione (LabDoc), University of Calabria, Rende, Italy
elena.cardillo@iit.cnr.it
{antonietta.folino,roberto.trunfio,roberto.guarasci}@unical.it

**Abstract.** One of the main holdbacks towards a wide use of ontologies is the high building cost. In order to reduce this effort, reuse of existing Knowledge Organization Systems (KOSs), and in particular thesauri, is a valuable and much cheaper alternative to build ontologies from scratch. In the literature tools to support such reuse and conversion of thesauri as well as re-engineering patterns already exist. However, few of these tools rely on a sort of semi-automatic reasoning on the structure of the thesaurus being converted. Furthermore, patterns proposed in the literature are not updated considering the new ISO 25964 standard on thesauri. This paper introduces a new application framework aimed to convert thesauri into OWL ontologies, differing from the existing approaches for taking into consideration ISO 25964 compliant thesauri and for applying completely automatic conversion rules.

**Keywords:** Thesauri; Ontology; Knowledge Organization System; Conversion Framework; ISO 25964; OWL.

## 1 Introduction

Nowadays, Knowledge Organization Systems (KOSs) (e.g. thesauri, taxonomies, ontologies, classification systems) are assuming a key role in knowledge and information management. Even though a KOS can be used as a standalone application, to take benefit of its features it is generally integrated in larger information systems (search engines, content management systems, etc.). Thus, in the literature a consolidated issue is the interoperability of different KOSs for a cost-effective exchange of structured information. Driven by the philosophy of the Semantic Web [4] that widely relies on ontologies for knowledge sharing, reuse and inference, researchers are concentrating their effort towards the reuse of less formal terminological systems, and in particular thesauri, into ontologies. This is motivated by the fact that a huge amount of thesauri have been already realized in the last decades for almost every knowledge

[1] Authors have jointly collaborated in the drafting of the article. However, each has been involved in one or more paragraphs: E. Cardillo of Section 1, 2.1, and 3; A. Folino of Section 2.2., and 4.2; R. Trunfio of all Section 4; and R. Guarasci of Section 5.

domain [18] and some of them are steadily updated. So, there could be no benefit to replicate a domain modeling from scratch, but recovering and reusing the knowledge embedded into a thesaurus could speed-up the activity of building ontologies.

In order to address this challenging topic, this paper introduces an application framework for reusing standardized thesauri[2] into ontologies as defined by [7]. In particular, the framework uses new re-engineering rules formulated to be applied on an ISO 25964 compliant thesaurus [12, 13].

The paper is organized as follows: a brief description of the main similarities and differences between thesauri and ontologies, and between thesaurus entities before and after the introduction of ISO 25964 is given in Section 2; an updated overview of the literature is provided in Section 3; the description of the proposed re-engineering approach along with some examples of rules for the reuse of a standardized thesaurus into an OWL ontology are given in Section 4; finally, conclusions and future works are outlined in Section 5.

## 2 Thesauri and Ontologies: an overview

In this section, first a brief description of the similarities and differences between thesauri and ontologies is given; secondly, some important novelties introduced by the ISO 25964 standard in terms of thesaurus principles, interoperability and mapping with other types of controlled vocabularies are briefly reported.

### 2.1 Thesauri vs Ontologies

Some similarities between thesauri and ontologies can be identified. In particular, both describe a domain, include concepts and relations between them, use hierarchies.

Furthermore, both are used in applications of information management for cataloguing and in search engines. However, several differences must be taken into account, specifically depending on their origin and purposes. Thesauri have been used for a long time in librarian contexts as indexing tools and controlled vocabularies. As such they are thought to represent knowledge in a less formal and comprehensive manner than ontologies. They are not characterized by a level of conceptual abstraction as ontologies, which are originated from philosophy and conceived as accepted and formal ways of describing knowledge domains[3]. So difference between a concept and its lexicalization in a thesaurus is not clearly established.

Moreover, other fundamental differences are related to their structures. As clearly described in [10], with respect to thesauri, ontologies are characterized by: *(i)* an explicit representation of the types of relationships; *(ii)* the use of powerful formalisms

[2] Thesauri are defined as "controlled and structured vocabulary in which concepts are represented by terms, organized so that relationships between concepts are made explicit, and preferred terms are accompanied by lead-in entries for synonyms or quasi-synonyms" [12].

[3] Ontologies consist of a set of definitions of classes, properties, and individuals, together with a set of axioms expressing the relations between classes and properties, and a set of facts about particular individuals.

(e.g. axioms, relationships cardinality). These differences enable ontologies to reuse knowledge of a domain, make domain assumptions explicit, and allow access to and evolution of legacy data, as well as automated reasoning.

### 2.2 ISO 25964 vs other standards on thesauri

In this section we briefly analyze structural and conceptual differences between ISO norms on thesauri in order to allow for the proposal of updated and more standardized re-engineering rules for building ontologies from thesauri.

From a more comprehensive perspective, the need for a new norm about thesauri and other controlled vocabularies was due to the technological evolution and to the new role recognized to thesauri as information retrieval tools and not only as indexing resources. This required new specifications about data models, exchanging formats, interoperability with other vocabularies, etc., all present in the ISO 25964 [12, 13]. ISO 25964 completely replaces the previous ISO 2788:1986 [15] and ISO 5964:1985 [14] standards, and almost reuses the content of two other national norms: ANSI/NISO Z39.19:2005 [2] and BS 2783-5:2008 [5].

Given the aim of the paper, we focus here only on three main changes that contribute to approach thesauri towards ontologies and that are of interest for re-engineering purposes (see [6] for a complete analysis):

- ***Term-based vs Concept-based thesaurus.*** Considering thesaurus structure, ISO 25964 provides a data model (absent in ISO 2788:1986) that formally represents its objects and relationships. The major difference that can be observed in this data model is the clear distinction between concepts and terms[4] (*ThesaurusConcept* is separated from *ThesaurusTerm* and from *PreferredTerm*, *hasPreferredLabel*, *hasNonPreferredLabel*, etc.). In fact, similarly to ontologies, concepts in ISO 25964 are defined as units of thoughts lexically represented by terms. Thus, a thesaurus is considered as a concept-based resource, rather than a term-based one.

- ***Thesaurus structure.*** The ISO 25964 adds important elements for organizing the structure of a thesaurus, such as the *thesaurus array*[5] and *concept group*[6] elements, that were not explicitly illustrated in the previous ISO norm. This implies that in a conversion process these constructs should be taken into account, so new patterns have to be introduced in order to reengineer them into ontology elements.

- ***Semantic relationships.*** One of the innovations introduced by the current norm is the possibility to make explicit the nature of semantic relationships, by further specifying the standard ones. This possibility contributes to make thesauri more similar to ontologies, as well as the clear distinction between *concept* and *term*. In

[4] This does not mean that in the ISO 2788:1986 the same meaning was recognized to terms and concepts, but for practical aims they were treated as synonyms.

[5] A *thesaurus array* is defined in [13, p.4] as "group of sibling concepts".

[6] A *concept group* is defined in [12, p.1] as "a group of concepts selected by some specified criteria, such as relevance to a particular subject area".

particular, some changes regarding the *equivalence* and the *hierarchical* relationships are worth mentioning:

- *Equivalence relationship*. The ISO 25964 clarifies, first of all, that it relates terms rather than concepts (more than one term for expressing a unique concept). In addition to the standard tags USE/UF, the norm establishes that other tags could be used, depending on the kind of terms to be put into relation: AB for abbreviations, FT for the full form of a term, SP for spelling variant.
- *Hierarchical relationship*. It holds between two or more concepts that express subordinate and super-ordinate meanings at different levels and is depicted through the tags BT/NT (i.e. narrower term and broader term, respectively). To extend the richness of thesauri, hierarchical relationships can be further divided into generic (NTG/BTG), partitive (NTP/BTP) and instantial (NTI/BTI). Both ISO 2788 and 25964 clarify that concepts can be hierarchically related only if they belong to the same conceptual category (objects, actions, properties, etc.) and if the so called *all-and-some test* is verified. Moreover, the ISO 25964 specifies that this relationship holds "*between a pair of concepts when the scope of one of them falls completely within the scope of the other*" [12]. This criterion is undoubtedly important to guarantee the correctness of the hierarchy.

## 3 Related Works

In the last decades the conversion of thesauri into ontologies has taken the attention of the scientific community, that produced several stimulating works. Nonetheless, few of them rely on a computer-aided conversion process. Moreover, most of the existing approaches were conceived before the publication of ISO 25964.

Two pioneering works described in [22] and [19] show the creation of RDFS Ontologies reusing domain specific thesauri: the former in particular proposes highly structured semantic descriptions for a subset of art-object (antiquate furniture) from the Art and Architecture Thesaurus (AAT)[7]; the latter approach is tested on the well-known AGROVOC thesaurus[8]. In both cases BT/NT relationships are mapped to the is-a relation in the ontology and terms of the thesaurus to classes.

A semiautomatic approach is proposed in [3] where the TERMINAE method for ontology extraction from texts is applied to a thesaurus. Here three hypotheses about the reuse of terms relations are proposed: *(i)* preferred terms in the thesaurus are indexed to constitute terms identifying domain concepts in the ontology; *(ii)* relations between terms and preferred terms are synonym relations in the thesaurus, that means putting together terms as label of the same concept in the ontology; *(iii)* relations between preferred terms are indexed to identify concepts relations. Even if some interesting conversion rules are presented, the application of this methodology to particular use cases is very limited (i.e. an application for the medical domain).

[7] http://www.getty.edu/research/tools/vocabularies/aat/
[8] http://aims.fao.org/standards/agrovoc/about/

More close to our work is the conversion approach proposed by [9], where they formalize an ISO 25964 compliant thesaurus into an OWL ontology[9], assuming that a standard-compliant thesaurus is *concept-based*, rather than *term-based*, as observed in the norm. This implies that thesaurus concepts and facets are treated as classes in the corresponding ontology, while thesaurus terms (synonyms or quasi-synonyms introduced by the equivalence relationships) are represented in the ontology as class labels. In addition, checking and refinement of thesaurus relationships are performed in order to: explicitly distinguish between different types of hierarchical relationships, manage cycles, orphans, polyhierarchy, etc. (e.g. the *generic relation* (NTG/BTG) are translated into *is-a* relation in the ontology, and the *instantial* one (NTI/BTI) into the distinction between class and individuals). Moreover, in order to guarantee similarity in design choices, an alignment of the thesaurus (i.e. AGROVOC) with top-level ontologies is proposed. However, also in this case, the approach "has not been tested and refined on the scale of re-engineering a complete thesaurus" [9].

In addition to these approaches, in the last few years different patterns for the re-engineering of non-ontological resources (including thesauri) into ontologies have been proposed. We mainly refer here to those proposed in [20] and [21], whose objective is the conversion of both terms/concepts and semantic relationships provided by thesauri (USE/UF; BT/NT; RT) into ontology classes, individuals and relations.

Other approaches that try to convert thesauri, but also classification systems into more formal resources as ontologies can be found in [8, 16, 17].

Considering the literature, the aim of our research is to demonstrate that, although these studies proved usefulness and applicability, some of the existing patterns need to be revised/integrated in the light of the mentioned ISO norm 25964 [12, 13].

Furthermore, due to the common issues related to the conversion process (e.g. the ambiguity in the distinction between a class and an instance, difficulties in the specification of the nature of thesaurus relationships) in our opinion, a challenge for the future is the development of new methodological and application frameworks that allow for more specific, standard based, and automatic conversions.

## 4 A Framework for Reusing ISO 25964 Thesauri into OWL Ontologies

By taking advantages of the analysis provided in Section 2, this section briefly outlines the concepts at the basis of our conversion framework. The framework's aim is to export the knowledge embedded into an ISO 25964 compliant thesaurus and to reuse it as the skeleton for a formalized ontology. Specifically, the application constructs an ontology in OWL 2 DL[10] starting by an RDF-based schema for modelling an ISO 25964 standardized thesaurus. The framework is implemented in Java 7 and uses APIs taken from the well-known *Apache Jena*[11] framework for building Seman-

[9] https://www.w3.org/TR/owl2-overview/
[10] http://www.w3.org/TR/owl-features/
[11] https://jena.apache.org/

tic Web and Linked Data application. In particular, it uses the `Model Loader` and the `Model Translator` APIs. The former is devoted to enable the loading of a thesaurus formalized through a model specified via RDF[12] which is handled by the `RDF` API from Apache Jena.

The `Model Translator` API applies a set of rules designed to identify classes, properties, instances and semantic relations from the RDF graph and consequently to extract an ontological schema. For this purpose, the `Ontology` API from Apache Jena is used to represent a formal logical descriptions in OWL 2 DL. The use of OWL 2 DL is motivated by the fact that it is decidable and its tableau reasoners prove to be tractable in practice on several scales of problems (despite the poor theoretical complexity). Moreover, for reasons of performance, all the conversion rules are hard coded in the Java library. The Java API is available upon request from the authors.

### 4.1 Model Loader API

As mentioned before, the `Model Loader` is devoted to represent a standardized thesaurus in an RDF-graph based on a SKOS (Simple Knowledge Organization System) [13] and SKOS-XL[14] extension for modelling ISO 25964 thesauri, known as iso-thes-25964 [11] (for convenience, we define the prefix `it: <http://purl.org/iso25964/skos-thes>`). It must ensure the consistency of the RDF graph with respect to the domains of the semantic relations and entailment rules provided by the iso-thes-25964 and those inherited by SKOS. To this extent, the iso-thes-25964 extension promotes the reuse of some of the standard SKOS classes (i.e. `skos:Concept`, `skos:ConceptScheme`). However, it introduces important classes like `it:ConceptGroup`, aimed to represent group of concepts selected by some specified criteria, such as relevance to a particular subject area (see Sec. 3.18 from [13]) and `it:ThesaurusArray`, used in our framework to represent a node label with characteristic of division (see Sec. 11 from [12]).

Moreover, it provides new semantic relations (e.g. `it:broaderGeneric`, `it:broaderInstantial`, `it:broaderPartitive`, `it:narrowerGeneric`, `it:narrowerInstantial`, `it:narrowerPartitive`, etc.), which integrate and specialize the existing SKOS relations intended to be used for thesauri representation (e.g.: `skos:broader`, `skos:broaderTransitive`, `skos:hasTopConcept`, `skos:inScheme`, `skos:narrower`, `skos:narrowerTransitive`, `skos:related`, `skos:topConceptOf`, `skos:member`).

Finally, following ISO 25964, we extended iso-thes-25964 by introducing the class `Facet` as sub-class of `skos:Collection`. `Facet` is defined as disjoint with classes `it:Concept`, and `it:ThesaurusArray`. Moreover, it is in domain of `it:superGroup` and in range of `it:inScheme`. In our opinion, the choice to introduce a new class is supported by the need to ensure a non-ambiguous representation of facets in the RDF graph.

---

[12] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222
[13] http://www.w3.org/2004/02/skos/
[14] http://www.w3.org/TR/skos-reference/skos-xl.html

### 4.2 The Model Translator API: Conversion rules

The `Model Translator` applies a set of rules in order to extract an OWL 2 DL ontology from an ISO 25964 standardized thesaurus formalized according with the `Model Loader` specifications. The `Model Translator` can translate in an OWL class or property elements of a thesaurus such as concepts, concept groups, facets, thesaurus arrays, as well as associative (RT), hierarchical (TT, BT/NT, BTG/NTG, BTP/NTP, BTI/NTI) and equivalence relationships (USE/UF). The RDF graph is explored hierarchically starting by the nodes identified as facets, top terms and terms with no broader definition. Finally, concept groups nodes are explored. For space reasons, we give examples only for those rules that need clearer explanations.

**Facets and Top Terms.** Facets are used to group together concepts from the same category, that means concepts sharing similar properties, such that members of a facet are mutually exclusive with the members of another. According to the standard, the following set of facets should be used: *things*, *types*, *parts*, *properties*, *materials*, *processes*, *activities*, *products*, *by products*, *patients*, *agents*, *space and time*. In this perspective, although facets can be personalized, in our framework facets are considered as the classes of a foundation ontology. Hence, rule *(1)* is applied:

*(1)* "*Once a* `Facet` *node is found, it must be mapped as a class that is subclass of* `owl:Thing` *and has no other parent relations*".

When a `skos:Concept`, identified as a *Top term*, is found in the RDF graph, then it must be mapped to an ontological class, as stated by the following rule:

*(2)* "*Given a* `skos:Concept <A>` *and a* `skos:ConceptScheme <C>` *such that* `<A> skos:topConceptOf <C>`, *then: if exists one and only one* `Facet <F>` *such that* `<F> skos:member <A>`, *then* `<A>` *is mapped as subclass of the ontological class defined for* `<F>`*; otherwise* `<A>` *is mapped as a subclass of* `owl:Thing` *and it has no other parent relations". Moreover, if another* `Facet <E>` *exists, then a relation* `owl:disjointWith` *must be used between the resulting OWL classes for nodes* `<A>` *and each class under facet* `<E>`".

For translation convenience, whenever no TT relation are explicitly provided in the thesaurus, we assume that a concept that has not broader concepts is a top concept.

**Hierarchical relationships**. Generally speaking, whenever a BT/NT or a BTG/NTG relationship is found in the RDF graph, the following rule comes:

*(3)* "*Given two* `Concept` *nodes* `<A>` *and* `<B>` *such that* `<A> skos:narrower <B>`, *then* `<A>` *and* `<B>` *are mapped as two OWL classes such that* `<B> rdfs:subClassOf <A>`".

Further rules are needed for the hierarchical relationships BTP/NTP and BTI/NTI. Specifically, a BTP/NTP relation in our framework entails rule *(4)*:

> *(4)* "*Given three* `Concept` *nodes* `<A>`, `<B1>` *and* `<B2>`, *such that* `<A>` `it:broaderPartitive` `<B1>`, `<B2>`, *then* `<B1>` *and* `<B2>` *are each represented in OWL either as classes linked to node* `<A>` *by a part of relationship in the case they have further narrower concepts in the thesaurus, or as individuals of node* `<A>`".

A narrower concept represented via the BTI/NTI relation generally becomes instance of the OWL class obtained from the broader concept. However, as stated in [23], the following exception may occur: if at a certain level of the hierarchy in the thesaurus an instance of a NTI/BTI relation is also the broader concept of one or more concepts in other BT, BTP, or BTG relations, then it cannot be converted in the ontology as an individual. To clarify this exception, the following example is provided:

```
Countries
  NTI United States of America

United States of America
  NTP Alabama
  NTP Alaska
  NTP Arizona
```

This is mapped in RDF via the iso-thes-25964 model as follows using *turtle notation*:

```
@prefix ex: <http://example.com/skos/thes#> .
...
ex:id1 rdf:type skos:Concept;
skos:prefLabel "Countries"@en;
  it:narrowerInstantial ex:id2.
ex:id2 rdf:type skos:Concept;
  skos:prefLabel "United States of America"@en;
  it:broaderInstantial ex:id1;
  it:narrowerPartitive ex:id3, ex:id4, ex:id5.
ex:id3 rdf:type skos:Concept;
  skos:prefLabel "Alabama"@en;
  it:broaderPartitive ex:id2.
ex:id4 rdf:type skos:Concept;
  skos:prefLabel "Alaska"@en;
  it:broaderPartitive ex:id2.
ex:id5 rdf:type skos:Concept;
  skos:prefLabel "Arizona"@en;
it:broaderPartitive ex:id2.
```

Since ontologies necessarily distinct between classes and individuals, in order to enable reasoning and inferencing, rule *(5)* is adopted:

(5) "*Given two* `Concept` *nodes* `<A>` *and* `<B>` *such that* `<A> it:broaderInstantial <B>`*: if exists a* `Concept` *node* `<C>` *such that* `<B> skos:broader <C>`*, or* `<B> it:broaderGeneric <C>` *or* `<B> it:broaderPartitive <C>`*, then the same rule for NTP concepts is applied; otherwise,* `<B>` *becomes an instance of the OWL class defined for node* `<A>`".

According to rules (4) and (5), in the simplest case (NTP converted as individuals) the following OWL 2 DL is obtained:

```
<owl:Class rdf:ID="#United_States_of_America">
  <owl:subClassOf rdf:parseType="#Countries">
  </owl:subClassOf>
</owl:Class>
<United_States_of_America rdf:ID="Alabama">
...
</United_States_of_America>
<United_States_of_America rdf:ID="Alaska">
...
</United_States_of_America>
<United_States_of_America rdf:ID="Arizona">
...
</United_States_of_America>
```

**Thesaurus Array.** Another type of node of the RDF graph which is translated in our framework is `it:ThesaurusArray`, i.e. a node label used to represent a collection of siblings concepts with a common characteristic of division. As reported in ISO 25964, a node label is put between two concepts related by the NT/BT relation. A thesaurus array is not a concept itself. Moreover, following ISO 25964 definition for node labels, all the concepts under a node label represent disjoint classes. An example from the EARTh thesaurus [1] follows.

```
Forecasting
  [Forecasting by length]
    NT Long-term forecasting
    NT Short-term forecasting
  [Forecasting by target]
    NT Drought forecasting
    NT Earthquake forecasting
```

From this example, as stated in Sec. 11 of [12], it can be asserted that the concepts following the "by" clause in the node label represent a property of the concept "Forecasting". Thus, we map "length" as an object property between the OWL class for concept "Forecasting" and the two OWL classes for the concepts "Long-term forecasting" and "Short-term forecasting". Specifically, the following rules are applied:

(6) "*Given two* `Concept` *nodes* `<A>` *and* `<B>` *and a* `ThesaurusArray` *node* `<TA>` *such that* `<A> skos:broader <B>` *(or* `<A> it:broaderGeneric <B>` *or* `<A> it:broaderPartitive <B>`*), and* `<A> it:subordinateArray <TA>` *and* `<TA> skos:member <B>`*, then* `<B>` *is mapped as a subclass of the OWL class defined for node* `<A>` *and an* `owl:ObjectProperty` *is defined between* `<A>` *and* `<B>` *with* `rdf:ID="<TA>"`".

(7) "*Given two* `Concept` *nodes* `<B1>` *and* `<B2>` *and a* `ThesaurusArray` *node* `<TA>`*, such that* `<TA> skos:member <B1>, <B2>`*, then the constructor* `owl:disjointWith` *must be used between the resulting OWL classes for nodes* `<B1>` *and* `<B2>`".

According with these rules, the following OWL 2 DL conversion is obtained:

```
<owl:Class rdf:ID="#Forecasting">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:ID=
       "#Forecasting_by_length">
      <owl:oneOf rdf:parseType="Collection">
        <owl:Class rdf:ID="#Long-term_forecasting">
        </owl:Class>
        <owl:Class rdf:ID="#Short-term_forecasting">
        </owl:Class>
      </owl:oneOf>
    </owl:Class>
    <owl:Class rdf:ID="#Forecasting_by_target">
      <owl:oneOf rdf:parseType="Collection">
       <owl:Class rdf:ID="#Drought_forecasting">
        </owl:Class>
        …
      </owl:oneOf>
    </owl:Class>
  </owl:unionOf>
</owl:Class>
```

Since a `ConceptScheme` can be divided in `ConceptGroup` nodes, each one typically used to represent a self-contained thesaurus identified by a typical subject area (e.g. to represent a domain, theme, microthesaurus or simply a "group"). Thus, in our opinion, all the OWL classes defined from the `it:Concept` nodes belonging to an `it:ConceptGroup` are simply enriched with a datatype property via `owl:topDataProperty`, by considering the concept group as a literal that annotates all the concepts from that group.

Finally, observe that all the SKOS labels associated to a `skos:Concept` node in the graph are preserved in the ontology for completeness. Labels are used, in our framework, primarily to formalize a USE/UF relation between the terms associated to the same concept.

## 5 Conclusions and Future Work

In this paper we described an alternative application framework to convert thesauri into OWL 2 ontologies. The framework includes a translator module that, starting from an ISO 25964 compliant thesaurus, applies conversion rules to obtain the ontology. From the methodological point of view, along with the classical conversion rules focused on the hierarchical relations, the proposed framework introduced some additional conversion rules for thesaurus entities (e.g. facets, thesaurus arrays, partitive relations) that enable it to provide a stronger formalization.

As stated above, in this paper we focused only on some of the entities introduced or updated by the new ISO standard. In fact, the framework currently does not support some semantic relations, like USE+/UF+, and RT, nor it is designed to handle with interoperability between controlled vocabularies. These features are part of an extension of the framework currently under development.

Moreover, since at the moment very few example of ISO 25964 fully compliant thesauri exists (e.g. AAT), another interesting research stream is related to the necessity for new modules for the standardization of existing thesauri into ISO 25964 compliant ones. This will allow for a complete evaluation of the proposed framework.

## Acknowledgements

This work is supported by the "Science & Technology Digital Library" (S&TDL) Project, funded by the Agency for Digital Italy (AgID).

## References

1. R. Albertoni, M. De Martino, S. Di Franco, V. De Santis, and P. Plini. EARTh: an Environmental Application Reference Thesaurus in the Linked Open Data Cloud. In Semantic Web Journal, 2012, vol. 5(2): 165-171.
2. ANSI/NISO. Documentation – Guidelines for the construction, format, and management of monolingual controlled vocabularies. 2005. Report ANSINISO Z3919.
3. N. Aussenac-Gilles, S. Despres, and S.Azulman. The TERMINAE method and platform for ontology engineering from texts. In Proceedings of the Conference on Ontology Learning and Population: Bridging the gap between Text and Knowledge, Amsterdam, NL, 2008, pp. 199–223.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. In *The Scientific American*, 2001, vol. 284(5), pp. 34–43.
5. British Standards Institution (BSI). Documentation - Structured vocabularies for information retrieval. Guide. Exchange formats and protocols for interoperability. 2008. Report BS 8723-5.
6. S.G. Dexter Clarke and M. L. Zeng. From ISO 2788 to ISO 25964: The evolution of thesaurus standards towards interoperability and data modelling. In *Information Standards Quarterly (ISQ)*, 2012, vol. 24, n. 1.

7. T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing?. In International Journal of Human-Computer Studies, 1995, vol. 43(5–6), pp. 907–928.
8. M. Hepp and J. de Bruijn. GenTax: A Generic Methodology for deriving OWL and RSF-S Ontologies from Hierarchical Classifications, Thesauri and Inconsistent Taxonomies. In The Semantic Web: Research and Applications, E. Franconi, M. Kifer, M. Michael and W. May (eds.) Heidelberg, DE: Springer, 2007, pp. 129–144.
9. D. Kless, L. Jansen, J. Lindenthal, and J. Wiebensohn. A method for re-engineering a thesaurus into an ontology. In Proceedings of International Conference on Formal Ontology in Information Systems (FOIS 2012), Gray, AU, 2012, pp. 133-146.
10. D. Kless and S. Milton. Comparison of thesauri and ontologies from a semiotic perspective. In Proc. of the 6th Australian Ontology Workshop (AOW 2010), T. Meyer, M. A. Orgun and K. Taylor (eds.) Adelaide, AU: Australian Computer Society, 2010, pp. 35–44.
11. A. Isaac and J. De Smedt. ISO 25964 SKOS extension. 2013. Available: http://purl.org/iso25964/skos-thes.
12. International Standard Organization (ISO). Documentation - Thesauri and interoperability with other vocabularies: Part 1, Thesauri for information retrieval, 2011. Report ISO 25964.
13. International Standard Organization (ISO). Documentation. Thesauri and interoperability with other vocabularies: Part 2, Interoperability with other vocabularies, 2013. Report ISO 25964.
14. International Standard Organization (ISO). Documentation – Guidelines for the establishment and development of multilingual thesauri, 1985. Report ISO 5964.
15. International Standard Organization (ISO). Documentation – Guidelines for the establishment and development of monolingual thesaurus, 1986. Report ISO 2788.
16. P. Li and Y. Li. On Transformation from the thesaurus into domain ontology. Advanced Materials Research, 2013, vol. 756 – 759, pp. 2698–2704.
17. S. Mouhim, T. Khalid, C. Cherkaoui, H. Douzi, and D. Mammas. A methodological approach for converting thesaurus into domain ontology: application to tourism. In International Journal of Engineering and Innovative Technology, 2013, vol. 3(5), pp. 315–319.
18. A. Shiri and C. Revie. Thesauri on the Web: current developments and trends. In Online In-formation Review, 2000, vol. 24, no. 4, pp. 273–280.
19. D. Soergel, B. Lauser, A Liang, F. Fisseha, J. Keizer, and S. Katz. Reegineering Thesauri for New Applications: The AGROVOC example. In Journal of Digital Information, 2004, vol. 4(4).
20. B. M. Villazón Terrazas. A method for Reusing and Re-engineering Non-ontological Resources for Building Ontologies. Ph.D. Disseertation, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Madrid, ES, 2011.
21. B. M. Villazón Terrazas and Freddy Priyatna. Building Ontologies by using Re-engineering Patterns and R2RML Mappings. In Eva Blomqvist, Aldo Gangemi, Karl Hammar and, Marìa del Carmen Suàrez-Figueroa (eds.). Proceedings of the 3rd Workshop on Ontology Patterns, Boston, USA, November 12, 2012, CEUR WS, vol. 929, pp.109-120, 2012.
22. B. J. Wielinga, A. T. Schreiber, J. Wielemaker, and J. A. C. Sandberg. From thesaurus to on-tology". In Proceedings of the 1st International Conference on Knowledge Capture (K-CAP '01), Victoria, CA, pp. 194–201, October 2001.
23. L. Will. The ISO 25964 data model for the structure of an information retrieval thesaurus. In Bulletin of the American Society for Information Science and Technology, 2012, vol. 38(4), pp. 48–51.

# Digging Ontology Correspondence Antipatterns

Anselmo Guedes, Fernanda Baião and Kate Revoredo

Department of Applied Informatics. Federal University of the State of Rio de Janeiro – UNIRIO, Rio de Janeiro, Brazil
{anselmo.guedes, fernanda.baiao, katerevoredo}@uniriotec.br

**Abstract.** A correspondence antipattern is a set of generic correspondences between two ontologies that represents an incorrect alignment. It is useful to help identify incorrect correspondences between two ontologies, thus improving the Ontology Matching process. The specification of a correspondence antipattern requires the identification and correct understanding of a relevant alignment problem, and its representation in a proper modeling language. In this work we investigate the last three editions of OAEI challenge datasets so as to identify correspondence antipatterns from frequent and recurring errors; some of the the resulting antipatterns are presented and discussed.

**Keywords:** ontology matching, correspondence antipatterns, inconsistent alignment.

## 1 Introduction

As the research and practice on Ontology become more popular and evolve, several ontology artifacts arise for the same universe of discourse. However, they differ among each other in several perspectives, such as distinct representation languages (syntactic heterogeneity), variations in names referring to the same entity (terminological heterogeneity), different conceptualizations for the same domain (conceptual heterogeneity) and entities being perceived differently (semiotic heterogeneity) [7]. The Ontology Matching area [7][8] deals with all these problems, being considered by many authors the key element for heterogeneity reduction between ontologies.

The Ontology Matching task consists in identifying the correct correspondences among entities of multiple ontologies, which it is a necessary condition for establishing the interoperability among them [8]. A number of techniques can be used to identify correspondences between the entities of two ontologies, including the analysis of subsumption between classes and the similarity between the entity names. However, current results of state-of-the-art techniques are neither complete nor precise, i.e., they are not able to identify all existing correspondences between two ontologies and sometimes suggest correspondences that do not exist [9]. With regard to precision errors, suggesting a correspondence that does not exist may lead to either logical or ontological incompatibilities.

On the other hand, in the context of software development, *antipatterns* are considered a valuable tool for the identification of bad or incorrect practices in the software

development process. Antipatterns prevent or hamper a good execution of the software development or maintenance process. In the context of ontology matching, bad solutions consist of incorrect (including missing) or problematic correspondences. A correspondence antipattern is a matching model for identifying problematic correspondences that may occur repeatedly in ontology matching processes. A correspondence antipattern may be useful in several scenarios in which Ontology Matching is applied (such as in ontology merging, ontology comparison, query translation), since it helps refining an alignment produced by an ontology matching tool.

Looking for correspondence antipatterns, we "dig" the alignments available by OAEI and apply a methodology previously proposed in [11] for building correspondence antipatterns.

This work is divided as follows: Section 2 shows an overview about ontology correspondence antipatterns, Section 3 presents how we "dig" some correspondence antipatterns from the data published by OAEI, Section 4 presents related works and, finally, Section 5 points final considerations of this work.

## 2 Correspondence Antipatterns

Ontology matching identifies correspondences between the entities of multiple ontologies, and it is a necessary condition to establish interoperability between them [8]. According to Euzenat [7], technically the ontology matching process occurs by taking two ontologies $O$ and $O'$ as input, optionally considering a set of resources $r$, a set of parameters $p$ and an initial alignment $A$. The result of this process is an alignment $A'$ between the ontologies $O$ and $O'$, and may be represented as $A' = f(O, O', A, p, r)$. Basically, ontology matching is a process in which semantic links between entities of ontologies are established. This process results in a set of semantic links, where each semantic link is called a correspondence. The set of correspondences is called an alignment. Correspondences may stand for several relations, such as equivalence or subsumption [7]. In this work, we consider only equivalence correspondences.

Due to possible precision errors that every ontology alignment tool is subject to, it may be the case that a correspondence included in an ontology alignment is not correct. Take, for example, a real problem illustrated in Figure 1, showing an alignment problem that occurs in the last three OAEI[1] editions, between *ConfOf* and *Edas* ontologies. The Ontology Alignment Evaluation Initiative (OAEI) is a coordinated international initiative whose goal is to evaluate the strengths and weaknesses of the ontology alignment tools. OAEI organizes annual campaigns addressing several domains, and publishes the results of the evaluated tools. The correspondence between the *ConfOf*.*Conference* and *Edas*.*Conference* classes is a problematic one. Let's analyze this case: suppose that $x$ is an instance of *Edas*.*Conference*. Since an equivalent relationship between the entities *Edas*.*Conference* and *ConfOf*.*Conference* has been established, we may deduce that there is a possible world $w$ in which $x$ is an instance of *ConfOf*.*Conference* as well. Since *ConfOf*.*Conference* is a specialization of *Con-*

[1] http://oaei.ontologymatching.org/

*fOf.Event*, *x* is necessarily an instance of *ConfOf.Event* in *w*. We also notice that there is an equivalence correspondence established between *ConfOf.Event* and *Edas.Conferece_Event*. Thus, *x* is also an instance of *Edas.Conference_Event* in *w*. However, considering that *Edas.Conference_Event* and *Edas.Conference* are disjoint classes, there should be no possible world in which *x* instantiates both *Edas.Conference* and *Edas.Conference_Event* simultaneously, which leads to a contradiction, thus evidencing an alignment problem.

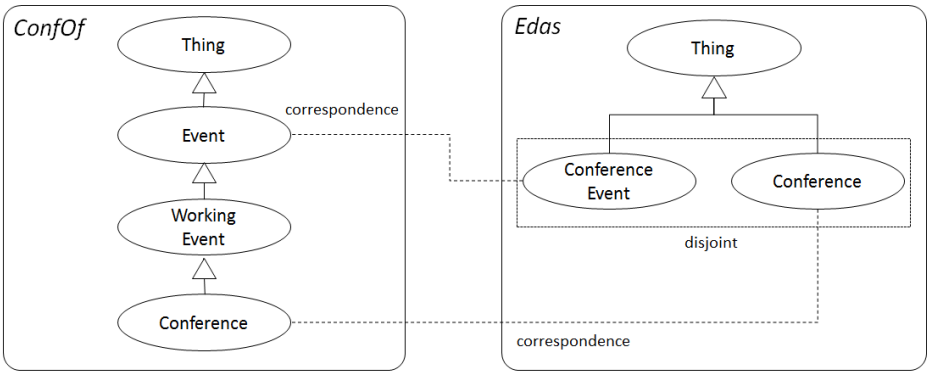


**Fig. 1.** Fragment of two ontologies and an alignment problem.

Patterns assist in building a collective experience based on the skills of domain specialists. On the other hand, an antipattern is a description of a given solution to a common problem that generates, definitely, negative consequences.

Given two ontologies *O* and *O'* to be aligned, a correspondence antipattern is a set of generic, domain-independent correspondences and/or non-correspondences between the entities of *O* and *O´* that lead to a contradiction. The purpose of a correspondence antipattern is, then, to help domain specialists in identifying a mismatch (a wrong correspondence) within an alignment.

We may generalize the example scenario illustrated in Figure 1 as follows: Consider a class e1 in an ontology o1 that is a subclass of a class e2, which in turn is subclass of a class e3 in o1. If class e3 in the ontology o1 is equivalently correspondent to class e2 in ontology o2, and classes e1 (from ontology o2) and e2 (from ontology o2) are disjoint, then class e1 from ontology o1 cannot equivalently match class e1 from ontology o2. As shown in [28], this correspondence antipattern can be represented as follows:

$$\{(?o1{:}?e1 \equiv ?o2{:}?e1) \sqcap (?o1{:}?e1 \sqsubseteq ?o1{:}?e2) \sqcap (?o1{:}?e2 \sqsubseteq ?o1{:}?e3) \sqcap (?o1{:}?e3 \equiv ?o2{:}e2) \sqcap (?o2{:}?e1 \sqcap ?o2{:}?e2 \sqsubseteq \bot)\} \quad (1)$$

## 3 Digging Correspondence Antipatterns

As shown in [11], for the development of correspondence antipatterns, the first step is to have the correct understanding of the problem being treated. When properly understood, the identified problem can result in correspondence antipatterns templates. Figure 2 presents the methodology proposed in [11], which can assist in the construction of a correspondence antipattern. This methodology focuses on responding to key issues which are essential for an antipattern identification.

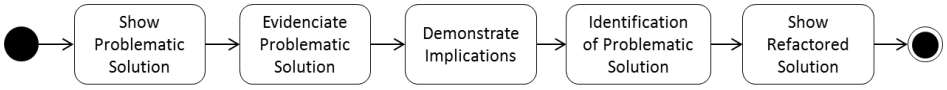


**Fig. 2.** Methodology to build a correspondence antipattern.

The methodology was applied on the results provided by the OAEI in the last three editions (2011.5, 2012 and 2013). The identification of correspondence antipatterns considered recurring incorrect correspondences generated by the evaluated tools. We identified incorrect correspondences by comparing tool results with the reference alignment published by OAEI. Each step of this process will now be briefly explained and illustrated in the OAEI scenario.

**First step**: *Show problematic solution*. The first step towards the construction of correspondence antipatterns is the correct understanding of the problem being treated. To start the search for correspondence antipatterns, the first step was the identification of incorrect correspondences, or false positives, in the set of selected alignments. False positives are the correspondences found by the evaluated tools that are not in the reference alignments. Within the universe of identified incorrect correspondences, we selected those that most frequently occurred (i.e., that were identified by many of the evaluated tools). We selected 40 incorrect correspondences, which were the ones that occurred over 50% of the analyzed alignments, as shown in Table 1. The columns *Ontology 1* and *Ontology 2* denotes the ontologies being aligned and the columns *Entity 1* and *Entity 2* denotes the entities involved in the incorrect correspondences found. The *Total Problems* column shows the quantity of alignments analyzed in which the incorrect correspondence was found. The *Total Alignments* column shows the quantity of alignments analyzed. *Percent* is calculated as *Total Problems* / *Total Alignment*.

**Table 1.** Inconsistent correspondences found in the set of alignments.

| Error Nº | Ontology 1 | Ontology 2 | Entity 1 | Entity 2 | Total Problems | Total Alignments | Percent |
|---|---|---|---|---|---|---|---|
| 1 | Conference | Ekaw | Invited talk | Invited Talk | 53 | 56 | 95% |
| 2 | Cmt | Iasted | Document | Document | 53 | 57 | 93% |
| 3 | Edas | Ekaw | Presenter | Presenter | 53 | 57 | 93% |
| 4 | Iasted | Sigkdd | Document | Document | 53 | 57 | 93% |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | Conference | Ekaw | Conference partic-ipant | Conference Participant | 52 | 56 | 93% |
| 6 | Edas | Iasted | Person | Person | 52 | 57 | 91% |
| 7 | Conference | Iasted | Presentation | Presentation | 52 | 56 | 93% |
| 8 | Conference | ConfOf | Conference | Conference | 52 | 56 | 93% |
| 9 | Edas | Ekaw | Conference | Conference | 52 | 57 | 91% |
| 10 | Cmt | Conference | Reviewer | Reviewer | 51 | 56 | 91% |
| 11 | Conference | Edas | Conference | Conference | 51 | 56 | 91% |
| 12 | ConfOf | Edas | Conference | Conference | 50 | 57 | 88% |
| 13 | Conference | Ekaw | Conference | Conference | 49 | 56 | 88% |
| 14 | Edas | Ekaw | ConferenceSession | Conference Session | 48 | 57 | 84% |
| 15 | Cmt | ConfOf | Paper | Paper | 47 | 57 | 82% |
| 16 | Conference | Ekaw | Paper | Paper | 47 | 56 | 84% |
| 17 | Conference | Sigkdd | Conference | Conference | 47 | 56 | 84% |
| 18 | Cmt | Conference | Paper | Paper | 47 | 56 | 84% |
| 19 | ConfOf | Edas | hasEmail | hasEmail | 46 | 57 | 81% |
| 20 | ConfOf | Ekaw | Paper | Paper | 46 | 57 | 81% |
| 21 | Iasted | Sigkdd | pay | pay | 44 | 57 | 77% |
| 22 | ConfOf | Edas | hasPhone | hasPhone | 43 | 57 | 75% |
| 23 | Cmt | Sigkdd | name | Name | 43 | 57 | 75% |
| 24 | Iasted | Sigkdd | obtain | obtain | 42 | 57 | 74% |
| 25 | Cmt | ConfOf | writtenBy | writtenBy | 41 | 57 | 72% |
| 26 | ConfOf | Edas | hasPostalCode | hasPostalCode | 41 | 57 | 72% |
| 27 | ConfOf | Edas | hasStreet | hasStreet | 40 | 57 | 70% |
| 28 | Cmt | Sigkdd | date | Date | 40 | 57 | 70% |
| 29 | ConfOf | Edas | hasTopic | hasTopic | 39 | 57 | 68% |
| 30 | mouse | human | MA 0000065 | NCI C12685 | 39 | 45 | 87% |
| 31 | ConfOf | Edas | hasCountry | hasCountry | 39 | 57 | 68% |
| 31 | mouse | human | MA 0000323 | NCI C12378 | 39 | 45 | 87% |
| 33 | Cmt | Ekaw | writtenBy | writtenBy | 38 | 57 | 67% |
| 34 | ConfOf | Ekaw | writtenBy | writtenBy | 38 | 57 | 67% |
| 35 | mouse | human | UNDEFINED part of | UNDEFINED part of | 37 | 45 | 82% |
| 36 | Conference | Iasted | is given by | is given by | 37 | 56 | 66% |
| 37 | mouse | human | MA 0000003 | NCI C12919 | 36 | 45 | 80% |
| 38 | Cmt | Edas | email | hasEmail | 31 | 57 | 54% |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 39 | Conference | Edas | Call for paper | CallForPapers | 29 | 56 | 52% |
| 40 | Conference | Edas | has an email | hasEmail | 27 | 56 | 48% |

**Second Step:** *Evidentiate problematic solution*. For a solution to be considered problematic, this should in fact occur [11]. Table 1 confirms that these errors are recurrent. The *Total Problems* column of Table 1 shows the total occurrences of the correspondence in the last three editions of the OAEI.

**Third Step:** *Demonstrate Implications*. For each incorrect correspondence, the error and its implications are analyzed according to the classification of types of inferences examined in [10]. Some of the errors found and their implications are presented as follows.

*o*

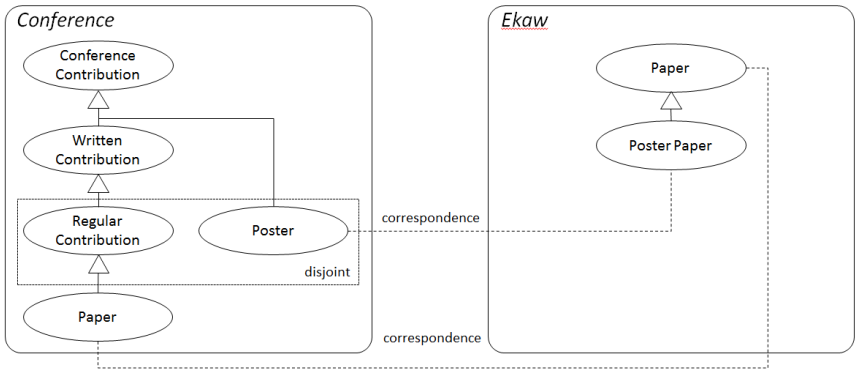


**Fig. 3.** Alignment problem between Conference and Ekaw ontologies.

Error Number 16: In the set of alignments analyzed, the correspondence ⟨*conference.paper, ekaw.paper,* ≡, _⟩ occurs 47 times. By analyzing the correspondence together with the aligned ontologies we identified the following problem: let *e1* be a class in an ontology *o1* which is subclass of a class *e2*, which in turn is a disjoint class of a class *e3*, also in ontology *o1*. If class *e1* in ontology *o1* equivalently corresponds to class *e1'* in ontology *o2*, class *e2* in ontology *o1* corresponds to class *e2* in ontology *o1* and class *e2'* in *o2* is a subclass of *e1* in ontology *o1*, then there is a contradiction (more specifically, a disjointness-subsumption contradiction alignment problem [10]). Figure 3 shows the case identified on the correspondence number 16, where the above problem occurs.

Error Number 20: In the set of alignment analyzed, the correspondence ⟨*confof.paper, ekaw.paper,* ≡, _⟩ occurs 46 times. By analyzing the correspondence together with the aligned ontologies we established the follow problem: let *e1* be a class in ontology *o1* that is disjoint with class *e2* in the same ontology *o1*, and a class *e1'* in ontology *o2* that specializes class *e2*' in the same ontology *o2*. If class *e1* in *o1* equivalently corresponds to class *e1*' in *o2* and class *e2* in *o1* equivalently corresponds to class *e2*' in *o2*, then there is a contradiction (a disjointness-subsumption contradiction

alignment problem [10]). Figure 4 shows the case identified on the correspondence number 20, where the above problem occurs.

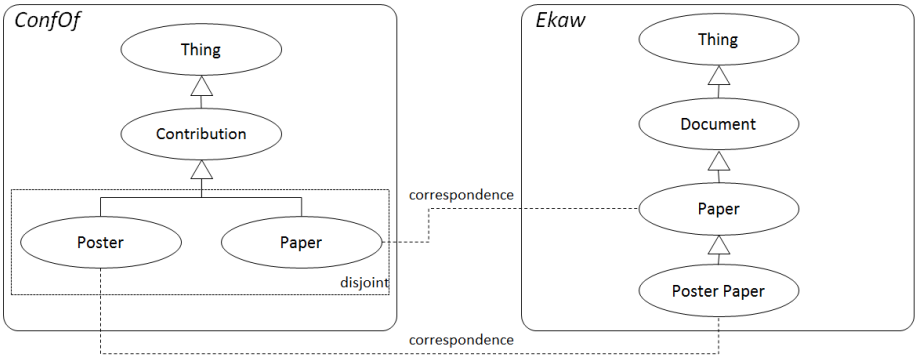


**Fig. 4.** Alignment problem between ConfOf and Ekaw ontologies.

Error Number 25: In the set of alignment analyzed, the correspondence ⟨*cmt.writtenBy, confof.writtenBy,* ≡, _⟩ occurs 41 times. By analyzing the correspondence together with the aligned ontologies we established the following problem: let *p1* be a property in ontology *o1* that has class *e1* as its domain and class *e2* as its range, both in ontology *o1*, and a property *p1'* in an ontology *o2* that has class *e1'* as its domain class *e2*' as its range, both in ontology *o2*. If *p1* in *o1* equally corresponds to the property *p1'* in *o2*, but class *e1* in *o1* does not correspond to class *e1'* in *o2* or class *e2* in *o1* does not correspond to class *e2'* in *o2*, then there is a domain and range incompleteness alignment problem. Figure 5 shows the case identified on the correspondence number 25, where the above problem occurs.

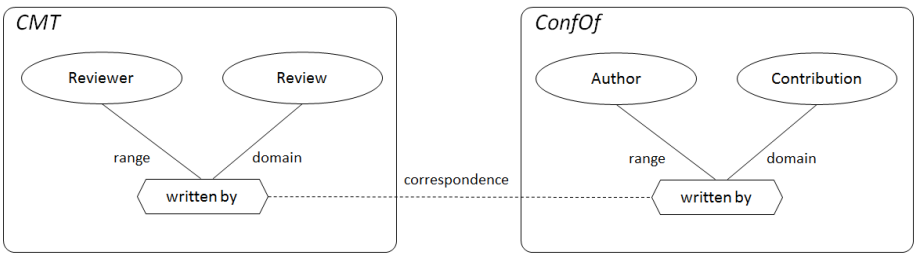


**Fig. 5.** Alignment problem between CMT and ConfOf ontologies.

Error Number 27: In the set of alignment analyzed, the correspondence ⟨*confof.hasStreet, edas.hasStreet,* ≡, _⟩ occurs 40 times. By analyzing the correspondence together with the aligned ontologies we established the following problem: let *p1* be a property in an ontology *o1* that has classes *e1* and *e2* as its domain, both in ontology *o1*, and a property *p1*' in an ontology *o2* that has as its domain a class *e1*' in ontology *o2*. If *p1* in *o1* equally corresponds to the property *p1'* in *o2* and class *e1'* in *o2* does not correspond to any domain class of *p1* in *o1*, then there is a domain and range in-

completeness alignment problem. Figure 6 shows the case identified on the correspondence number 27, where the above problem occurs.

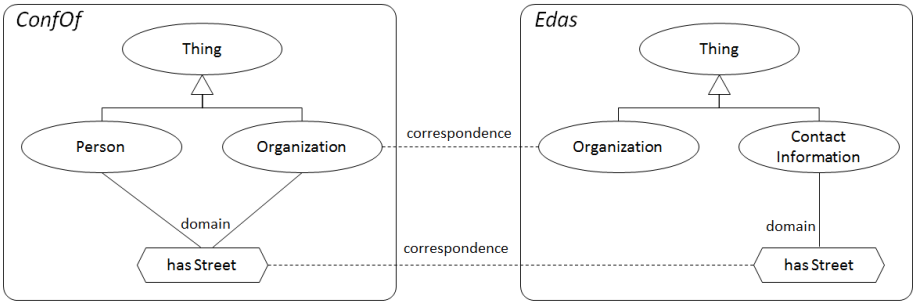


**Fig. 6.** Alignment problem between ConfOf and Edas ontologies.

**Fourth Step:** *Identification of the Problematic Solution*. The formal representation of how to identify an alignment problem is what gives life to correspondence antipattern. For each problem analyzed was created one correspondence antipattern, as summarized in Table 2.

**Table 2.** Antipatterns builded from alignment problems.

| Antipattern Item | Short Description |
|---|---|
| **Name** | OCA02 - Disjointness-subsumption contradiction with disjoint classes with subclasses. |
| **Antipattern general form** | (o1:e1 ≡ o2:e1') ⊓ (o2:e2' ⊑ o2:e1') ⊓ (o1:e1 ⊓ o1:e3' ⊆ ⊥) ⊓ (o1:e2 ≡ o2:e2') ⊓ (o1:e2 ⊑ o1:e3) |
| **Name** | OCA03 - Disjointness-subsumption contradiction with disjoint classes without subclasses. |
| **Antipattern general form** | (o1:e1 ≡ o2:e1') ⊓ (o2:e2' ⊑ o2:e1') ⊓ (o1:e1 ⊓ o1:e ⊆ ⊥) ⊓ (o1:e2 ≡ o2:e2') |
| **Name** | OCA04 - Domain and range incompleteness with no correspondence in domains or ranges |
| **Antipattern general form** | (o1:p1≡o2:p1') ⊓ ((o1:e1 ∈ domain(o1:p1) ⊓ o2:e1 ∈ domain(o2:p1') ⊓ ∄(o1:e1≡o2:e1')) ⊔ (o1:e2 ∈ range(o1:p1) ⊓ o2:e2' ∈ range(o2:p1) ⊓ ∄(o1:e2≡o2:e2'))) |
| **Name** | OCA05 - Domain and range incompleteness with no correspondence in domains |
| **Antipattern general form** | (o1:p1≡o2:p1') ⊓ (o1:e1 ∈ domain(o1:p1) ⊓ o2:e1' ∈ domain(o2:p1') ⊓ ∄(o1:e1≡o2:e1')) |

For the construction and computational representation of a correspondence antipattern, we adopt EDOAL (Expressive Declarative Ontology Alignment Language), an open and agnostic language [2] [11]. A fragment of the *OCA02 - Disjointness-*

*subsumption contradiction with disjoint classes with subclasses* correspondence anti-pattern EDOAL representation is illustrated as follows:

```
<map>
  <cell>
    <entity1><Class rdf:about="?o1:?e1"/></entity1>
    <entity2><Class rdf:about="?o2:?e1"/></entity2>
    <relation rdf:resource="equivalence"/>
  </cell>
  <cell>
    <entity1><Class rdf:about="?o2:?e2" /></entity1>
    <entity2><Class rdf:about="?o2:?e1" /></entity2>
    <relation rdf:resource="subsumedBy"/>
  </cell>
  <cell>
    <entity1><Class rdf:about="?o1:?e1" /></entity1>
    <entity2><Class rdf:about="?o1:?e3" /></entity2>
    <relation rdf:resource="disjoint"/>
  </cell>
  <cell>
    <entity1><Class rdf:about="?o1:?e2" /></entity1>
    <entity2><Class rdf:about="?o2:?e2" /></entity2>
    <relation rdf:resource="equivalence"/>
  </cell>
  <cell>
    <entity1><Class rdf:about="?o1:?e2" /></entity1>
    <entity2><Class rdf:about="?o1:?e3" /></entity2>
    <relation rdf:resource="subsumedBy"/>
  </cell>
</map>
```

**Fifth Step:** *Refactored Solution*. Refactoring in this case means repairing the alignment to eliminate logical inconsistencies. This is not a trivial activity, since there may exist many solutions for a specific scenario. Morevover, the best solution may also depend on the task, or even point some problem in the semantics of the correspondence. Therefore, this task is currently carried out by the specialist, with no automatic support. Further evolution of this approach will investigate automatic approaches for alignment refactoring.

## 4 Related Work

In ontology research, Ontology Design Patterns (ODPs) are an emerging approach that favors the reuse of encoded experiences and good practices. ODPs are modeling solutions to solve recurrent ontology development problems [1]. Compared with Software Engineering, where patterns have been used for a long period, patterns in Ontology Engineering are still in infancy [2]. The earliest works addressing the issue of patterns in Ontology Engineering are from the beginning of the 2000s. Sales and colleagues present semantic antipatterns for ontology engineering [3]. These antipatterns capture error-prone modeling decisions, which can result in the creation of models that allow for unintended model instances (representing undesired state of affairs). The antipatterns presented by [3] have been empirically elicited through an approach

of ontology conceptual models validation via visual simulation. In [12], the authors collect a list of common antipatterns that can be found in ontologies and that cause a large percentage of inconsistency problems Besides, their list some antipatterns that do not have an impact on the logical consequences of the ontology being developed, but are important to reduce the number of errors in the intended meaning of ontologies or to improve their understandability.

Correspondence patterns, proposed by [2], are essentially correspondences and sets of correspondences with generic entities. They act as role models to help find correspondences more precise than simply relate one entity to another one. Each correspondence pattern is a generic solution to a problem of alignment. Author of [2] proposed a library of correspondence patterns for design that represent solutions to different recurrent mismatches which are quite hard for matchers using usual matching techniques. Padilha [4] proposes design patterns and antipatterns for ontology alignment using high-level ontologies. The proposed design patterns were built based on the OntoUML [5], ontology modeling language which considers the ontological distinctions and axiomatic theories proposed in Foundational Ontology Unified (UFO). The patterns described are design patterns modeling, and there is no any kind of implementation thereof.

## 5 Final Considerations

Ontology matching is a very active research field in the scientific community, where various techniques and approaches have been proposed. However, existing tools are still likely to identify incorrect correspondences between the entities of the ontologies that are being aligned. The identification of recurrent errors may serve as input for the construction of correspondence antipatterns. A correspondence antipattern is a set of generic correspondences between two ontologies that represents an incorrect alignment. They assist in identifying incorrect correspondences in a given alignment, and should be computationally representated in an open and agnostic language.

OAEI is an important initiative that provides the community with the results os evaluations of several ontology matching techniques and tools. This published data constitutes a rich environment for analyzing recurrent errors in practical alignments.

In this work, the results provided by OAEI in three evaluation editions (2011.5, 2012 and 2013) were analyzed. The identified recurring alignment problems were considered and some correspondence antipatterns were specified and codified and EDOAL, following the methodology proposed in [11].

Future works include the exhaustive analysis and identification of correspondence antipatterns from other OAEI datasets, and the construction of a framework to make use of these antipatterns in refining ontology alignments.

## Acknowledgment

Fernanda Baião is partially funded by CNPq brazilian research council under the project 309069/2013-0.

## References


1. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: eXtreme Design with Content Ontology Design Patterns. In: Proc. Workshop on Ontology Patterns, Washington D.C., USA (2009)
2. Falbo, R., Barcellos, M., Nardi, J., Guizzardi, G.: Organizing Ontology Design Patterns as Ontology Pattern Languages. In: 10th Extended Semantic Web Conference (2013)
3. Sales, T., Barcelos, F., Guizzardi, G.: Identification of Semantic Antipatterns in Ontology-Driven Conceptual Modeling via Visual Simulation. 4th International Workshop on Ontology-Driven Information Systems (2012)
4. Padilha, N.: Padrões e antipadrões de correspondências para melhoria do alinhamento de ontologias bem fundamentadas. Dissertação de Mestrado. Universidade Federal do Estado do Rio de Janeiro (2013)
5. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models, Ph.D. Thesis, University of Twente, The Netherlands (2005)
6. Buschmann, F., Meunier, R., Rohnert, H., Sornmerlad, P., Stal. M.: Pattern-Oriented Software Architecture. A System of Patterns, John'Wiley & Sons Ltd., England (1996)
7. Euzenat, J., Shvaiko, P.: Ontology Matching, 2nd edition, Springer-Verlag, Berlin (2013)
8. Ehrig, M.: Ontology Alignment: Bridging the Semantic Gap, Springer Science + Business Media, LLC (2007)
9. Grau, B. C., Dragisic, Z., Eckert, K., Euzenat, J., Ferrara, A., Granada, R., Ivanova, V., Jiménez-Ruiz, E., Kempf, A. O., Lambrix, P.: Results of the Ontology Alignment Evaluation Initiative 2013, in *Proc*. *8th ISWC workshop on ontology matching (OM)*, pp. 61–100 (2013)
10. Jean-Mary, Y., Shironoshita, E., Kabuka, M.: Ontology matching with semantic verification. Journal Web Semantics: Science, Services and Agents on the World Wide Web archive. Volume 7 Issue 3, pp. 235-251 (2009)
11. Guedes, A., Baião, F., Revoredo, K.: On the Identification and Representation of Ontology Correspondence Antipatterns. 8th International Workshop on Modular Ontologies in 8th International Conference on Formal Ontology in Information System. *To be appear*. (2014)
12. Roussey, C., Corcho, O., Vilches-Blázquez, L.: A catalogue of OWL ontology antipatterns. In Proceedings of the fifth international conference on Knowledge capture (K-CAP '09), pp. 205-206 (2009)

# An Ontology Design Pattern for Cooking Recipes – Classroom Created

Monica Sam, Adila Alfa Krisnadhi, Cong Wang, John Gallagher, Pascal Hitzler

Department of Computer Science and Engineering, Wright State University, USA

**Abstract.** We present a description and result of an ontology modeling process taken to the classroom. The application domain considered was cooking recipes. The modeling goal was to bridge heterogeneity across representational choices by developing a content ontology design pattern which is general enough to allow for the integration of information from different web sites. We will discuss the pattern developed, and report on corresponding insights and lessons learned.

## 1 Introduction

The pattern which we will describe in this paper was developed as part of a class which the last named author conducted at Wright State University in Spring 2014. The class was called *Knowledge Representation for the Semantic Web*, and was an entry-level graduate class, with the goal of conveying to the students the fundamentals of Semantic Web knowledge representation languages.

The teacher had been experimenting in the past with a rather "classical" approach to teaching this subject, which essentially followed [1]. In this approach, topics progressed from light-weight to heavy-weight, starting with RDF and RDFS, their semantics and completion algorithms, the OWL syntax and intuitive semantics, followed by the introduction of description logics, their formal semantics and tableaux-based reasoning procedures. Practical modeling exercises progressed from shallow to deep, essentially starting with taxonomies which were then step by step (as material in the class progressed) expanded and adorned with more sophisticated axiomatization. Logical foundations were conveyed without proofs, as in the textbook, with the goal of allowing the students to develop a rough intuition of the logical underpinnings without diving too deeply into the technical details.

While the classes (and the textbook, for that matter) were well received by the students, these classes fell short of one particular goal which the teacher had: most students did *not* develop an intuition for formal semantics and logical deduction, i.e. the understanding of the languages remained at an informal and rather shallow and mostly syntactic level. While this is not necessarily a problem regarding useful learning outcomes for the students, the teacher was motivated to pursue alternative routes.

For the class in Spring 2014, contents and approaches were radically modified. Explicit treatment of RDF and OWL was deferred to a brief overview at

the end of the class, while the bulk of the material taught was on formal logic around description logics and rules (datalog and logic programming), plus some existential rules, with full formal proofs of (un)decidability results and of soundness and completeness of the presented algorithms. The main emphasis was thus to engulf the students in the formal matter, at which point the concrete W3C standards like RDF and OWL became almost afterthoughts, the syntax and semantics of which could be conveyed very quickly in the end as special cases of what had been dealt with on a more foundational level over the course of four months.

A second major change was adopted: The former class project, which was to built an ontology progressively starting from light-weight (taxonomy) to heavy-weight (OWL and rules axioms) was completely dropped. Instead, the whole class met for a 7-hour session (with breaks) to practice collaborative ontology design pattern (ODP) modeling. In the two previous class sessions, the students had received a brief introduction to ODPs, as well as a minimalistic primer on RDF and linked data. The primary examples discussed were the Semantic Trajectory pattern [2], the Cruise pattern [3] and from Linked MDB (linkedmdb.org). The students did know that they would collaboratively model a concrete ODP, but they did not learn before the class session what notion it would be.

For the modeling session, the class was split into two groups, which initially were to work independently. The charge was as follows.

*Design an ontology design pattern which can be used as part of a "recipe discovery" website. The pattern shall be set up such that content from existing recipe websites can "in principle" be mapped to the pattern (i.e., the pattern gets populated with data from the recipe websites). On the discovery website, detailed graph-queries (using the pattern) shall produce recipes from different recipe websites as results.*

Students were instructed to first look at some popular recipe websites, then to formulate, in natural language, some example queries to the website such as "I have cabbage and potatoes, how can I make that into a nice meal in 45 minutes?" They were then to develop a graph structure as basis for a pattern and check that their queries work with these.

In a second stage, the groups were mixed, with one representative from each group changing to the other group. This representative brought the original questions from his original group and both patterns were modified to accommodate the additional types of queries. The groups then added axioms in the form of OWL or rules to constrain the formal semantics. Figures 1 and 2 depict the two patterns the groups came up with, and we will discuss these further in Section 4.

In a third stage, each group was charged with describing how their pattern could be mapped to the other pattern, and asked to report on the problem points, i.e., where and why mapping of parts of the pattern did not really work.

The whole class finally merged all insights into one draft pattern, which is depicted in Figure 3 with only minor modifications – mostly regarding the choice of names for classes and properties, and a more detailed axiomatization. This resulting pattern is laid out in detail in this paper. To the teacher's own surprise,

the pattern turned out very well, and the class feedbacks were excellent. We will discuss further aspects of the class pattern modeling experience towards the end and discuss the resulting pattern.

## 2 Modeling Recipe

The term *recipe* has several contextual meanings. It can be defined in a general sense as a method to obtain a desired end. When used in the context of cooking, it is generally considered to be a set of instructions on preparing a culinary dish. As such, it could be viewed as an object with properties such as ingredients and time needed. Alternatively, it could be viewed as a *process*, which takes in some input, has a series of steps to be executed, and produces some output. The time taken to execute the steps and the utensils needed also help describe the recipe.

A cooking recipe, apart from the instructions and ingredients, sometimes contain information that help categorize it better for various needs. A recipe could be described as vegetarian or belonging to a particular course and appropriately categorized. Application-specific information may also be needed to tailor the recipe for specific purposes.

Modeling a piece of particular domain knowledge as an ODP typically requires involvement of domain experts. In modeling recipes for this paper, however, the presence of domain experts is not strictly required since information about recipes is widely available online and easily understandable by most lay people. Moreover, we are not so much interested in the actual resulting pattern, but rather on the experience of using ODP as a pedagogical element for teaching Semantic Web knowledge representation languages. In place of sessions with domain experts, a brainstorming session in the class was carried out to gather the information necessary to model recipe in this paper. The resulting pattern was obtained from scratch since the participants were not exposed to previous attempts for modeling cooking recipe or, in general, food and the related notions, some of which we survey below.

Cantais et al. [4] presented a Food ontology, which works together with the Diets and Product ontologies to provide a recommendation to users on a healthy choice of food in accordance to their health condition. They proposed a formalization based on a use case scenario over diabetic patients. Recipe is not modeled explicitly in these ontologies, although the recommender system that uses them can suggest combination of ingredients appropriate to the dieting requirements.

Mota and Agudo [5] presented an ontology of ingredients, which was used as a part of a recipe adaptation system. Recipe is not modeled as a process as in our pattern, but rather, as a simple collection of those ingredients that can be added when needed or removed when unwanted.

Ribeiro et al. [6] described a Cooking ontology as an example how a spoken dialogue system can be enriched with a domain knowledge. The ontology itself is very fine-grained and complex containing more than 1000 classes divided into seven ontology modules. Recipe is modeled here as a combination of preparation, cooking, and presentation phases, each of which consists of a sequence of tasks.

Other relevant notions such as ingredient, utensil, measure, and unit are also included to model various situations, which may involve variation of measures and units, classification of food items (e.g., alcoholic and non-alcoholic), and differences in courses of the food item due to geographical locations. Recipe can thus be seen as a process like in our pattern, although in our case, the pattern is not as fine-grained as this one to retain a relatively high degree of extensibility.

To populate our pattern (or suitable extension thereof) with data, one can in principle use an approach using indentation and tagging structures in XML documents to learn to extract recipe information in the form of semantic annotations from web pages [7].

## 3 Competency Questions

The two groups were first asked to look at existing recipe websites and formulate the kind of questions the information in the websites helped answer. Group A made the assumption that recipe websites would contain information in standard data types and tried to drill down information to the most primitive level. Accordingly, their questions were also more specific. In the following section, we will look at some of the questions each group came up with, and how the two groups modeled similar information differently. Some of the questions that Group A came up with were as follows:

**Question 1** "Breakfast dishes I can prepare with 2 potatoes and 100 grams of wheat flour."

Here, group A assumed that quantities such as 2 potatoes and 100 grams of wheat flour should be clearly and unambiguously presented in each website. Group A modeled this information using the FoodItem and quantityOfFood classes. Group B had included ingredient information in their Ingredient and Quantity classes.

**Question 2** "Gluten-free desserts with less than 100 calories."

Group A modeled all food categorization in a single class called FoodClass. Food classifications such as vegetarian, gluten-free; flavor information such as spicy, sweet; foods with classifications such as Easter eggs, cuisine and course information are modeled by this class. The above question would be answered by Group A's model using the FoodClass and NutritionInfo classes, and may be answered by the Keyword class in Group B's model.

**Question 3** "Mexican dishes which do not use chili."

An interesting aspect of the model of Group A is that both ingredients and food products are modeled as foods with quantities. This will be discussed later in detail. The above question would be answered by the FoodClass and isOptional property of the Ingredient class in Group A's model. Group B modeled this information using their Cuisine class. It might have been more difficult to query

for the absence of an ingredient using Group B's model.[1] Group A was certain that there should be a difficulty level associated with each recipe and included that in their model. They also associated an Author pattern with each of their recipes. This would help answer questions such as the following:

**Question 4** "Easy Gordon Ramsey breakfast dishes."

It was noted that the problem of uniformity in representing levels of difficulty across websites might show up when modeling this concept. Whereas Group A had an Author associated with each recipe, Group B did not. Therefore the above question couldn't be answered by their model. Cooking utensils needed and time to cook were some information that was agreed to be vital to answer questions such as the following:

**Question 5** "Grilled meat in less than 1 hour."

The above utensil and time information were modeled by both groups. Group A used the utensil and Time classes for this and Group B used CookingTool and OWL:Time classes for this. Group B also had some similar ideas as to what ought to be included in a model for recipe websites. They identified flavor, texture, cuisine and serving temperature as essential attributes of any recipe, to answer questions such as the following:

**Question 6** "Spicy Korean beef dishes."

For the above question, Group B's Cuisine, Flavor and Ingredient classes were modeled. Group A would have answered this question with the FoodClass and Ingredient classes.

**Question 7** "Crunchy brownie recipes."

Group B modeled the Texture class and Keyword class to answer queries such as the above, whereas Group A modeled this information using the FoodClass and Keyword classes.

**Question 8** "Cold appetizers."

Group B had a specific ServingTemperature class to answer queries on the temperature the dish should be served at, while Group A modeled such information only with FoodClass and Keyword classes. They also included cooking utensil and difficulty level:

**Question 9** "Baked/Mashed potatoes."

The above question would need information on the utensil to be used to prepare the recipe, since baking involves an oven and mashing presumably requires a masher or some similar utensil, and this information is modeled using the Utensil class by Group A and CookingTool class by Group B.

[1] (Local) closed world reasoning is of course also required for this type of query.

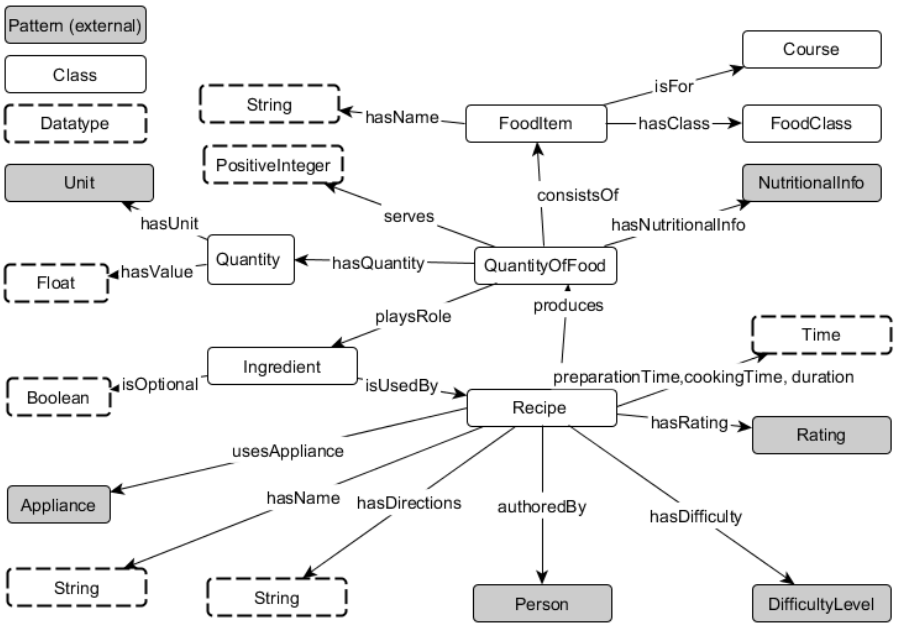


Fig. 1: Recipe modeled by group A.

**Question 10** "Easy desserts with less than 100 calories."

Both the groups had modeled a DifficultyLevel class and NutritionalInfo class. The above query would be answered by these classes. Some information such as nutritional information, time to cook, utensils needed, difficulty level were identified and modeled similarly by both groups.

## 4 In-class Modeling – A Discussion

Based on these competency questions, the two groups of students came up with two rather different drafts of a recipe pattern. These drafts are depicted in Figures 1 (Group A) and 2 (Group B).

Initially, both the groups modeled recipe as a class with relationships to other classes. So they defined properties that objects of the class would have. There were ingredients which became a property of the class and there were instructions that was also a property of the class, as was the dish that was produced. With this approach instructions were just another property of the recipe. An alternate way was proposed by the teacher, in which the recipe became a process, and therefore a description of the procedure became now a defining characteristic of it. The students then modified their patterns based on this thinking.

While Group A connected both ingredients and food products as belonging to a single category, which they called FoodItem, Group B continued to view

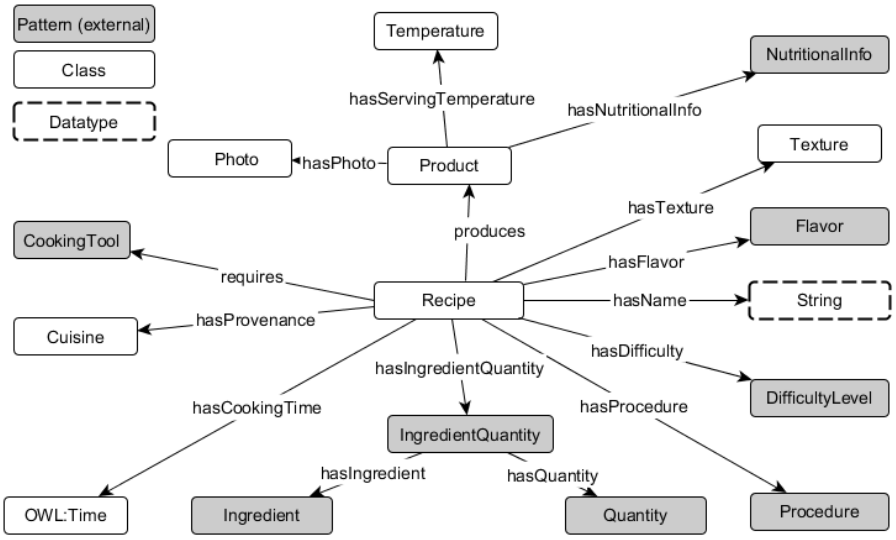


Fig. 2: Recipe modeled by group B.

them separately and modeled them as Product and Ingredient. Both the groups independently came to the conclusion that there had to be some modeling of the concept of quantity to be associated with FoodItems. Group A decided that quantity would in turn have a unit and a value associated with it, whereas Group B did not make that decision. Group A associated all the information related to categorization – based on spice levels, based on cuisine, based on dietary content as all belonging to a single class they called FoodClass. They had a separate class for the course of the dish the recipe was for. Group B decided to maintain separate, the notions of Cuisine, Flavor and Texture. Both the groups independently came up with the idea of NutritionalInfo that had nutritional information about the FoodItem produced as a result of the recipe. Both the groups came up with ideas of time associated with preparation and cooking and the appliances used to make the food item. Both the groups also came up with the idea of a difficulty level associated with the procedure. While the ingredient had an associated property which marked whether it was optional in the design of Group A, Group B had not modeled that information. Also Group A had an author associated with the recipe which Group B did not, while Group B had a Photo associated with the recipe that Group A did not. Group A also had a Rating associated with the recipe that Group B did not.

Of course, the two groups made different modeling choices, and as part of the class session, after producing the initial drafts just discussed, the two groups were asked to attempt to develop mappings, one group from version A to version B, and one group vice versa as the third stage of the class modeling session (see the overview in Section 1). They were charged with developing a loss-less mapping,

if possible, and to report on issues where such a mapping was not possible. This task aligned with the general charge for the modeling session (see Section 1): the charge called for a very general pattern to which content could always be mapped. So difficulties found in the mapping exercise would indicate too specific modeling choices.

A direct mapping could indeed be made between the two models for the following concepts pairs (and corresponding properties): Recipe – Recipe; Ingredient as QuantityOfFood – Ingredient with IngredientQuantity; Quantity – Quantity; Appliance – CookingTool; DifficultyLevel – DifficultyLevel; Name – Name; Directions – Procedure; NutritionalInfo – NutritionalInfo; Time – OWL:Time; FoodClass – Cuisine.

The individual pieces of information about the recipe that each model had which was not in the other model could not be accomodated unless the model was expanded. The author modeled in Group A's version, the photograph modeled in Group B's version could not be mapped directly. The unit and value associated with Quanitity in Group A's model would need an expansion of Group B's model. The Temperature modeled in Group B's model could not be accomodated in Group A's model. The Optional Concept associated with Group A's model could not be directly mapped into Group B's model. The Rating associated with Group A's model had no equivalent in Group B's model. The Course associated with Group A's model had no equivalent in Group B's model. The choice of Group A to use a class QuantityOfFood which can act both as recipe ingredient and as recipe product found general agreement, partially because of the perceived conceptual clarity of the modeling, partially because it was realized that sometimes the recipe product becomes in turn an ingredient in another recipe. So the approach by Group A was adopted for the final version.

The mapping attempts furthermore exposed strong ontological commitments regarding some classes used as the range for properties, e.g. demanding a Float as quantity value. In the ensuing discussion it was realized that the quantity value class may be a complex entity, i.e. a pattern in its own right. E.g., recipes may specify a "pinch of salt", or "salt and pepper to taste".

Missing information about Photograph in Group A's model and Author, Rating in Group B's model was filled by creating a new class called InformationObject which conceptualized information associated with the Recipe. The missing information in Group A's model about serving temperature, flavor and cuisine and in Group B's model about course and food class were modeled into another object called FoodClass. By associating a quantity with ingredient, it was decided that the need for an optional indicator field was removed.

The in-class discussion, in particular seeing the different modeling choices, the attempt and failure to produce mappings, and the final consensus to establish the final pattern draft (Figure 3) provided the students with an understanding of the difficulties of making, using, and reusing conceptual models, which would have been much more difficult to convey without a group modeling exercise.

| Name | Type | Explanation |
|---|---|---|
| *hasIngredient* | *Recipe* × *QuantityOfFood* | An ingredient of the recipe |
| *produces* | *Recipe* × *FoodItem* | A dish produced by the recipe |
| *consistsOf* | *QuantityOfFood* × *FoodItem* | The quantity of either an ingredient or dish |
| *classifiedAs* | *FoodItem* × *FoodClassification* | The classification of an ingredient or disth |
| *hasName* | *FoodItem* × *String* | The name of an ingredient or dish |
| *hasProcessDescription* | *Recipe* × *ProcessDescription* | the description of the recipe process |
| *hasTextualDescription* | *ProcessDescription* × *Document* | The text description of the recipe process |
| *preparationTime,cookingTime* | *Recipe* × *timeInterval* | The duration of the recipe process |
| *requires* | *Recipe* × *Utensil* | A utensil needed for the recipe |
| *hasName* | *Recipe* × *String* | The name of a recipe |
| *hasInformationObject* | *Recipe* × *InformationObject* | The information object of a recipe |
| *hasURL* | *InformationObject* × *URL* | The URL information of a recipe |
| *hasKeyword* | *InformationObject* × *String* | The keyword information of a recipe |
| *hasAuthor,hasRecommender* | *InformationObject* × *Person* | The author information of a recipe |
| *hasDifficultyLevel* | *InformationObject* × *DifficultyLevel* | The difficulty level information of a recipe |
| *hasRating* | *InformationObject* × *Rating* | The rating of a recipe |

Table 1: Basic relations present in the finalized model.

## 5 Finalized Pattern and OWL Formalization

We now present the finalized design pattern, based on the previously described conceptual foundations. The concrete OWL ontology, which was created using Protégé, can be found at http://www.pascal-hitzler.de/data/RecipeODP.owl. In order to answer the competency questions, an ontology design pattern needs to distinguish a number of relations. We introduce these abstract relations in Table 1, before formalizing them.

In the following, we respectively discuss the classes and properties within the pattern and formally encode them using the Web Ontology Language (OWL) [8]. We make use of Description Logics (DL) [1] notation, as we believe this improves the readability and understandability of the axioms presented. Note that tractable reasoning is important for producing an efficient implementation of the pattern. A schematic view of the pattern is shown in Figure 3.

***Recipe.*** A Recipe is a class which is described as a process. It may take as ingredients instances of QuantityOfFood and also produce instances of QuantityOfFood. The instructions on how to execute the recipe is provided in the ProcessDescription pattern which is assumed to have steps in a time-ordered fashion. A recipe also requires some utensils for execution which is modeled as a Utensil pattern, and requires some amount of time for execution described by a pattern named TimeInterval. These patterns are already assumed to exist in the domain and are left as conceptual elements that are not drilled down in detail for this exercise. Each instance of the recipe class also has a name which is of the String data type.

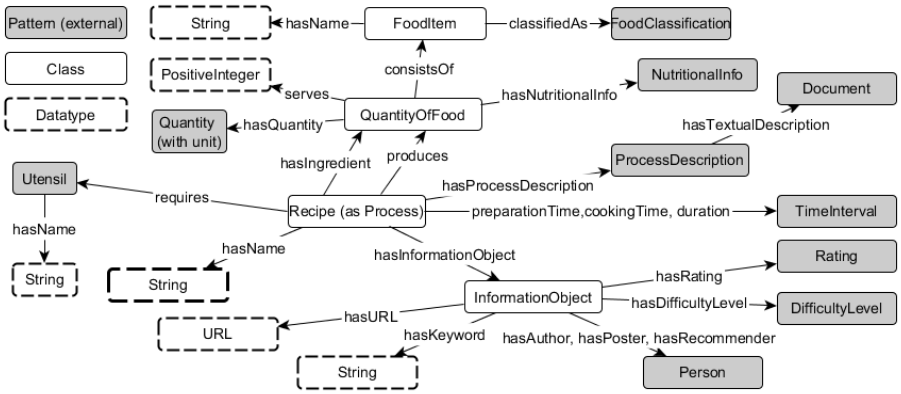


Fig. 3: Recipe as a process modeled along with other patterns (marked grey) needed to describe a recipe.

***QuantityOfFood.*** QuantityOfFood is a class that models the quantity of both ingredients and food products produced by the recipe. It can be described as serving a number of people which is of data type positive integer. It also has some nutritional information associated with it, which is modeled as a pattern called NutritionalInfo. The quantity it has is described using a pattern called quantity which is some quantity along with a unit.

***FoodItem.*** FoodItem is a class that models an ingredient or food product which in turn has its quantity described by QuantityOfFood. The FoodItem has a name which is modeled using a String data type and has a food classification which in turn is defined as a pattern named FoodClassification. FoodClassification has been modeled in a very broad sense to include such classification as vegetarian, vegan, gluten-free to seasonal, or occassion-specific information such as thanksgiving meals or even meals for a purpose such as lunch-box meals.

***InformationObject.*** InformationObject is a class that contains information needed to describe a recipe. Some of its properties are hasURL, hasKeyword (of String data type), hasAuthor, hasPoster, hasRecommender - which is a Person described by an external pattern, and a difficulty level, also modeled as an external pattern. It also includes Rating information, which also could pose challenges, if the rating systems are all not normalized.

**OWL Profile.** Full list of axioms for the pattern are given in Appendix A of the extended technical report available from http://www.pascal-hitzler.de/pub/2014-recipe-tr.pdf. From these axioms, we know our recipe pattern falls into the Description Logic $\mathcal{ELIF}(D)$. An $\mathcal{ELIF}$-Tbox can be reduced to an $\mathcal{ELI}$-Tbox whose size is linear in the size of the original one [9].

In $\mathcal{ELI}$, subsumption w.r.t. GCIs is ExpTime-complete [10], and $\mathcal{ELI}$ knowledge bases can be classified by a completion-rule based algorithm[2] [11,9]. However, if we rewrite range restrictions of properties to be of universal scope, e.g., $\exists \text{hasIngredient}^{-}.\top \sqsubseteq \text{QuantityOfFood}$, $\mathcal{ELI}$ can even reduced to $\mathcal{EL}$ [12], such that the subsumption checking can be done in polynomial time [10].

## 6 Informal Evaluation

In this section, we illustrate how our pattern can be specialized to the content of specific websites.

For the website *www.allrecipes.com*, most of the information given for recipes can be modeled based on our pattern, except for the addition of a couple of InformationObjects of type image and video. The recipes also have a Review associated with them, but do not contain information on difficulty level.

Another example is the website *www.bettycrocker.com*. The information on this website can be modeled by a slight extension of our pattern to include expert tips to the process description. Also, this website has review information, and no difficulty level.

A third example is taken from *www.epicurious.com*. This website uses information such as the main ingredients, the dietary considerations this recipe would meet – for example, being vegan or high fiber and the season associated with the dish to tag the recipe. Recipes are also categorized. The information in this website could be modeled by of course adding the Review InformationObject, removing the DifficultyLevel and adding in another InformationObject for popularity which lists the number of times the recipe had been downloaded. In all cases, the key page contents can be captured with specializations of our recipe pattern. Further details are provided in Appendix B of the extended technical report available from http://www.pascal-hitzler.de/pub/2014-recipe-tr.pdf.

## 7 Conclusions

The class exercise produced a reasonable outcome in terms of the resulting content pattern. The students experienced the power of collaborative modeling to obtain versatile patterns, and overall the experience was resulted in very positive feedbacks.

A lesson learned (for the teacher) is that it seems necessary to convey some amount of thorough logical underpinnings in order to effectively teach introductory ontology modeling: An understanding of logical reasoning with ontology axioms seemed to have helped in avoiding typical beginner's mistakes regarding ontology modeling, such as confusing part-of relationships with subclass relationships or mistakes arising from vacuous truth via the universal quantifier.

[2] A practical reasoner for this fragment, called CB reasoner, can be found at https://code.google.com/p/cb-reasoner/.

*Acknowledgements* This work was partially supported by the National Science Foundation under award 1017225 "III: Small: TROn – Tractable Reasoning with Ontologies."

## References

1. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRC Press (2010)
2. Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., Kolas, D.: A geo-ontology design pattern for semantic trajectories. In Tenbrink, T., Stell, J.G., Galton, A., Wood, Z., eds.: Spatial Information Theory – 11th International Conference, Conference on Spatial Information Theory 2013, Scarborough, UK, September 2-6, 2013. Proceedings. Volume 8116 of Lecture Notes in Computer Science., Heidelberg, Springer (2013) 438–456
3. Krisnadhi, A., Arko, R., Carbotte, S., Chandler, C., Cheatham, M., Finin, T., Hitzler, P., Janowicz, K., Narock, T., Raymond, L., Shepherd, A., Wiebe, P.: An ontology pattern for oceanographic cruises: Towards an oceanographer's dream of integrated knowledge discovery. OceanLink Technical Report 2014.1, available from http://www.oceanlink.org/ (2014)
4. Cantais, J., Dominguez, D., Gigante, V., Laera, L., Tamma, V.: An example of food ontology for diabetes control. In: Proceedings of the International Semantic Web Conference 2005 workshop on Ontology Patterns for the Semantic Web. (2005)
5. Mota, S.G., Agudo, B.D.: ACook: Recipe adaptation using ontologies, case-based reasoning systems and knowledge discovery. In Cordier, A., Nauer, E., eds.: Proceedings of the Cooking With Computers workshop. (2012) 41–45
6. Ribeiro, R., Batista, F., Pardal, J.P., Mamede, N.J., Pinto, H.S.: Cooking an ontology. In: Artificial Intelligence : Methodology, Systems, Applications, Springer (2006) 213–221
7. Ciancarini, P., Iorio, A.D., Nuzzolese, A.G., Peroni, S., Vitali, F.: Semantic annotation of scholarly documents and citations. In Baldoni, M., et al., eds.: AI*IA 2013: Advances in Artificial Intelligence – XIIIth International Conference of the Italian Association for Artificial Intelligence, Turin, Italy, December 4-6, 2013. Proceedings. Volume 8249 of Lecture Notes in Computer Science., Springer (2013)
8. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., eds.: OWL 2 Web Ontology Language: Primer. W3C Recommendation 27 October 2009 (2009) Available from http://www.w3.org/TR/owl2-primer/.
9. Vu, Q.H.: Subsumption in the Description Logic $\mathcal{ELHIFR}+$ w.r.t. General TBoxes. PhD thesis, Technical University Dresden (2008)
10. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In Kaelbling, L.P., Saffiotti, A., eds.: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005, Professional Book Center (2005) 364–369
11. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In Boutilier, C., ed.: Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. (2009) 2040–2045
12. Carral, D., Feier, C., Grau, B.C., Hitzler, P., Horrocks, I.: EL-ifying ontologies. In Demri, S., Kapur, D., Weidenbach, C., eds.: Automated Reasoning – 7th International Joint Conference, Vienna, Austria, July 19-22, 2014. Proceedings. Volume 8562 of Lecture Notes in Computer Science., Springer (2014) 464–479

# Ontology Patterns for Clinical Information Modelling

Catalina Martínez-Costa[1], Daniel Karlsson[2], Stefan Schulz[1]

[1]Institute for Institute for Medical Informatics, Statistics and Documentation, Medical University of Graz, Austria
[2]Department of Biomedical Engineering, Linköping University, Sweden

{catalina.martinez,stefan.schulz}@medunigraz.at, Daniel.Karlsson@liu.se

**Abstract.** Motivated by our experiences of representing clinical information using OWL DL, which often resulted in highly complex expressions, we propose the use of ontology content patterns to facilitate this task. They are based on a set of formal ontologies, constrained by the concepts and relations of a top-level one, which reduces arbitrariness in ontology design. We propose their application to information encoded by electronic health records specifications and ontology-based terminologies, in order to provide semantic interoperability across heterogeneously represented data, and to guide the creation of clinical models and detect semantic inconsistencies across them. We provide examples of their application to achieve the above mentioned tasks and discuss the limitations and further research issues.

**Keywords:** ontology content patterns, electronic health standards, SNOMED CT

## 1 Introduction

Despite a wide-spread use of computers in clinical documentation, the semantic interoperability of information kept in electronic health record (EHR) systems is insufficient [1]. A plurality of EHR representations together with medical terminologies like SNOMED CT [2], have been proposed in recent years to structure clinical information and to provide standardized codes for frequently used medical terms, respectively.

Existing EHR standards and medical terminologies were developed in isolation and major problems exist when they are combined. Projects such as the HL7 TermInfo [3] or more recently the Clinical Information Modeling Initiative (CIMI) [4] and the European network SemanticHealthNet [5], have attempted to provide solutions by addressing the lack of division between ontology-based medical terminologies and information models (provided by EHR representations). This is commonly known as the *boundary problem* [6].

TermInfo provides a set of rules for the combined use of the HL7 information model and SNOMED CT; CIMI proposes a set of modelling patterns, defined as clin-

ical models that are intended to act as guide for the creation of new ones. Clinical models constrain information model structures to represent particular data capture and communication use cases. In medicine it is often not possible to impose one universal data form, such as for recording diagnostic information. Thus, CIMI associates each clinical model with a set of iso-semantic models (models heterogeneously structured but with the same meaning), from which one is selected as the preferred one and mappings are established across them.

CIMI or HL7 based models that implement the TermInfo specification might work well in isolation, but semantic interoperability issues arise when interacting with others, which are not necessarily compatible, whilst the anticipation of all possible iso-semantic representations will lead to an explosion of models. The European network SemanticHealthNet addresses this problem by providing clinical model information structures with a set of expressions, based on a shared ontological framework. This framework allows representing both (ontology-based) medical terminologies and information models, and implements the classical distinction between ontology [7] (what exists – independently of being known or observed) and epistemology [8] (what is known, suspected, planned, etc.).

The inherent complexity of this representation is addressed by using semantic patterns as intermediate representations, which is the focus of this paper.

## 2 Background

### 2.1 EHR Structured Clinical Models

Several EHR standards and specifications propose representing clinical information by using clinical models based on a reference information model (RM). Clinical models, also known as archetypes (e.g. openEHR/ISO 13606 archetypes) [9,10] or HL7 CDA documents [11], constrain a set of standardized information structures provided by some reference model (RM), to represent EHR data. They are used for modeling particular use cases for clinical data capture and communication. As an example, the ISO 13606 archetype of **Fig. 1** constrains information structures (e.g. *CLUSTER*, *ELEMENT*, etc.) to represent a medical questionnaire consisting of questions groups. The use of terminologies and ontologies within clinical models is known as *terminology binding*. **Fig. 1** shows how the information structure *ELEMENT[at0003]* is bound to the SNOMED CT concept *Past history of diabetes mellitus*. Interpreted within the context of the clinical model, it is a question, and its allowed answers are yes / no.

In practice, the division line between ontologies and information models is often crossed both by ontologies (where they represent epistemic and temporal information aspects, such as “known present” or “past history of”) and by RMs and clinical models (where they carry their own ontology without reference to external standards, here the fact that it is a question).

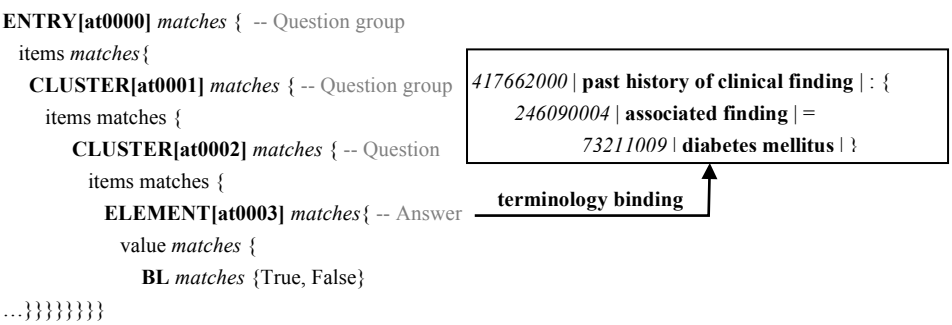


**Fig. 1.** (Left) ISO 13606 archetype excerpt to record questionnaire; (Right) Binding of an information structure to a SNOMED CT concept.

## 2.2 Ontology-based medical terminologies: SNOMED CT

Ontologies formally describe properties and relations of types of entities. Domain-independent categories, relations and axioms are typically provided by top-level ontologies [12], whereas the types of things that make up a domain are represented by domain ontologies. In the former one we find categories like *Process*, *Material entity*, *Quality*, etc., whereas in a clinical domain ontology we would find *Diabetes mellitus type 1*, *Left index finger*, or *Aspirin*, i.e. the classes of entities corresponding to the terms used in clinical documentation and reporting, and defined by the properties shared by all of their individual members.

Medical terminologies have evolved in the last years to include definitional knowledge about their terms, by using an ontological framework in order to help humans and computers to recognize the intended meaning of their terms, for proper coding of, retrieval of, and inferencing about biomedical data, as well as for maintenance of the terminology itself [13]. An example is SNOMED CT, a clinical terminology covering all aspects of clinical medicine, with about 300,000 representational units (called SNOMED CT concepts) and terms in several languages.

Due to the legacy of its predecessors, SNOMED CT does not only provide codes for clinical terms proper but also for contextual statements, which are often represented in information models. An example of this is the *Situation with explicit context* concept hierarchy (i.e. context model), in which we find terms such as *Suspected deep vein thrombosis* or *No past history of venous thrombosis*. We have largely harmonized the SNOMED CT content with basic top-level classes and relations of BioTopLite upper ontology [14] (e.g. *btl:Process*, *btl:Quality*, *btl:Condition*, *btl:Situation*, etc), in order to better distinguish clinical from information entities. Based on [15] we interpret SNOMED CT concepts from the *clinical finding* hierarchy as clinical situations and reinterpreted the SNOMED CT context model [16]. **Fig. 2** shows the OWL DL representation of a post-coordinated[1] expression that follows the context model and represents *past history of diabetes*. Past history is a temporal aspect that specializes the meaning of the finding diabetes mellitus.

[1] Post-coordination describes the representation of a term using a combination of two or more of them (e.g. *past history of clinical finding* and *diabetes mellitus*)

*'past history of clinical finding (situation)'*
and **RoleGroup** some (
(**'Associated finding (attribute)'** some *'Diabetes mellitus (disorder)'*) and
(**'Finding context (attribute)'** some '*Known present (qualifier value)'*) and
(**'Temporal context'** some *'In the past (qualifier value)'*) and
(**'Subject relationship context'** some *'Subject of record (person)'*))

**Fig. 2.** OWL DL SNOMED CT representation of an expression based on the post-coordination of two terms (**past history of clinical finding** and **diabetes mellitus** linked by the linkage concept **associated finding**). Terms using *italics* represent ontology classes, **bold face** represents ontology object properties.

# 3 Methods

A shared OWL DL [17] ontological framework is proposed that allows relating EHR information models with medical terminologies [18] in an unambiguous way. It is supported by a the use of semantic patterns in order to provide semantic interoperability across heterogeneously represented data and to guide the creation of clinical models and detect semantic inconsistencies across them.

## 3.1 Clinical Information Semantic Patterns

The semantic patterns we propose represent recurrent clinical information modelling aspects and can therefore be considered ontology design content patterns applied to clinical information. They are inspired by the experience of modelling clinical information based on ontologies. As ontology patterns they help to reduce the arbitrariness that exists when representing clinical information, by using a set of OWL DL formal ontologies as standard modelling framework [19].

Two ontologies, the SNOMED CT ontology (prefix sct) and an information ontology (prefix shn) are rooted in the biomedical top-level ontology BioTopLite (prefix btl). The use of BioTopLite standardizes the ontology development process, by providing a set of logical axioms which constrain how both ontologies are related. We use SNOMED CT as common reference point for representing the healthcare domain. The information ontology provides a set of classes that represent contextual and temporal information aspects (e.g. diagnostic information, past history, provisional, etc.) and refer to the SNOMED CT concepts.

Each pattern can be considered a small ontology based on the previous framework, to be used as a building block for a particular modelling use case. For that, they can be specialized and composed by following similar principles to object oriented languages [20].

According to [21], content patterns are language-independent and should be encoded in a high order representation language. Nevertheless, their representation in a logic-based language allows the use of DL reasoning [22], which can be used to ensure the consistency of the patterns and to allow inference-related tasks. On the left

side, **Fig. 3** shows the graphical representation of a pattern that represents the past history of some patient clinical situation. The right side, shows a concrete instance of that pattern that represents the statement "Past history of diabetes mellitus". Other examples of patterns are "Family history of clinical situation" or "Plan to perform some clinical process".

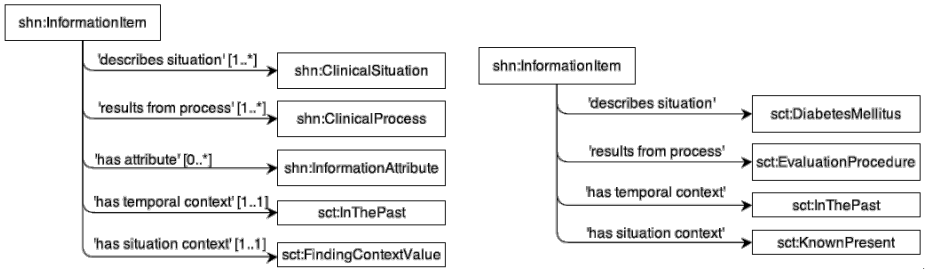


**Fig. 3.** (Left) Graphical representation of the history-situation pattern; (Right) Instance of the history-situation pattern; Squares represent ontology classes and unidirectional arrows predicates enhanced by cardinality constraints.

Within SemanticHealthNet, we have elaborated two representations of semantic patterns: an OWL 2 DL and a RDF [23] representation. The OWL-based representation describes a pattern as a set of logical axioms. **Table 1** shows the OWL rendering of the history-situation pattern as pieces of information (shn:InformationItem) that are acquired by performing some clinical process (shn:ClinicalProcess) and that refer to clinical situations (shn:ClinicalSituation) of a given type (if any), which happened in the past (sct:InThePast). Additionally, it allows expressing epistemic information aspects (shn:InformationAttribute) that indirectly refer to the situation (e.g. severe, present, etc.).

| |
|---|
| *shn:InformationItem*<br>and **shn:isAboutSituation** only *shn:ClinicalSituation*<br>and **btl:isOutcomeOf** some *shn:ClinicalProcess*<br>and **shn:hasInformationAttribute** some *shn:InformationAttribute*<br>and **shn:hasInformationAttribute** some *sct:InThePast*<br>and **shn:hasInformationAttribute** some *sct:FindingContextValue* |

**Table 1.** OWL DL representation of history-situation pattern; Terms using *italics* represent ontology classes, **bold face** represents ontology object properties.

**Table 2** shows the RDF representation, which consists of a set of Subject-Predicate-Object (SPO) triples. Both representations are connected as follows: The subject and object parts of a triple correspond to ontology classes, and the predicates to ontology expressions. **Table 3** provides the OWL DL translation of the RDF predicates. This allows the implementation of automatic translations from a 'closer to user' RDF representation into a representation in OWL DL, which would require a more in-depth understanding of DL syntax and semantics. In the following we will describe the use

of semantic patterns regarding EHR clinical models and ontology-based terminologies as SNOMED CT.

| |
|---|
| *shn:InformationItem* ´describes situation´ *shn:ClinicalSituation* |
| *shn:InformationItem* ´results from process´ *shn:ClinicalProcess* |
| *shn:InformationItem* ´has attribute´ *shn:InformationAttribute* |
| *shn:InformationItem* ´has temporal context´ *sct:InthePast* |
| *shn:InformationItem* ´has situation context´ *sct:FindingContextValue* |

**Table 2.** Subject-Predicate-Object (SPO) triple representation; Italic terms represent ontology classes and terms in quotes represent predicates. Note that predicates are not equivalent to OWL object properties.

| Predicate | OWL DL expression |
|---|---|
| 'describes situation' | SUBJ subClassOf **shn:isAboutSituation** only OBJ |
| 'results from process' | SUBJ subClassOf **btl:isOutcomeOf** some OBJ |
| 'has attribute' | SUBJ subClassOf **shn:hasInformationAttribute** some OBJ |
| 'has temporal context' | SUBJ subClassOf **shn:hasInformationAttribute** some OBJ |
| ´has situation context´ | SUBJ subClassOf **shn:hasInformationAttribute** some OBJ |

**Table 3.** Translations of RDF predicates into OWL DL axioms within the shared ontology framework

### 3.2 The role of semantic patterns regarding EHR clinical models and medical domain ontology-based terminologies

Assuming that a limited set of top-level semantic patterns that can be specialized and composed is sufficient to represent a great variety of clinical information, we propose the use of semantic patterns as proxy to the semantic representation of clinical information encoded by EHR structured clinical models and ontology-based medical terminologies. They act as a template, with fix and variable parts, and guide the mapping process in which the correspondences between information model structures and their values are defined with regards to the ontology. Dashed arrows in Fig. 4 indicate the correspondences between the clinical model from Fig. 1 and the history-situation pattern.

As observed, the pattern is applied to both, the SNOMED CT term used as binding and the clinical model information structures. Three correspondences have been provided. Two between the *CLUSTER[at0002]* binding and the pattern triples that represent the situation and its temporal context. Diabetes mellitus is placed as subclass of *shn:ClinicalSituation*. One between the value of *ELEMENT [at0003]* and the pattern triple that represents if the situation is present (*sct:KnownPresent*) or absent (*sct:KnownAbsent*). Both are represented as subclasses of *sct:FindingContextValue,* and will be selected depending of the value of the model instance (True or False).

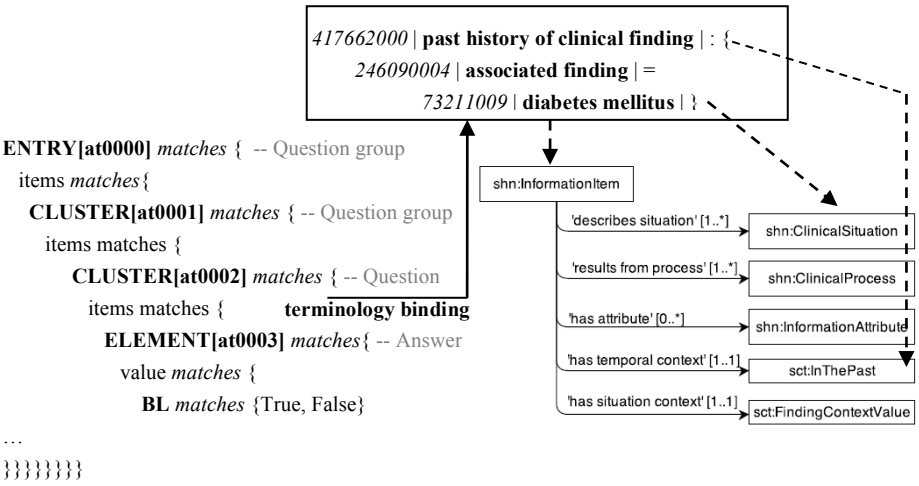


**Fig. 4.** (Left) ISO 13606 archetype and SNOMED CT binding to record the question "past history of diabetes" (Y/N); (Right) Graphical representation of the history-situation pattern

## 4 Results

In the following we will describe the potential of semantic patterns for each of the tasks introduced in the Methods section. We will use the history-situation pattern as example.

### 4.1 Semantic patterns provide interoperability across heterogeneously represented data

We will use the history-situation pattern to provide semantic interoperability across two past history data instances captured by two heterogeneous fictitious applications used at a GP consultation and at a hospital. **Fig. 5** shows their interfaces. They have been designed attending to different requirements and therefore record the information at different levels of detail. At the hospital (right), the specialist records additional information about the patient past situation (i.e. cause and severity). However, the GP only records the situation itself (left).

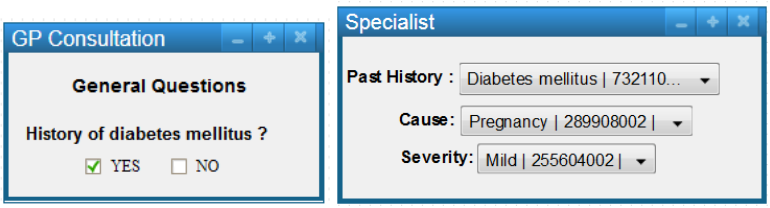


**Fig. 5.** (Left) Past history recording at the GP; (Right) Past history recording at the specialist.

Each of the above applications is based on a different ISO 13606 clinical model. The GP application is based on the questionnaire model introduced in Section 2.1. The left part of **Fig. 6** shows the model used by the hospital application. Both are different in terms of structure but not syntax, since both implement the same standard.

In order to access information recorded by both applications, independently of their source representation, the correspondences between each clinical model and the history-situation pattern are defined. **Fig. 4** depicted the correspondences between the questionnaire model and the pattern. Following, dashed arrows in **Fig. 6** show the correspondences for the hospital model. This model allows recording the severity of the past disease and its cause, requiring the use of the situation pattern, by composition. The situation pattern, allows providing more detail information such as when it occurs, where, associated situations, etc.

Once the correspondences between the models and the patterns are established, when the former ones are instantiated with patient data, the instances of the patterns are also created, in a similar way to the one shown in **Fig. 3**. If OWL DL instances are created, it is possible to perform homogeneous queries on instances from both applications and retrieve their results [24].

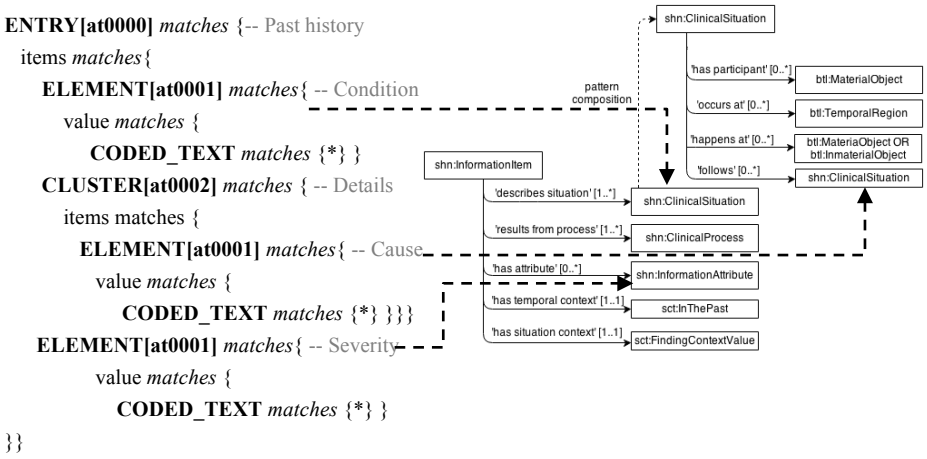


**Fig. 6.** ISO 13606 clinical model that records past history of condition, its cause and severity

Besides, the use of the ontology framework and DL reasoning allows performing queries at different granularity level: E.g. "Information about all patients with past history of some endocrine disease", without specifying whether diabetes or a different one.

### 4.2 Semantic patterns guide the creation of clinical models and detect semantic inconsistencies

Semantic patterns can guide the development of new clinical models if the latter are created by following the constraints dictated by a set of limited top-level patterns.

Top-level patterns are based on a set of generic ontology classes and predicates that can be specialized and composed by following the ontology constraints. These constraints can be used to determine which elements include in a clinical model or in a terminology binding.

As a difference with clinical models, where their elements are only structurally related (e.g. list, tree, etc.), within patterns they are connected by semantic relationships (e.g. *shn:isAboutSituation*, *btl:isOutcomeOf*, etc.). These relationships can be used to guide the decision of the elements to include in a model, reducing the existing arbitrariness. Now this is mainly a non-constrained modeller decision that might lead to the creation of non-interoperable models even for the same use case.

If semantic patterns are not applied at clinical models design time, they still can be used to detect semantic inconsistencies across them. As an example, **Fig. 7** shows an excerpt of a CIMI model that records observation results. It records: (i) what is observed, *ELEMENT[at0001]* (e.g. color of the eye); (ii) the reason to perform the observation, *ITEM[at0002]* (e.g. problem wearing contact lens); (iii) the method used to observe, *ITEM[at0003]* (e.g. eye examination); (iv) the status of the observation, *ELEMENT[at0004]* (e.g. performed, planned); and (v) the priority to perform the observation, *ELEMENT[at0005]* (e.g. high, normal).

```
CLUSTER[at0000] matches { -- Observable
  item matches {
    ELEMENT[at0001] occurrences matches {1} matches { -- Name
      value matches { TEXT matches {*}}}
    ITEM[at0002] occurrences matches {0..*} -- Reason
    ITEM[at0003] occurrences matches {0..*} -- Method
    ELEMENT[at0004] occurrences matches {0..1} matches { -- Status
      value matches { CODED_TEXT matches {*}}}
    ELEMENT[at0005] occurrences matches {0..1} matches { -- Priority
      value matches { TEXT matches {*}}}
  }}
```

**Fig. 7.** Excerpt of the CIMI model (CIMI-CORE-CLUSTER.observable.v1.0.0) to record observation results

**Fig. 8** shows another CIMI model that records observation requests and references the above model by composition (keyword "use_archetype"). Besides, it also references a model to record observation actions. Within this last model we have found a content overlapping with the observation result one, since it also provides elements for recording the reason, method, status and priority of the observation.

```
ENTRY[at0000.1] matches { -- Observation
  link matches {LINK[at0.1] occurrences matches {0..*} -- Associated request}
  data matches {
      use_archetype CLUSTER [CIMI-CORE-CLUSTER.observable.v1] -- Observable
      use_archetype CLUSTER [CIMI-CORE-CLUSTER.finding.v1] -- Results
      use_archetype CLUSTER [CIMI-CORE-CLUSTER.observe_action.v1] -- Observe action
      …
```

**Fig. 8.** Excerpt of the CIMI model (CIMI-CORE-ENTRY.observation.v4.0.0) to record an observation request and its result

Semantic patterns could avoid such an overlapping situation, by providing formal modelling guidelines, based on the ontological framework, to distinguish across what is observed, the observation procedure and the result of the observation.

Additionally, as already mentioned, they can help to guide or detect inconsistencies regarding terminology bindings. For instance, the pattern logic axiom (*shn:InformationItem* and **shn:isAboutSituation** only *shn:ClinicalSituation*), relates an information entity (i.e. *shn:InformationItem*) with a clinical entity (*shn:ClinicalSituation)* and the latter is equivalent to SNOMED CT *clinical findings*. Therefore, if a model information structure is mapped to that axiom, its value is only valid if it is of the type *clinical finding*.

When clinical models are instantiated with patient data, semantic patterns can also be used to check that the data entered complies with the constraints defined at the model level.

## 5 Discussion and conclusions

In this work we have proposed semantic patterns as ontology design content patterns applied to the representation of clinical information. They were motivated by our experiences of representing clinical information using OWL DL, which often resulted in highly complex expressions.

The EHR standards community has put a lot of effort in providing standardized means to represent the EHR. However, the complexity of the medical domain and their heterogeneous data capture and re-use needs does not make it easy. One of the reasons might be the high degree of freedom provided when modelling clinical information, which is mainly formally constrained in terms of structure but without considering the meaning of what is being represented.

Aware of this gap, and concerned about the need of providing standardized modelling means, we propose an ontological framework, in order to represent both information and medical entities, constrained by a top-level ontology which reduces arbitrariness in ontology design. Semantic patterns are based on this framework and therefore constrained by their concepts and relations. In [25], the advantages of using a top-level ontology for creating ontology design content patterns were described, stating that it provides it with an existing backbone structure and well-defined relations.

Semantic patterns provide a more intuitive representation and standardize their development process, yet allowing flexibility through specialization and composition. We have proposed their representation in OWL DL and in RDF. The former one allows logical reasoning and therefore more advanced exploitation of information, although it might be more difficult to implement in a real system, due to performance issues. In the latter case, the RDF representation although less expressive and therefore more limited in terms of information exploitation, might be more adequate. Correspondences between both representations exist, what might allow using the most suitable one for each use case.

In this work we have demonstrated how semantic patterns can be applied to EHR clinical models and ontology-based terminologies (1) to provide semantic interopera-

bility across heterogeneously represented data and (2) commented their potential use to guide the creation of clinical models and detect semantic inconsistencies across them.

By looking at the content patterns available at the NeOn repository [26], we did not find specific patterns for the modelling of clinical information. However, patterns such as the agent-role or the action ones can be applied.

There are numerous new issues that arise from the use of semantic patterns for EHR modelling that still have to be investigated. These include the selection of the right set of patterns to be used for modelling specific pieces of clinical information, who would create and maintain the patterns and who would manage and validate them.

Other issues must be further investigated, such as providing evidence that a set of top-level semantic patterns for modelling clinical information can be rather small, with increasing complexity and expressiveness coming from specialization and composition. So far we have only worked with limited modelling examples and we need more evidence of the real benefit of using patterns; what is hard to obtain without appropriate tools that implement them.

Further research should include the potential of semantic patterns for detecting semantic inconsistencies across existing clinical models, considering their specialization, composition and cardinality constraints. Languages such as SPIN [27] or RDF shapes [28] could be helpful for their representation and are subject of our research.

**Acknowledgements.** This work has been funded by the SemanticHealthNet Network of Excellence within the EU 7th Framework Program, Call:FP7-ICT- 2011-7, agreement 288408. http://www.semantichealthnet.eu/

## References

1. Stroetmann, V.N., Kalra, D., Lewalle, P., et al: Interoperability for better health and safer healthcare. Deployment and research roadmap for Europe. (2009); http://www.empirica.com/publikationen/documents/2009/semantic-health-report.pdf
2. International Health Terminology Standards Development Organisation (IHTSDO). http://www.ihtsdo.org/ (accessed August 2014).
3. HL7 TermInfo Project Wiki. Guidance on Overlap between RIM and SNOMED CT Semantics. http://wiki.hl7.org/index.php?title=TermInfo_Project# (accessed August 2014).
4. Clinical Information Modeling Initiative (CIMI). http://informatics.mayo.edu/CIMI/index.php/Main_Page (accessed August 2014)
5. Semantic Interoperability for Health Network. http://www.semantichealthnet.eu/ (accessed August 2014).
6. Rector, A., Qamar, R., Marley, T.: Binding Ontologies & Coding systems to Electronic Health Records and Messages, (2009) Appl Ontol 2009;4:51-69.
7. Quine, W.V.: On what there is. Quintessence-Basic Readings from the Philosophy of W.V.Quine. Belknap Press, Cambridge 2004; Gibson, R. (ed.).
8. Bodenreider, O., Smith, B., Burgun, A.: The Ontology-Epistemology Divide: A Case Study in Medical Terminology. Third International Conference on Formal Ontology in Information Systems (FOIS 2004). IOS Press; 2004:185–95.

9. Beale, T. Archetypes: Constraint-based domain models for future-proof information systems. Eleventh OOPSLA Workshop on Behavioral Semantics: Serving the Customer. Seattle, Washington, USA: Northeastern University; 2002:16-32.
10. ISO EN13606 Electronic Health Record Communication Part 2: Archetype interchange specification. CEN TC/251, 2008
11. Dolin, R.H., Alschuler, L., Boyer, S., et al.: The HL7 Clinical Document Architecture, release 2. J Am Med Inform Assoc 2006;13:30-39
12. Schulz, S., Jansen, L.: Formal ontologies in biomedical knowledge representation. Yearbook of Medical Informatics 2013;8(1):132-46
13. Cimino, J.J., Zhu, X.: The practical impact of ontologies on biomedical informatics. Yearb Med Inform [Internet] .2006;124–35. http://www.ncbi.nlm.nih.gov/pubmed/17051306
14. Schulz, S., Boeker, M.: BioTopLite: An Upper Level Ontology for the Life Sciences. Evolution, Design and Application. *Informatik 2013*. U. Furbach, S. Staab; editors(s). IOS Press; 2013
15. Schulz, S., Rector, A., Rodrigues, J.M., Spackman, K.: Competing interpretations of disorder codes in SNOMED CT and ICD. AMIA Annu Symp Proc. 2012;2012:819-27. Epub 2012 Nov 3. PubMed PMID: 23304356; PubMed Central PMCID: PMC3540515.
16. Martínez-Costa, C., Schulz, S.: Ontology-based reinterpretation of the SNOMED CT context model. Proceedings of the 4th International Conference on Biomedical Ontology. CEUR Workshop Proceedings 2013; 1040:90-95.
17. W3C OWL working group. OWL 2 Web Ontology Language, Document Overview. W3C Recommendation 11 December 2012. http: //www.w3.org/TR/owl2-overview (accessed August 2014).
18. Schulz, S., Martínez-Costa, C.: How Ontologies Can Improve Semantic Interoperability in Health Care. In: Riaño, D; Lenz, R; Miksch, S; Peleg, M; Reichert, M; Teije, A editors(s). Lecture Notes in Computer Science. 8268: Berlin Heidelberg: Springer International Publishing; 2013;1-10
19. Presutti, V., Gangemi, A.: Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies. In Proceedings of the 27th International Conference on Conceptual Modeling *(ER '08),* Springer-Verlag 2008, Berlin, Heidelberg, 128-141.
20. Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In Proceedings of the Fourth International Semantic Web Conference, 2005:262-276
21. Gangemi, A., Presutti, V.: Ontology Design Patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, Second edition, pp. 221 – 243, Springer (2009)
22. Baader, F., Calvanese, D., McGuinness, D.L., et al. The Description Logic Handbook, Cambridge University Press, New York, NY; 2007
23. W3C Resource Description Framework (RDF). http://www.w3.org/RDF/ (accessed August 2014).
24. Martínez-Costa, C., Schulz, S.: Ontology Content Patterns as Bridge for the Semantic Representation of Clinical Informatics. Applied Clinical Informatics eHealth special issue. 2014; 5(3): 660-669
25. Seddig-Raufie, D., Jansen, L., Schober, D., Boeker, M., Grewe, N., Schulz, S. Proposed actions are no actions: re-modeling an ontology design pattern with a realist top-level ontology. J Biomed Semantics. 2012;5(Suppl 2):S2.
26. NeOn repository Ontology Design Patterns.org (ODP) http://ontologydesignpatterns.org (accessed August 2014).
27. SPARQL Inference Notation (SPIN) http://spinrdf.org/ (accessed August 2014).
28. Shape Expressions 1.0 Definition. http://www.w3.org/Submission/2014/SUBM-shex-defn-20140602/ (accessed August 2014).

# An Ontology Design Pattern for Material Transformation

Charles Vardeman[1], Adila A. Krisnadhi[2,3], Michelle Cheatham[2], Krzysztof Janowicz[4], Holly Ferguson[1], Pascal Hitzler[2], Aimee P. C. Buccellato[1], Krishnaprasad Thirunarayan[2], Gary Berg-Cross[5], and Torsten Hahmann[6]

[1] University of Notre Dame,
[2] Wright State University,
[3] University of Indonesia
[4] University of California, Santa Barbara
[5] Spatial Ontology Community of Practice (SOCOP), USA
[6] University of Maine

**Abstract.** In this work we discuss an ontology design pattern for material transformations. It models the relation between products, resources, and catalysts in the transformation process. Our axiomatization goes beyond a mere surface semantics. While we focus on the construction domain, the pattern can also be applied to chemistry and other domains.

## 1 Introduction & Motivation

According to the United Nations, the construction industry and related support industries are leading consumers of natural resources. Consumption of these natural resources result in the emission of energy, and thus carbon and other greenhouse gases, which are then"embodied" in the consumption process. Efforts have been made to quantify these emissions through measures of embodied energy, carbon and water but are lacking due to poor quality of data sources, lack of understanding of uncertainty in the data, lack of geospatial attributes necessary for proper calculation of embodied properties, understanding regional and international variation in data, incompleteness of secondary data sources and variation in manufacturing technology that lead to significant variation calculated values [2]. One methodology for quantification of embodied energy is through input-output life cycle analysis utilizing process data that compile a life cycle inventory of a construction product. By analyzing a "cradle to grave" path of individual building components, the embodied energy sequestered in all building materials during all processes of construction, in on-site construction and final demolition and disposal of a buildings constituent components gives a measure of total embodied energy for a given structure. Sources of embodied energy include the amount of the energy consumed in construction, prefabrication, assembly, transportation of materials to a building structure, initial manufacturing building materials, in renovation and refurbishment of the structure through it's lifetime [1]. The Green Scale Project[1] is studying the feasibility of creating a

[1] http://www.greenscale.org

geospatial-temporal knowledge base (KB) which facilitates mapping of national energy and fuel production to individual construction site localities and construction material manufacture localities as linked open data. Such a knowledge base would facilitate the calculation of embodied energy for a given construction component as a query of the embodied energy required for manufacture and transportation of it's constituent parts. This KB will use ontology design patterns to formally describe the transportation and transformation processes.

Transportation of a manufacturing component from location to location and the energies associated with that transportation can be modeled via the Semantic Trajectory pattern (STODP) [3]. The remaining contribution to the total embodied energy is the energy required for transformation or assembly of one or more components into the desired manufactured artifact.

In this work we discuss the development of a Material Transformation pattern[2] to contextualize this transformation process from raw components and the required equipment to a final manufactured artifact. Chaining this pattern with STODP will facilitate understanding of a complete manufacturing process from raw material extraction to assembly of all components needed for that product. The presented work was done in two 2-day sessions involving domain experts from architecture, computational chemistry, and geography, as well as ontology engineers at GeoVoCampDC2013[3] and GeoVoCampWI2014[4]. We present a full axiomatization that goes beyond mere surface semantics [4] (e.g., a simple type hierarchy). During the development, several competency questions that a domain expert may ask were discussed. These include:

- *"What material resources were required to produce a product?"*
- *"Where did the transformation take place?"*
- *"What was the time necessary for the transformation?"*
- *"What materials or conditions were necessary for the transformation to occur?"*

## 2 Material Transformation Pattern

The Material Transformation pattern is visualized in Fig. 1, including the extension with entities relevant for representing energy information, which are green-colored and use dashed line. For formalization, we use the Description Logic (DL) notation, which can easily be encoded using syntax of the OWL 2. The core part of the pattern is intended to describe change(s) that occur between the input material of the transformation and its output. In this core part, the `MaterialTransformation` class represents concrete instances of material transformation. We distinguish inputs of a material transformation into `Resource`, which represents types of material that may undergo a change (into a different type of material) in the transformation, and `Catalyst`, which represents types of material needed by the transformation, but remain unchanged by it. A `MaterialTransformation` has some `Resource` as input (1), and some `Product`, which is also some type of material, as output (2). Axiom (5) asserts

[2] http://ontologydesignpatterns.org/wiki/Submissions:Material_Transformation
[3] http://vocamp.org/wiki/GeoVoCampDC2013
[4] http://www.ssec.wisc.edu/meetings/geosp_sem/

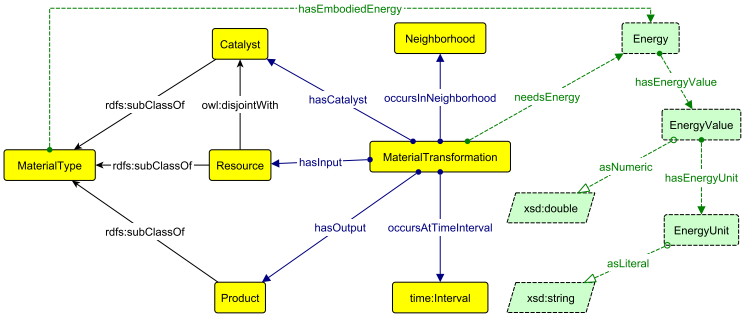


**Fig. 1.** Material Transformation Pattern with Energy Information

that every `Resource`, `Catalyst` and `Product` is some `MaterialType`, while (6) and distinguishes `Resource` from `Catalyst`. Axiom (3) and (4) assert that a `MaterialTransformation` occurs in a spatial `Neighborhood`[5] and a time interval, modeled using the `Interval` class from the W3C's OWL Time ontology[6].

$$\texttt{MaterialTransformation} \sqsubseteq \exists \texttt{hasInput}.\texttt{Resource} \tag{1}$$

$$\texttt{MaterialTransformation} \sqsubseteq \exists \texttt{hasOutput}.\texttt{Product} \tag{2}$$

$$\texttt{MaterialTransformation} \sqsubseteq \exists \texttt{occursInNeighborhood}.\texttt{Neighborhood} \tag{3}$$

$$\texttt{MaterialTransformation} \sqsubseteq \exists \texttt{occursAtTimeInterval}.\texttt{time:Interval} \tag{4}$$

$$\texttt{Resource} \sqcup \texttt{Catalyst} \sqcup \texttt{Product} \sqsubseteq \texttt{MaterialType} \tag{5}$$

$$\texttt{Resource} \sqcap \texttt{Catalyst} \sqsubseteq \bot \tag{6}$$

We express changes occurring within a material transformation, using first-order logic, that it has an input that is not part of the output (7); and an output that is not part of the input, in a formula analogous to (7).

$$\forall x(\texttt{MaterialTransformation}(x) \rightarrow \exists y(\texttt{hasInput}(x,y) \wedge \neg \texttt{hasOutput}(x,y))) \tag{7}$$

These formulas, however, cannot be expressed in the OWL framework, but there are extensions of DL that can express them. For example, using boolean constructors on properties [5], axiom (7) is expressed in DL as:

$$\texttt{MaterialTransformation} \sqsubseteq \exists(\texttt{hasInput} \sqcap \neg \texttt{hasOutput}).\top$$

Meanwhile, for the remaining properties of the core part of the pattern, we assert the guarded domain and range restrictionsas exemplified for the `hasInput` property in (8) and (9) below. Such guarded restrictions are preferable over the unguarded versions (i.e., of the form $\mathsf{dom}(P) \sqsubseteq A$ and $\mathsf{range}(P) \sqsubseteq B$) as they introduce weaker ontological commitments and thus foster reuse.

$$\exists \texttt{hasInput}.\texttt{Resource} \sqsubseteq \texttt{MaterialTransformation} \tag{8}$$

$$\texttt{MaterialTransformation} \sqsubseteq \forall \texttt{hasInput}.\texttt{Resource} \tag{9}$$

[5] Neighborhood provides a toplogical definition for specifing nearness. This could be specified in different ways such as using positional coordinates, a bounded area on a map, or a named region such as a place, city or factory.

[6] http://www.w3.org/TR/owl-time/

For the scenario where we need to calculate the embodied energy in the output of a material transformation, we can extend the pattern with additional energy information as depicted in Fig. 1. In the axiomatization, we then assert that a `MaterialTransformation` needs some `Energy` (10), while each material type has some embodied energy (11). Energy itself is abstracted as an instance of the `Energy` class, which has some energy value and unit.

$$\texttt{MaterialTransformation} \sqsubseteq \exists \texttt{needsEnergy}.\texttt{Energy} \tag{10}$$

$$\texttt{MaterialType} \sqsubseteq \exists \texttt{hasEmbodiedEnergy}.\texttt{Energy} \tag{11}$$

$$\texttt{Energy} \sqsubseteq \exists \texttt{hasEnergyValue}.\texttt{EnergyValue} \tag{12}$$

$$\texttt{EnergyValue} \sqsubseteq \exists \texttt{hasEnergyUnit}.\texttt{EnergyUnit} \sqcap \exists \texttt{asNumeric}.\texttt{xsd:double} \tag{13}$$

$$\texttt{EnergyUnit} \sqsubseteq \exists \texttt{asLiteral}.\texttt{xsd:string} \tag{14}$$

Embodied energy in the output as a result of a material transformation can be calculated by aggregating embodied energy of the input and catalyst, together with energy requirement of the material transformation itself. This cannot be done within OWL, but is relatively straightforward to implement in the application as all the necessary information are easily retrievable from the populated pattern. Furthermore, if the application allows updates on the data populating the pattern, we can chain two instantiations of this pattern and include STODP.

## 3 Conclusion and Future Work

Although it is beyond the scope of the present work, the Material Transformation pattern should be sufficiently generic to describe other types of transformation processes ranging from chemical reactions to creation-annihilation events in high energy physics. We believe the pattern to be of general use to broader product life cycle inventories outside the construction domain.

*Acknowledgements.* We are grateful for the inputs from Lamar Henderson, Deborah MachPherson, Laura Bartolo, and Damian Gessler to improve the pattern. Vardeman, Buccellato and Ferguson would like to acknowledge funding from the University of Notre Dame's Center for Sustainable Energy, School of Architecture, College of Arts and Letters and Center for Research Computing in support of this work. Gary Berg-Cross acknowledges funding from the NSF grant 0955816, INTEROP-Spatial Ontology Community of Practice. Vardeman would like to acknowledge funding from NSF grant PHY-1247316 "DASPOS: Data and Software Preservation for Open Science." Adila Krisnadhi, Michelle Cheatham, and Pascal Hitzler acknowledge support by the National Science Foundation under award 1017225 "III: Small: TROn – Tractable Reasoning with Ontologies."

## References


1. Dixit, M.K., Fernández-Solís, J.L., Lavy, S., Culp, C.H.: Need for an embodied energy measurement protocol for buildings: A review paper. Renewable and Sustainable Energy Reviews 16(6), 3730–3743 (2012)
2. Dixit, M.K., Fernández-Solís, J.L., Lavy, S., Culp, C.H.: Identification of parameters for embodied energy measurement: A literature review. Energy and Buildings 42(8), 1238–1247 (2010)
3. Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., Kolas, D.: A geo-ontology design pattern for semantic trajectories. In: Spatial Information Theory, pp. 438–456. Springer (2013)
4. Janowicz, K., Hitzler, P.: Thoughts on the complex relation between linked data, semantic annotations, and ontologies. In: 6th international workshop on Exploiting semantic annotations in information retrieval. pp. 41–44. ACM (2013)
5. Rudolph, S., Krötzsch, M., Hitzler, P.: Cheap boolean role constructors for description logics. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) Logics in Artificial Intelligence, 11th European Conference, JELIA 2008, Proceedings. Lecture Notes in Computer Science, vol. 5293, pp. 362–374. Springer (2008)

# An Ontology Design Pattern for Activity Reasoning

Amin Abdalla[1], Yingjie Hu[2], David Carral[3], Naicong Li[4], Krzysztof Janowicz[2]

[1] Institute for Geoinformatics,Vienna University of Technology, Austria
[2] Department of Geography, University of California Santa Barbara, USA
[3] Kno.e.sis Center, Wright State University, USA
[4] University of Redlands, USA

**Abstract.** Activity is an important concept in many fields, and a number of activity-related ontologies have been developed. While suitable for their designated use cases, these ontologies cannot be easily generalized to other applications. This paper aims at providing a generic ontology design pattern to model the common core of activities in different domains. Such a pattern can be used as a building block to construct more specific activity ontologies.

## 1 Introduction

*Activity* is an important research topic in many fields, such as artificial intelligence, human geography, transportation research, psychology, and human-computer interaction. As a result, there are a number of conceptual models that attempt to capture the semantics of activities. Existing activity ontologies (e.g.,[5] and [3]), however, are often designed for specific use cases and cannot be easily generalized to applications in other domains. This makes reuse difficult and raises the question whether there is a common, domain-independent core.

Two main perspectives on activity modeling can be identified from the literature: a spatiotemporal-centric and a workflow-centric perspective. The first one treats activities as a set of temporally-ordered entities in space and time. This perspective has often been found in the literature on *time geography* [8], which attempts to capture human activities in the form of spatiotemporal constraints. This perspective has been translated into software systems capable of computing and analyzing spatial and temporal activity properties. However, this perspective does not consider the logical relations between activities, such as *dependency* or *component* relations.

The second perspective treats activities as a workflow. This view is often found in planning-related applications, in which *preconditions* and *effects* of activities are important. Representative examples include the Planning Domain Definition Language (PDDL), or the Process Specification Language (PSL-core) [7]. Some patterns (e.g., Action ODP, Planning ODP, and Event ODP) accessible via the ODP portal[5], as well as the TOVE (Toronto Virtual Enterprise) ontology [5], also share this workflow-centric perspective, with an emphasis on activities that consume or occupy limited resources.

This work aims at developing a more generic ontology design pattern (ODP) that incorporates parts of both perspectives. Such a generic ODP can be employed as a building block or strategy for designing more specific activity ontologies. While the PROV ontology[6] also models activities and the associated entities, it focuses on recording the changes of entities and the representation of provenance information. Given the

[5] http://ontologydesignpatterns.org
[6] http://www.w3.org/TR/prov-o/

fast development of ubiquitous sensor networks and the Internet of Things, more data about human activities are becoming available. These rich amount of data enable new applications, such as activity-based personal information management [1] and human trajectory modeling [9]. Thus, a generic activity ODP can help semantically annotate human activity data, thereby facilitating information retrieval as well as automatic reasoning.

Deriving an ontology design pattern requires a generic use case which can capture the recurring problems in different application domains [6]. *Competency questions* have been recognized as a good approach to detect and generalize the modeling requirements from multiple domains. They are queries that a domain expert would be expected to run against a knowledge base. For the proposed activity ODP, such competency questions include:

- **Question 1**: *"What are the requirements (or outcomes) of an activity?"*
- **Question 2**: *"What is the place (or deadline) of an activity?"*
- **Question 3**: *"What activities need to be completed first in order to start this activity?"*
- **Question 4**: *"What are the other activities which can be started after this activity?"*
- **Question 5**: *"What are the activities supported by this place?"*
- **Question 6**: *"What activities happen before (or in parallel, or after) this activity?"*

## 2 Pattern Description and Formalization

This section presents the activity pattern by discussing the more interesting classes, properties, and axioms. Description Logics (DL) notation has been used to present the axioms. To encode the pattern, we use the logic fragment $DLP_{\exists}$ as defined in [2], which allows for polynomial time reasoning. The proposed activity ODP has also been formally encoded using the Web Ontology Language (OWL). It is available at `http://descartes-core.org/ontologies/activities/1.0/ActivityPattern.owl` . A schematic view of the pattern is shown in Figure 1 .

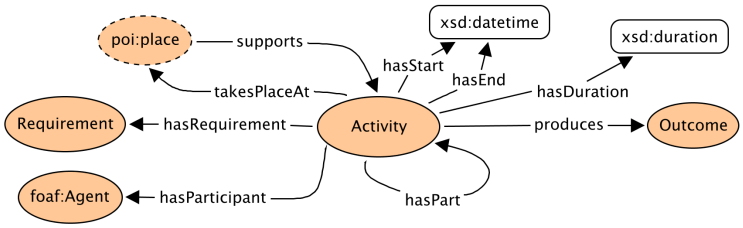


**Fig. 1.** A schematic view of the Activity ODP.

**Activity:** In accordance with PSL, our pattern allows activities to potentially consist of several component activities (which can yet again be associated with further component activities). In this way, aggregation over a set of activities into higher level activities is possible. We make use of the properties hasPart and isPartOf to formally denote this relation. These two roles, which are inverse roles with respect to each other, are declared both transitive and reflexive. Also, the Activity class is declared as disjoint with the classes of Requirement and Outcome.

We make use of the following axioms to enforce these characteristics [7]

$$\mathsf{hasPart}^{-} \equiv \mathsf{isPartOf} \tag{1}$$

$$\mathsf{hasPart} \circ \mathsf{hasPart} \sqsubseteq \mathsf{hasPart} \tag{2}$$

$$\top \sqsubseteq \exists \mathsf{hasPart.Self} \tag{3}$$

**Requirements and Outcomes:** Dependency relations are important to model multiple activities. To capture these relations, we make use of Requirements and Outcomes, i.e., the required inputs and resulting outputs of any given activity. In some cases, the outcome of one activity might be a requirement of another. If this is the case, we say the former activity precedes the latter, assuming that an outcome is only produced after an activity was finished. Thus precedes does depict a logical relation that requires temporal precedence. We define the properties precedes and isPrecededBy as inverse roles, and declare them as transitive and irreflexive.

$$\mathsf{hasOutcome} \circ \mathsf{isRequirement} \sqsubseteq \mathsf{precedes} \tag{4}$$

**Agent:** The class of foaf:Agent from the FOAF ontology[8] has been employed to represent an actor or an autonomous agent whose behavior is intentional. The foaf:Agent class can also be substituted by its sub classes, such as foaf:Group or foaf:Person, and therefore allows ontology engineers to further specify what type of participant is involved in the activity. The hasParticipant property depicts the involvement of an foaf:Agent in an activity.

**Spatiotemporal Relations:** The spatiotemporal information associated to activities is captured through the following properties.

- **takesPlaceAt.** This property indicates the place where an activity takes place. It can be used as a hook to align to other ODPs, e.g., the POI pattern.
- **hasStart.** This property indicates the time an activity starts.
- **hasEnd.** This property indicates the time an activity ends.
- **hasDuration.** This property indicates the time period that an activity lasts. The value of duration should be equal to the difference between the start and end time of an activity.

It is worth to note that the above spatiotemporal properties can be used to represent not only past activities (i.e., activities that have already happened) but also future activities (i.e., activities scheduled in the future).

The proposed activity ODP also distinguish two types of activities, namely **Fixed Activity** and **Flexible Activity**, as defined in the time geography literature [8,4]. These two types of activities can often be found in our daily life. *Fixed activities* refer to the activities that must be completed at a particular point in space and time (e.g., attending a meeting at the conference room at 3:30 pm). *Flexible activities* are activities which can be completed at a time and space range. For example, buying grocery after work is a flexible activity since it can be completed at any time after work and in different

[7] The full axiomatization is not presented here due to lack of space. However, a complete OWL version is available online at Descartes-Core.

[8] http://xmlns.com/foaf/0.1/

stores. We define the following axioms to formally encode and automatically classify these two types of activities.

$$\exists \mathsf{hasStart}.\top \sqcap \exists \mathsf{hasEnd}.\top \sqsubseteq \mathsf{FixedActivity} \quad (5)$$

$$\exists \mathsf{hasStart}.\top \sqcap \exists \mathsf{hasDuration}.\top \sqsubseteq \mathsf{FixedActivity} \quad (6)$$

$$\exists \mathsf{hasEnd}.\top \sqcap \exists \mathsf{hasDuration}.\top \sqsubseteq \mathsf{FixedActivity} \quad (7)$$

$$\mathsf{FlexibleActivity} \sqcap \exists \mathsf{hasStart}.\top \sqcap \exists \mathsf{hasEnd}.\top \sqsubseteq \bot \quad (8)$$

$$\mathsf{FlexibleActivity} \sqcap \exists \mathsf{hasStart}.\top \sqcap \exists \mathsf{hasDuration}.\top \sqsubseteq \bot \quad (9)$$

$$\mathsf{FlexibleActivity} \sqcap \exists \mathsf{hasEnd}.\top \sqcap \exists \mathsf{hasDuration}.\top \sqsubseteq \bot \quad (10)$$

## 3 Conclusions

This paper proposed a generic ODP to capture the common core of activities in different domains. Specifically, it incorporates two perspectives towards activity modeling, namely the spatiotemporal perspective and the workflow perspective, which can often be found in existing work. Such a pattern can be used as a building block to design more domain specific ontologies.

## Acknowledgement

This work is supported by the NSF under award 1017255 and "La Caixa" Foundation.

## References

1. Abdalla, A., Weiser, P., Frank, A.U.: Design principles for spatio-temporally enabled pim tools: A qualitative analysis of trip planning. In: Geographic Information Science at the Heart of Europe, pp. 323–336. Springer (2013)
2. Carral, D., Scheider, S., Janowicz, K., Vardeman, C., Krisnadhi, A.A., Hitzler, P.: An ontology design pattern for cartographic map scaling. In: The Semantic Web: Semantics and Big Data, pp. 76–93. Springer (2013)
3. Catarci, T., Habegger, B., Poggi, A., Dix, A., Ioannidis, Y., Katifori, A., Lepouras, G.: Intelligent user task oriented systems. In: Proceedings of the Second SIGIR Workshop on Personal Information Management (PIM) (2006)
4. Chen, X., Kwan, M.P.: Choice set formation with multiple flexible activities under space–time constraints. International Journal of Geographical Information Science 26(5), 941–961 (2012)
5. Fox, M.S., Chionglo, J.F., Fadel, F.G.: A common-sense model of the enterprise. In: Proceedings of the 2nd Industrial Engineering Research Conference. vol. 1, pp. 425–429 (1993)
6. Gangemi, A.: Ontology design patterns for semantic web content. In: The Semantic Web–ISWC 2005, pp. 262–276. Springer (2005)
7. Gruninger, M., Menzel, C.: The process specification language (psl) theory and applications. AI magazine 24(3), 63 (2003)
8. Hägerstraand, T.: What about people in regional science? Papers in regional science 24(1), 7–24 (1970)
9. Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., Kolas, D.: A geo-ontology design pattern for semantic trajectories. In: Spatial Information Theory, pp. 438–456. Springer (2013)