Ontology Design Patterns (ODPs) for bio-ontologies

Mikel Egaña Aranguren (1), Robert Stevens (1), Erick Antezana (2)

(1) Manchester university
(2) Flanders Institute for Biotechnology/Ghent university

Bio-ontologies SIG at ISMB 2007

Outline

Outline

Outline

Outline

Outline

Outline

Outline

Introduction

- Useful bio-ontology: “high resolution” (rich) and rigorous representation of knowledge domain; more interesting queries and inferences.
- But rich and rigorous modelling is difficult for bio-ontologists.
- All the expressive power of OBO or OWL is not used.
- Expressivity only in term labels: useful for humans but computationally useless.
- A solution: ready-made modelling “recipes”: Ontology Design Patterns (ODPs). Rich and rigorous modelling with less effort.

Introduction

- Useful bio-ontology: “high resolution” (rich) and rigorous representation of knowledge domain; more interesting queries and inferences.
- But rich and rigorous modelling is difficult for bio-ontologists.
- All the expressive power of OBO or OWL is not used.
- Expressivity only in term labels: useful for humans but computationally useless.
- A solution: ready-made modelling “recipes”: Ontology Design Patterns (ODPs). Rich and rigorous modelling with less effort.

Introduction

- Useful bio-ontology: “high resolution” (rich) and rigorous representation of knowledge domain; more interesting queries and inferences.
- But rich and rigorous modelling is difficult for bio-ontologists.
- All the expressive power of OBO or OWL is not used.
- Expressivity only in term labels: useful for humans but computationally useless.
- A solution: ready-made modelling “recipes”: Ontology Design Patterns (ODPs). Rich and rigorous modelling with less effort.

Introduction

- Useful bio-ontology: “high resolution” (rich) and rigorous representation of knowledge domain; more interesting queries and inferences.
- But rich and rigorous modelling is difficult for bio-ontologists.
- All the expressive power of OBO or OWL is not used.
- Expressivity only in term labels: useful for humans but computationally useless.
- A solution: ready-made modelling “recipes”: Ontology Design Patterns (ODPs). Rich and rigorous modelling with less effort.

Introduction

- Useful bio-ontology: “high resolution” (rich) and rigorous representation of knowledge domain; more interesting queries and inferences.
- But rich and rigorous modelling is difficult for bio-ontologists.
- All the expressive power of OBO or OWL is not used.
- Expressivity only in term labels: useful for humans but computationally useless.
- A solution: ready-made modelling “recipes”: Ontology Design Patterns (ODPs). Rich and rigorous modelling with less effort.

What are ODPs?

- ODPs: known solutions to recurrent modelling problems of ontology engineering.
- Tested in different systems (efficient) and well documented.
- Similar idea to design patterns in OOP, but applied to ontologies.

What are ODPs?

- ODPs: known solutions to recurrent modelling problems of ontology engineering.
- Tested in different systems (efficient) and well documented.
- Similar idea to design patterns in OOP, but applied to ontologies.

What are ODPs?

- ODPs: known solutions to recurrent modelling problems of ontology engineering.
- Tested in different systems (efficient) and well documented.
- Similar idea to design patterns in OOP, but applied to ontologies.

Example: Value Partition

- We want to represent that a parameter can only take certain values, e.g. regulation can only be positive or negative.
- OWL: Covering and disjoint axioms.
- How can we build such structure in the ontology? Using the ODP Value Partition as a template.

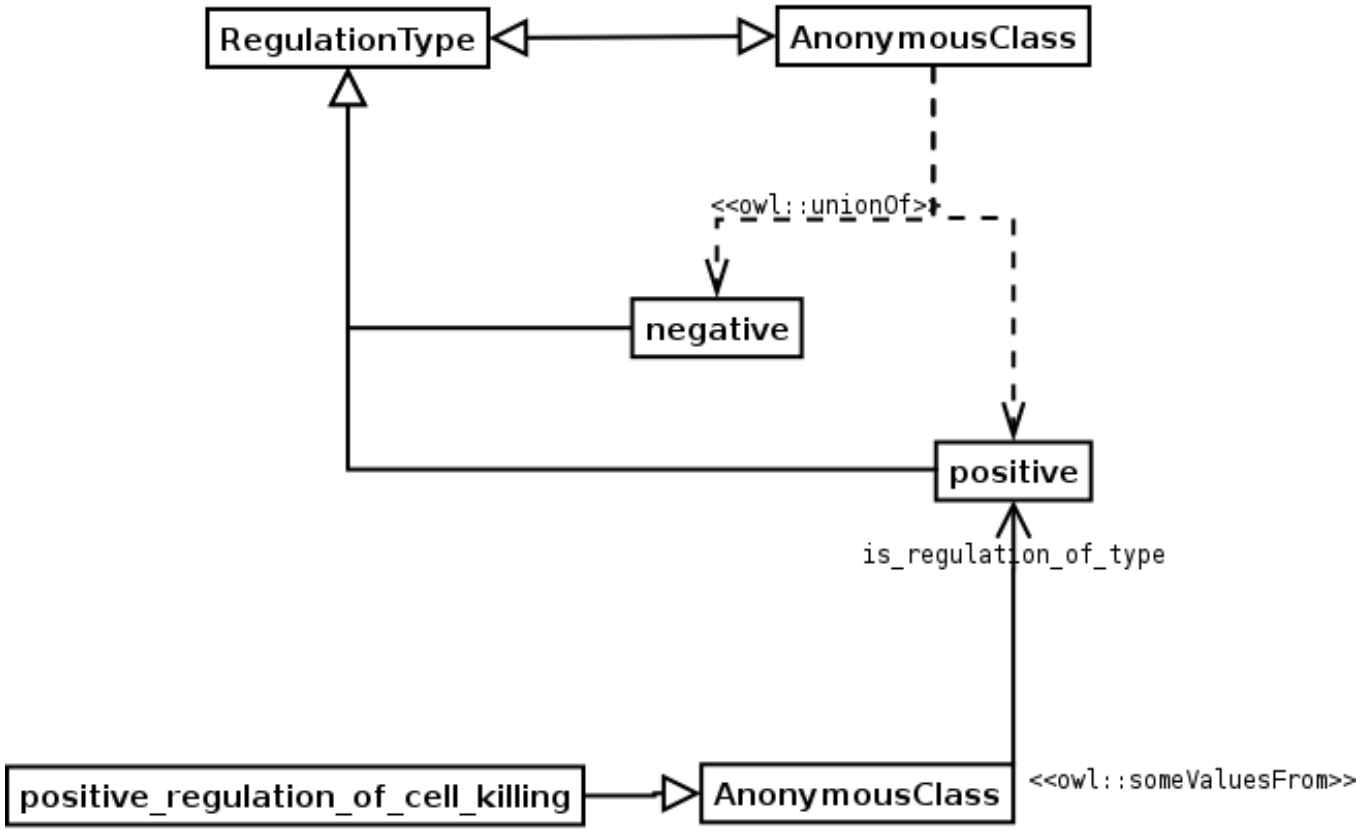
Example: Value Partition

- We want to represent that a parameter can only take certain values, e.g. regulation can only be positive or negative.
- OWL: Covering and disjoint axioms.
- How can we build such structure in the ontology? Using the ODP Value Partition as a template.

Example: Value Partition

- We want to represent that a parameter can only take certain values, e.g. regulation can only be positive or negative.
- OWL: Covering and disjoint axioms.
- How can we build such structure in the ontology? Using the ODP Value Partition as a template.

Example: Value Partition



Example: Upper Level Ontology

- OBO relations as a result of an upper level.
- A series of patterns capturing standard relationships:
  - Continuant part_of Continuant.
  - Continuant participates_in Occurent.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Advantages of using ODPs

- Rich and granular modelling.
- Focused development.
- Semantic encapsulation.
- Tooling.
- Robustness and modularity.
- Good communication.
- Documented modelling.
- Reasoning.
- Rapid prototyping.
- Alignment.
- Re-engineering.
- Comprehension of advances in KR.

Types of ODPs

- **Extensional ODPs:** Solutions to modelling situations at the limits of a KR language.
- **Good practice ODPs:** Capturing domain knowledge in perceived best practice: more modular, efficient and maintainable ontologies.
- **Domain Modelling ODPs:** Solutions to modelling situations within the scope of a KR language ("signature ODPs").

Direct application

- Recreate the structure of the ODP in the ontology “by hand”.
- Protégé wizards: `http://www.co-ode.org/downloads/wizard/`
- Import (OWL).

Direct application

- Recreate the structure of the ODP in the ontology “by hand”.
- Protégé wizards:
  `http://www.co-ode.org/downloads/wizard/`
- Import (OWL).

Direct application

- Recreate the structure of the ODP in the ontology “by hand”.
- Protégé wizards: `http://www.co-ode.org/downloads/wizard/`
- Import (OWL).

Application by condition matching: OPL

- OPL: Ontology Processing Language.
- Syntax for choosing entities in an ontology and adding new semantics to those entities.
- Syntax written in flat files and processed by the OPL engine: ODPs stored.
- `http://www.gong.manchester.ac.uk/downloads/`

Application by condition matching: OPL

- OPL: Ontology Processing Language.
- Syntax for choosing entities in an ontology and adding new semantics to those entities.
- Syntax written in flat files and processed by the OPL engine: ODPs stored.
- `http://www.gong.manchester.ac.uk/downloads/`

Application by condition matching: OPL

- OPL: Ontology Processing Language.
- Syntax for choosing entities in an ontology and adding new semantics to those entities.
- Syntax written in flat files and processed by the OPL engine: ODPs stored.
- http://www.gong.manchester.ac.uk/downloads/

Application by condition matching: OPL

- OPL: Ontology Processing Language.
- Syntax for choosing entities in an ontology and adding new semantics to those entities.
- Syntax written in flat files and processed by the OPL engine: ODPs stored.
- `http://www.gong.manchester.ac.uk/downloads/`

Application by condition matching: OPL

Ontology Processing Language (OPL)

```
SELECT ?x WHERE ?x label regulation;
ADD ?x equivalentTo (positive or negative);
ADD positive disjointWith negative;
```

Documentation system for ODPs

- Each ODP is described using some sections.
- Alpha version of public catalogue: `www.gong.manchester.ac.uk/ontologydesignpatterns/`
- Future implementation directly in OWL.

Documentation system for ODPs

- Each ODP is described using some sections.
- Alpha version of public catalogue: `www.gong.manchester.ac.uk/ontologydesignpatterns/`
- Future implementation directly in OWL.

Documentation system for ODPs

- Each ODP is described using some sections.
- Alpha version of public catalogue: `www.gong.manchester.ac.uk/ontologydesignpatterns/`
- Future implementation directly in OWL.

Sections of documentation system

- Name.
- A.K.A.
- URL.
- Classification.
- Motivation.
- Aim.
- Elements.
- Structure.
- Implementation.
- Result.
- Side effects.
- Sample.
- Known uses.
- Related ODPs.
- References.
- Additional information.

Actual applications of ODPs

- Gene Ontology Next Generation workflow.
- Normalisation of Gene Ontology's Molecular Function.
- Cell Cycle Ontology.
- Galen.

Actual applications of ODPs

- Gene Ontology Next Generation workflow.
- Normalisation of Gene Ontology's Molecular Function.
- Cell Cycle Ontology.
- Galen.

Actual applications of ODPs

- Gene Ontology Next Generation workflow.
- Normalisation of Gene Ontology's Molecular Function.
- Cell Cycle Ontology.
- Galen.

Actual applications of ODPs

- Gene Ontology Next Generation workflow.
- Normalisation of Gene Ontology's Molecular Function.
- Cell Cycle Ontology.
- Galen.

Conclusion

- ODPs make it possible to produce and maintain rich and rigorous bio-ontologies with less effort.
- Issues to solve:
  - Graphical metalanguage *a la* UML for ontologies.
  - Tools for easily creating, storing and sharing ODPs between bio-ontologists.
- Plenty of areas of biological knowledge in need of ODPs.

Conclusion

- ODPs make it possible to produce and maintain rich and rigorous bio-ontologies with less effort.
- Issues to solve:
  - Graphical metalanguage *a la* UML for ontologies.
  - Tools for easily creating, storing and sharing ODPs between bio-ontologists.
- Plenty of areas of biological knowledge in need of ODPs.

Conclusion

- ODPs make it possible to produce and maintain rich and rigorous bio-ontologies with less effort.
- Issues to solve:
  - Graphical metalanguage *a la* UML for ontologies.
  - Tools for easily creating, storing and sharing ODPs between bio-ontologists.
- Plenty of areas of biological knowledge in need of ODPs.

Acknowledgements

Mikel Egaña Aranguren is funded by Manchester University and EPSRC.

Erick Antezana is funded by EU (FP6, contract number LSHG-CT-2004-512143).