EUON Workshop, 25th September 2014

OOPS! (OntOlogy Pitfall Scanner!): A WEB-BASED TOOL FOR ONTOLOGY EVALUATION

María Poveda-Villalón, Mari Carmen Suárez-Figueroa and Asunción Gómez-Pérez
Ontology Engineering Group.
Universidad Politécnica de Madrid.
{mpoveda, mcsuarez, asun}@fi.upm.es

OOPS! website: http://www.oeg-upm.net/oops
Twitter account: @OOPSoeg

This work has been partially supported by the Spanish project "4V: volumen, velocidad, variedad y validez en la gestión innovadora de datos" (TIN2013-46238-C4-2-R) and the European project "Ready4SmartCities - ICT Roadmap and Data Interoperability for Energy Systems in Smart Cities" (FP7-608711).
Table of Contents

• Introduction
• Pitfall Catalogue
• OOPS! (OntOlogy Pitfall Scanner!)
• Conclusions and Future Work
Methodologies (e.g: Methontology [1, 2], On-To-Knowledge [3], DILIGENT [4], and the NeOn Methodology [5]) that support the ontology development transformed the art of building ontologies into an engineering activity.

The correct application of such methodologies benefits the ontology quality.

However

Developers must tackle a wide range of difficulties and handicaps when modelling ontologies.

These difficulties can imply the appearance of anomalies or worst practices in ontologies.

Ontology evaluation (checking the technical quality of an ontology against a frame of reference) is a crucial activity in ontology engineering projects.

A lot of work has been done in ontology evaluation:

- **generic quality evaluation frameworks** [1, 2, 3, 4, 5],
- **methods based on the final (re)use of the ontology** [6]
- **quality models based on features, criteria and metrics** [7, 8]
- **tools**: ODEclean, ODEval, XDTools, OntoCheck, EyeBall, MoKi, etc.

Ontology evaluation is still largely neglected by developers and practitioners:

(a) **the current time-consuming** and **tedious** nature of evaluating the quality of an ontology
(b) **the lack of awareness** of the necessity for evaluating ontologies we are producing and publishing throughout the web.

---

Our objective is

- To **ease** the activity of **ontology evaluation** (mainly people who are not ontological engineers)
- To **reduce time** and **effort** in **ontology evaluation**

For doing so we have

(a) **Created** a **catalogue** contained potential errors we have seen in other ontologies and other authors’ work
(b) **Established** mechanisms to keep this **catalogue updated** and **maintained**
(c) **Automated** the detection of several errors
(d) **Provided** a web-based **user interface** and a **Restful web service**
Creation of the pitfall catalogue & maintenance process

Ontological Engineering

Linked Data

Manual review of ontologies
Bibliography review

Ontology Experts

creation

1st catalogue

implementation

http://oeg-upm.net/oops

Live Catalogue

OOPS!

Ontology Experts

extension

pitfall review

pitfall suggestion

OOPS! Users

720 random ontologies submitted by users

Ontology Experts

OOPS! (OntOlogy Pitfall Scanner!): a web-based tool for ontology evaluation
Pitfall Catalogue (important) Notes

• Pitfalls **could represent** or **lead to** an error.
• Pitfalls are **not necessarily** errors. For example, pitfalls **might not** represent an error depending on:
  • Modelling decisions.
  • Context or scope of the ontology.
  • Ontology requirements.
• In addition not all the pitfalls are equally important.
• An **indicator (critical, important, minor)** has been **associated** to each pitfall according to their **possible** negative consequences
  • **Survey** on ontology pitfalls importance: [http://goo.gl/SEddMN](http://goo.gl/SEddMN)
OOPS! (OntOlogy Pitfall Scanner!): a web-based tool for ontology evaluation

www.oeg-upm.net/oops/catalogue.jsp
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Pitfall Catalogue</td>
</tr>
<tr>
<td><strong>OOPS! (OntOlogy Pitfall Scanner!)</strong></td>
</tr>
<tr>
<td>Conclusions and Future Work</td>
</tr>
</tbody>
</table>
Web-based tool
Available at http://www.oeg-upm.net/oops
Ontology development environment independent
No installation process required

Jena API: http://jena.sourceforge.net/
Java EE: http://www.oracle.com/technetwork/java/javaee/overview/index.html
HTML: http://www.w3.org/html/wg/
jQuery: http://jquery.com/
CSS: http://www.w3.org/Style/CSS/
• **32 pitfalls implemented** out of 40 included in the catalogue
• 1 Java class per pitfall implementation
• **Detection** automated in 3 ways:
  • **Lexical content analysis:** make use of the content of annotations and identifiers for detecting pitfalls. E.g: P22: Using different naming criteria in the ontology.
  • **Seeking a particular characteristic:** check general characteristics of the ontology not related to the internal structure of the ontology or to the content of the lexical entities. E.g: P36. URI contains file extension.
  • **Structural pattern:** analyze the internal structure of the ontology, seeking specific parts of the model. E.g: P5: Defining wrong inverse relationships.
OOPS! (OntOlogy Pitfall Scanner) helps you to detect some of the most common pitfalls appearing when developing ontologies.

To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

Scanner by URI:  
Example: http://data.semanticweb.org/ns/swc/swc_2000-05-09.rdf

Scanner by direct input: 

If you just include the RDF code here, the following Pitfalls will not be checked: P36. URI contains file extension, P37. Ontology not available, P40. Namespace hijacking

Uncheck this checkbox if you don’t want us to keep a copy of your ontology.

News!

Now you can integrate OOPS! pitfall detection with your own developments and tools simply by invoking the OOPS! RESTful Web Service.

Detecting common pitfalls in ontologies

Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

In this sense, OOPS! helps you to detect some of the most common pitfalls appearing within ontology developments. For example, OOPS! warns you when:

- The domain or range of a relationship is defined as the intersection of two or more classes. This warning could avoid reasoning problems in case those classes could not share instances.
- No naming convention is used in the identifiers of the ontology elements. In this case, the maintainability, the accessibility and the clarity of the ontology could be improved.
- A cycle between two classes in the hierarchy is included in the ontology. Detecting this situation could avoid modelling and reasoning problems.
- And many other problems described in the catalogue.

Have a look at OOPS! results for the DBpedia 3.8 and AKT Reference Ontology (Portal Ontology) ontologies.

Please, help us making OOPS! better. Feedback is more than welcome and you can also suggest new pitfalls!
OOPS! - How it works (i)

News!

Now you can integrate OOPS! pitfall detection with your own developments and tools simply by invoking the OOPS! RESTful Web Service.

Detecting common pitfalls in ontologies

Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

In this sense, OOPS! helps you to detect some of the most common pitfalls appearing within ontology developments. For example, OOPS! warns you when:

- The domain or range of a relationship is defined as the intersection of two or more classes. This warning could avoid reasoning problems in case those classes could not share instances.
- No naming convention is used in the identifiers of the ontology elements. In this case the maintainability, the accessibility and the clarity of the ontology could be improved.
- A cycle between two classes in the hierarchy is included in the ontology. Detecting this situation could avoid modelling and reasoning problems.
- And many other problems described in the catalogue.

Have a look at OOPS! results for the DBpedia 3.8 and AKT Reference Ontology (Portal Ontology) ontologies.

Please, help us making OOPS! better. Feedback is more than welcome and you can also suggest new pitfalls!
OOPS! (OntOlogy Pitfall Scanner!) helps you to detect some of the most common pitfalls appearing when developing ontologies. To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

**Ontology input area**

**Brief description**

**News!**

Now you can integrate OOPS! pitfall detection with your own developments and tools simply by invoking the OOPS! RESTful Web Service.

**Detecting common pitfalls in ontologies**

Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

In this sense, OOPS! helps you to detect some of the most common pitfalls appearing within ontology developments. For example, OOPS! warns you when:

- The domain or range of a relationship is defined as the intersection of two or more classes. This warning could avoid reasoning problems in case those classes could not share instances.
- No naming convention is used in the identifiers of the ontology elements. In this case the maintainability, the accessibility and the clarity of the ontology could be improve.
- A cycle between two classes in the hierarchy is included in the ontology. Detecting this situation could avoid modelling and reasoning problems.
- And many other problems described in the catalogue.

Have a look at OOPS! results for the DBpedia 3.0 and AKT Reference Ontology (Portal Ontology) ontologies.

Please, help us making OOPS! better. Feedback is more than welcome and you can also suggest new pitfalls!
OOPS! - How it works (i)

OOPS! (OntOlogy Pitfall Scanner!) helps you to detect some of the most common pitfalls appearing when developing ontologies. To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

Scanner by URI:
Example: http://data.semanticweb.org/ns/svo/swc_2009-05-09.rdf

If you just include the RDF code here, the following Pitfalls will not be checked: P36. URI contains file extension, P37. Ontology not available, P40. Namespace hijacking

Scanner by direct input:

☑ Uncheck this checkbox if you don’t want us to keep a copy of your ontology.

Go to advanced evaluation

News!
Now you can integrate OOPS! pitfall detection with your own developments and tools simply by invoking the OOPS! RESTful Web Service.

Detecting common pitfalls in ontologies
Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

In this sense, OOPS! helps you to detect some of the most common pitfalls appearing within ontology developments. For example, OOPS! warns you when:
- The domain or range of a relationship is defined as the intersection of two or more classes. This warning could avoid reasoning problems in cases those classes could not share instances.
- No naming convention is used in the identifiers of the ontology elements. In this case the maintainability, the accessibility and the clarity of the ontology could be improved.
- A cycle between two classes in the hierarchy is included in the ontology. Detecting this situation could avoid modelling and reasoning problems.
- And many other problems described in the catalogue.

Have a look at OOPS! results for the DBpedia 3.8 and AKT Reference Ontology (Portal Ontology) ontologies.

Please, help us making OOPS! better. Feedback is more than welcome and you can also suggest new pitfalls!

Suggestions & feedback

Want to help?
- Suggest new pitfalls
- Provide feedback

Documentation:
- Pitfall catalogue
- User guide
- Technical report

Related papers:
- EKAW 2012
- ESWC 2012 Demo
- Ontoqual 2010
- CAEPFA 2009

Web services:
- RESTful Web Service
OOPS! (OntOlogy Pitfall Scanner!) helps you to detect some of the most common pitfalls appearing when developing ontologies. To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

Scanner by URI:
Example: http://data.semanticweb.org/ms/svc/wc_2009-05-09.rdf

If you just include the RDF code here, the following Pitfalls will not be checked: P36. URI contains file extension, P37. Ontology not available, P40. Namespace hijacking.

Brief description

News!

Now you can integrate OOPS! pitfall detection with your own developments and tools simply by invoking the OOPS! RESTful Web Service.

Documentation:
- Pitfall catalogue
- User guide
- Technical report

Related papers:
- EKAW 2012
- ESWC 2012 Demo
- Ontoqual 2010
- CAIEPI 2009

Want to help?
- Suggest new pitfalls
- Provide feedback

Suggestions & feedback

Detecting common pitfalls in ontologies

Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

In this sense, OOPS! helps you to detect some of the most common pitfalls appearing within ontology developments. For example, OOPS! warns you when:
- The domain or range of a relationship is defined as the intersection of two or more classes. This warning could avoid reasoning problems in case those classes could not share instances.
- No naming convention is used in the identifiers of the ontology elements. In this case the maintainability, the accessibility and the clarity of the ontology could be improved.
- A cycle between two classes in the hierarchy is included in the ontology. Detecting this situation could avoid modelling and reasoning problems.
- And many other problems described in the catalogue.

Have a look at OOPS! results for the DBpedia 3.8 and AKT Reference Ontology (Portal Ontology) ontologies.

Please, help us making OOPS! better. Feedback is more than welcome and you can also suggest new pitfalls!
OOPS! Ontology Pitfall Scanner! helps you to detect some of the most common pitfalls appearing when developing ontologies. To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

Scanner by URI:
Example: http://data.semanticweb.org/ns/svo/swc_2009-05-09.rdf

If you just include the RDF code here, the following Pitfalls will not be checked: P36. URI contains file extension, P37. Ontology not available, P40. Namespace hijacking

Scanner by direct input:

Uncheck this checkbox if you don’t want us to keep a copy of your ontology.

News!
Now you can integrate OOPS! pitfall detection with your own developments and tools simply by invoking the OOPS! RESTful Web Service.

Detecting common pitfalls in ontologies
Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

In this sense, OOPS! helps you to detect some of the most common pitfalls appearing within ontology developments. For example, OOPS! warns you when:

- The domain or range of a relationship is defined as the intersection of two or more classes. This warning could avoid reasoning problems in case those classes could not share instances.
- No naming convention is used in the identifiers of the ontology elements. In this case the maintainability, the accessibility and the clarity of the ontology could be improved.
- A cycle between two classes in the hierarchy is included in the ontology. Detecting this situation could avoid modelling and reasoning problems.
- And many other problems described in the catalogue.

Have a look at OOPS! results for the DBpedia 3.8 and AKT Reference Ontology (Portal Ontology) ontologies.

Please, help us making OOPS! better. Feedback is more than welcome and you can also suggest new pitfalls!

Want to help?
- Suggest new pitfalls
- Provide feedback

Documentation:
- Pitfall catalogue
- User guide
- Technical report

Related papers:
- EKAW 2012
- ESWC 2012 Demo
- On-toqual 2010
- CAEPFI 2009

Web services:
- RESTful Web Service
OOPS! (OntOlogy Pitfall Scanner!) helps you to detect some of the most common pitfalls appearing when developing ontologies. To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

Scanner by URI:
Example: http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf

Scanner by direct input:

Uncheck this checkbox if you don’t want us to keep a copy of your ontology.

Now you can integrate OOPS! pitfall detection with your own developments and tools simply by invoking the OOPS! RESTful Web Service.

Want to help?
- Suggest new pitfalls
- Provide feedback

Documentation:
- Pitfall catalogue
- User guide
- Technical report

Related papers:
- EKAW 2012
- ESWC 2012 Demo
- Ontoqual 2010
- CAEPPIA 2009

Web services:
- RESTful Web Service

Brief description

Ontology input area

Suggestions & feedback

Detecting common pitfalls in ontologies

Modelling ontologies has become one of the main topics of research within ontological engineering because of the difficulties it involves. Developers must tackle a wide range of difficulties and handicaps when modelling ontologies that can imply the appearance of anomalies or errors in ontologies. Therefore, it is important to evaluate the ontologies in order to detect those potential problems.

In this sense, OOPS! helps you to detect some of the most common pitfalls appearing within ontology developments. For example, OOPS! warns you when:

- The domain or range of a relationship is defined as the intersection of two or more classes. This warning could avoid reasoning problems in case those classes could not share instances.
- No naming convention is used in the identifiers of the ontology elements. In this case the maintainability, the accessibility and the clarity of the ontology could be improved.
- A cycle between two classes in the hierarchy is included in the ontology. Detecting this situation could avoid modelling and reasoning problems.
- And many other problems described in the catalogue.

Have a look at OOPS! results for the DBpedia 3.8 and AKT Reference Ontology (Portal Ontology) ontologies.

Please, help us making OOPS! better. Feedback is more than welcome and you can also suggest new pitfalls!
Evaluation results

Results for P04: Creating unconnected ontology elements. 11 cases | Minor

Results for P05: Defining wrong inverse relationships. 2 cases | Critical

Two relationships are defined as inverse relations when they are not necessarily. For example, something is sold or something is bought; in this case, the relationships "isSoldIn" and "isBoughtIn" are not inverse.

- This pitfall appears in the following elements:
  - http://data.semanticweb.org/ns/swc/ontology#relatedToEvent may not be inverse of http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument
  - http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument may not be inverse of http://data.semanticweb.org/ns/swc/ontology#relatedToEvent

Results for P08: Missing annotations. 156 cases | Minor

Results for P11: Missing domain or range in properties. 83 cases | Important

Results for P12: Missing equivalent properties. 8 cases | Important

When an ontology is imported into another, classes that are duplicated in both ontologies are normally defined as equivalent classes. However, the ontology developer misses the definition of equivalent properties in those cases of duplicated relationships and attributes. For example, the classes "CITY" and "City" in two different ontologies are defined as equivalent classes; however, relationships "hasMember" and "has-Member" in two different ontologies are not defined as equivalent relations.

- The following relations could be defined as equivalent:

- The following attributes could be defined as equivalent:

Example generated using the ontology http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
## Evaluation results

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Results for P04: Creating unconnected ontology elements.</th>
<th>Results for P05: Defining wrong inverse relationships.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 cases</td>
<td>Minor</td>
</tr>
</tbody>
</table>

Two relationships are defined as inverse relations when they are not necessarily. For example, something is sold or something is bought; in this case, the relationships "isSoldIn" and "isBoughtIn" are not inverse.

- This pitfall appears in the following elements:
  - http://data.semanticweb.org/ns/swc/ontology#relatedToEvent may not be inverse of http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument
  - http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument may not be inverse of http://data.semanticweb.org/ns/swc/ontology#relatedToEvent

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Results for P08: Missing annotations.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>156 cases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Results for P11: Missing domain or range in properties.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>83 cases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Results for P12: Missing equivalent properties.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 cases</td>
</tr>
</tbody>
</table>

When an ontology is imported into another, classes that are duplicated in both ontologies are normally defined as equivalent classes. However, the ontology developer misses the definition of equivalent properties in those cases of duplicated relationships and attributes. For example, the classes "CITY" and "City" in two different ontologies are defined as equivalent classes; however, relationships "hasMember" and "has-Member" in two different ontologies are not defined as equivalent relations.

- The following relations could be defined as equivalent:

- The following attributes could be defined as equivalent:

Example generated using the ontology http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
## Evaluation results

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Pitfall frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOPS! - How it works (ii)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results for P04: Creating unconnected ontology elements.</th>
<th>11 cases</th>
<th>Minor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results for P05: Defining wrong inverse relationships.</td>
<td>2 cases</td>
<td>Critical</td>
</tr>
</tbody>
</table>

Two relationships are defined as inverse relations when they are not necessarily. For example, something is sold or something is bought; in this case, the relationships "isSoldIn" and "isBoughtIn" are not inverse.

- This pitfall appears in the following elements:
  - http://data.semanticweb.org/ns/swc/ontology#relatedToEvent may not be inverse of http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument
  - http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument may not be inverse of http://data.semanticweb.org/ns/swc/ontology#relatedToEvent

<table>
<thead>
<tr>
<th>Results for P08: Missing annotations.</th>
<th>156 cases</th>
<th>Minor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results for P11: Missing domain or range in properties.</td>
<td>83 cases</td>
<td>Important</td>
</tr>
<tr>
<td>Results for P12: Missing equivalent properties.</td>
<td>8 cases</td>
<td>Important</td>
</tr>
</tbody>
</table>

When an ontology is imported into another, classes that are duplicated in both ontologies are normally defined as equivalent classes. However, the ontology developer misses the definition of equivalent properties in those cases of duplicated relationships and attributes. For example, the classes "CITY" and "City" in two different ontologies are defined as equivalent classes; however, relationships "hasMember" and "has-Member" in two different ontologies are not defined as equivalent relations.

- The following relations could be defined as equivalent:

- The following attributes could be defined as equivalent:

Example generated using the ontology http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
Evaluation results

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Pitfall frequency</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results for P04: Creating unconnected ontology elements.</td>
<td>11 cases</td>
<td>Minor</td>
</tr>
<tr>
<td>Results for P05: Defining wrong inverse relationships.</td>
<td>2 cases</td>
<td>Critical</td>
</tr>
<tr>
<td>Results for P08: Missing annotations.</td>
<td>156 cases</td>
<td>Minor</td>
</tr>
<tr>
<td>Results for P11: Missing domain or range in properties.</td>
<td>83 cases</td>
<td>Important</td>
</tr>
<tr>
<td>Results for P12: Missing equivalent properties.</td>
<td>8 cases</td>
<td>Important</td>
</tr>
</tbody>
</table>

Two relationships are defined as inverse relations when they are not necessarily. For example, something is sold or something is bought; in this case, the relationships "isSoldIn" and "isBoughtIn" are not inverse.

- This pitfall appears in the following elements:
  - http://data.semanticweb.org/ns/swc/ontology#relatedToEvent may not be inverse of http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument
  - http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument may not be inverse of http://data.semanticweb.org/ns/swc/ontology#relatedToEvent

When an ontology is imported into another, classes that are duplicated in both ontologies are normally defined as equivalent classes. However, the ontology developer misses the definition of equivalent properties in those cases of duplicated relationships and attributes. For example, the classes "CITY" and "City" in two different ontologies are defined as equivalent classes; however, relationships "hasMember" and "has-Member" in two different ontologies are not defined as equivalent relations.

- The following relations could be defined as equivalent:

- The following attributes could be defined as equivalent:
## Evaluation results

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Pitfall description</th>
<th>Pitfall frequency</th>
<th>Importance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P04: Creating unconnected ontology elements.</strong></td>
<td>Two relationships are defined as inverse relations when they are not necessarily. For example, something is sold or something is bought; in this case, the relationships &quot;isSoldIn&quot; and &quot;isBoughtIn&quot; are not inverse.</td>
<td>11 cases</td>
<td>Minor</td>
</tr>
<tr>
<td><strong>P05: Defining wrong inverse relationships.</strong></td>
<td></td>
<td>2 cases</td>
<td>Critical</td>
</tr>
<tr>
<td><strong>P06: Missing annotations.</strong></td>
<td></td>
<td>156 cases</td>
<td>Minor</td>
</tr>
<tr>
<td><strong>P11: Missing domain or range in properties.</strong></td>
<td></td>
<td>83 cases</td>
<td>Important</td>
</tr>
<tr>
<td><strong>P12: Missing equivalent properties.</strong></td>
<td></td>
<td>8 cases</td>
<td>Important</td>
</tr>
</tbody>
</table>

### Results for P04: Creating unconnected ontology elements.

- This pitfall appears in the following elements:
  - http://data.semanticweb.org/ns/swc/ontology#relatedToEvent may not be inverse of http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument
  - http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument may not be inverse of http://data.semanticweb.org/ns/swc/ontology#relatedToEvent

### Results for P06: Missing annotations.

### Results for P11: Missing domain or range in properties.

### Results for P12: Missing equivalent properties.

When an ontology is imported into another, classes that are duplicated in both ontologies are normally defined as equivalent classes. However, the ontology developer misses the definition of equivalent properties in those cases of duplicated relationships and attributes. For example, the classes "CITY" and "City" in two different ontologies are defined as equivalent classes; however, relationships "hasMember" and "has-Member" in two different ontologies are not defined as equivalent relations.

- The following relations could be defined as equivalent:

- The following attributes could be defined as equivalent:

Example generated using the ontology http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
### Evaluation results

<table>
<thead>
<tr>
<th>Pitfall name</th>
<th>Pitfall frequency</th>
<th>Importance Level</th>
<th>Ontology elements affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Results for P04: Creating unconnected ontology elements.</td>
<td>11 cases</td>
<td>Minor</td>
<td></td>
</tr>
<tr>
<td>Results for P05: Defining wrong inverse relationships.</td>
<td>2 cases</td>
<td>Critical</td>
<td></td>
</tr>
<tr>
<td>Results for P08: Missing annotations.</td>
<td>156 cases</td>
<td>Minor</td>
<td></td>
</tr>
<tr>
<td>Results for P11: Missing domain or range in properties.</td>
<td>83 cases</td>
<td>Important</td>
<td></td>
</tr>
<tr>
<td>Results for P12: Missing equivalent properties.</td>
<td>8 cases</td>
<td>Important</td>
<td></td>
</tr>
</tbody>
</table>

Two relationships are defined as inverse relations when they are not necessarily. For example, something is sold or something is bought; in this case, the relationships "isSoldIn" and "isBoughtIn" are not inverse.

- This pitfall appears in the following elements:
  > http://data.semanticweb.org/ns/swc/ontology#relatedToEvent may not be inverse of http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument
  > http://data.semanticweb.org/ns/swc/ontology#hasRelatedDocument may not be inverse of http://data.semanticweb.org/ns/swc/ontology#relatedToEvent

When an ontology is imported into another, classes that are duplicated in both ontologies are normally defined as equivalent classes. However, the ontology developer misses the definition of equivalent properties in those cases of duplicated relationships and attributes. For example, the classes "CITY" and "City" in two different ontologies are defined as equivalent classes; however, relationships "hasMember" and "has-Member" in two different ontologies are not defined as equivalent relations.

- The following relations could be defined as equivalent:
  > http://swrc.ontoware.org/ontology#member, http://xmlns.com/foaf/0.1/member

- The following attributes could be defined as equivalent:
  > http://www.w3.org/2002/12/cal/ical#location, http://swrc.ontoware.org/ontology#location
  > http://swrc.ontoware.org/ontology#firstName, http://xmlns.com/foaf/0.1/firstName
  > http://swrc.ontoware.org/ontology#date, http://www.w3.org/2002/12/cal/ical#date

Example generated using the ontology http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf
Table of Contents

- Introduction
- Pitfall Catalogue
- OOPS! (OntOlogy Pitfall Scanner!)
- Conclusions and Future Work
Conclusions and Future Work (i)

**Conclusions**

| Catalogue | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   | o is fully **independent** of any ontology **development environment** .  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   | o works with **main web browsers** (Firefox, Chrome, Safari and IE).  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   | o does **not** involve **installation** process.  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   | o **RESTful web service** available  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   | • Everyone can test it, provide **feedback**, **suggest new pitfalls** to be included in the catalogue and implemented into the tool.  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   | o easy to use  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   |   • feedback from a number of users by emails and feedback form  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   | o broadly used  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   |   • >2000 executions from 48 countries  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           |   |   • >600 different ontologies  
|           | • It is **freely available** to users on the Web: http://www.oeg-upm.net/oops  
|           | • Importance indicators  
|           | • Linked Data requirements considered  
|           | • Integrated within third-party software: OntoHub (https://ontohub.org/), SmartCity catalogue (http://smartcity.linkeddata.es/), Widoco (https://github.com/dgarijo/Widoco/)  

**OOPS! OntOlogy Pitfall Scanner!**

• Currently **40** pitfalls including other authors’ work
• **Maintained** and open to users/experts/practitioners point of view (**collaborative**)
Conclusions and Future Work (ii)

https://ontohub.org/

**Bioportal**

Overview Ontologies File browser History Settings

**Protein-Protein Interaction Ontology** OWL

Ontology defined in the file /biportal/PPID.obo
http://ontohub.org/biportal/PPID

A structured controlled vocabulary for the annotation of experiments concerned with protein-protein interactions. Developed by the HUPO Proteomics Standards Initiative.

<table>
<thead>
<tr>
<th>AnnotationProperties</th>
<th>Classes</th>
<th>Individuals</th>
<th>ObjectProperties</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2453</td>
<td>3</td>
<td>5</td>
<td>2961182</td>
</tr>
</tbody>
</table>

1 2 3 4 5 ... Next Last

25 : | per page

- MS 1002518
- MS 1002517
- MS 1002516
- MS 1002515
Conclusions and Future Work (ii)

https://ontohub.org/

http://smartcity.linkeddata.es/
Conclusions and Future Work (ii)

https://ontohub.org/

http://smartcity.linkeddata.es/

A structured controlled vocabulary for the annotation of experiments concerned with protein-protein interactions. Developed by the HUPO Proteomics Initiative.

https://github.com/dgarijo/Widoco/
## Conclusions and Future Work (iii)

### Future Work

<table>
<thead>
<tr>
<th>Catalogue</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Continuous maintenance</td>
<td></td>
</tr>
<tr>
<td>• To include <strong>guidelines</strong> about how <strong>to solve</strong> each pitfall</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OOPS! OntOlogy Pitfall Scanner!</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• To <strong>automate</strong> the <strong>remaining</strong> 8 pitfalls</td>
<td></td>
</tr>
<tr>
<td>• To <strong>improve</strong> the <strong>detection methods</strong> of some of the already implemented pitfalls (mainly by using linguistic resources)</td>
<td></td>
</tr>
<tr>
<td>• To allow pitfalls <strong>definition</strong> following a <strong>formal language</strong>, according with their particular quality criteria</td>
<td></td>
</tr>
<tr>
<td>• To <strong>integrate</strong> it in <strong>ontology editors</strong> (E.g: NeOn Toolkit and Protégé)</td>
<td></td>
</tr>
<tr>
<td>• To provide compliance badges (on going work)</td>
<td></td>
</tr>
</tbody>
</table>

### Publications

Thanks for your attention!
EUON Workshop, 25th September 2014

OOPS! (OntOlogy Pitfall Scanner!): A WEB-BASED TOOL FOR ONTOLOGY EVALUATION

María Poveda-Villalón, Mari Carmen Suárez-Figueroa and Asunción Gómez-Pérez
Ontology Engineering Group.
Universidad Politécnica de Madrid.
{mpoveda, mcsuarez, asun}@fi.upm.es

OOPS! website: http://www.oeg-upm.net/oops
Twitter account: @OOPSoeg

This work has been partially supported by the Spanish project "4V: volumen, velocidad, variedad y validez en la gestión innovadora de datos" (TIN2013-46238-C4-2-R) and the European project "Ready4SmartCities - ICT Roadmap and Data Interoperability for Energy Systems in Smart Cities" (FP7-608711).