Ontology Engineering: From an Art to a Craft
The Case of the Data Mining Ontologies

Larisa Soldatova[1], Panče Panov[2], and Sašo Džeroski[2]

[1] Brunel University, London, United Kingdom
larisa.soldatova@brunel.ac.uk
[2] Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
{pance.panov, saso.dzeroski}@ijs.si

Abstract. In this paper, we report on our experience and discuss the problems we encountered while designing, implementing and revising a set of ontologies describing the domain of data mining. We focus on a set of key issues that we think are important and need to be addressed by the ontology engineering community. These include ontology evaluation, testing, versioning, the use of design patterns, the use of IT portal(s), re-usability, and compatibility. To illustrate the key issues we provide examples that originate from our work on the ontologies for data mining. We conclude the paper with a summary and some suggestions that we believe should be addressed by the ontology engineering research community.

1 Introduction

The progress of ontology engineering for the last two decades has been immense. It is now possible to learn ontologies from text, or construct it following the data-driven approach. Ontologies can be mapped, integrated and re-used. However many steps of ontology engineering remain an art: the design, testing, evaluation. It is still not engineering in a true sense.

For example, in the domain of machine learning, a subfield of computer science that explores the construction of algorithms that learn from and make predictions on data, there are clear procedures how a newly developed algorithm is evaluated, which performance measures should be used for the learning task at hand, and how one can compare with other algorithms defined for the same learning task. Another example is from computer programming, where tools provide complete support and procedures for software testing, versioning, and the use of patterns in the software engineering phase.

In this manuscript, we analyze the state of ontology engineering based on our extensive experience in the development of a number of ontologies, and in particular ontologies for the domain of data mining [1–3], identify the key bottlenecks in its progress and outline a way forward. We illustrate the key issues with examples from our ontologies for the domain of data mining.

2 Ontologies for the domain of data mining

For the domain of data mining and knowledge discovery, we have developed a modular ontology (named OntoDM) that is composed of three sub-ontologies that can be used together or as a stand-alone product, depending on the use case. This includes: the OntoDM-core ontology that represents core data mining entities [1][3], the OntoDT ontology that represents datatypes [2][4], and the OntoDM-KDD ontology that represents the process of knowledge discovery [3]. From the initial version of the ontology [4] to the current ontology releases, we have dealt with variety of problems and design choices largely due to the lack of guidelines for the development of IT ontologies. The simplest design decision in this case was to base our work on the established best practices from the biology and bio-medical domains (such as the OBO Foundry principles), which are still among the most developed domains in regards to ontologies.

In the course of the ontology development, other related ontologies describing the domains of machine learning, data mining and knowledge discovery from various perspectives appeared [5–7]. There was always a problem of how one can compare an ontology to other related domain ontologies and also how to evaluate the constructed ontology sufficiently. Another problem that arose is how to avoid the duplication of the efforts and to reuse classes from other DM ontologies. The reuse has not been straightforward, since the related ontologies were built using different design principles. Some of these issues are discussed further in the following section.

3 The Key Issues

Evaluation. In our work on ontologies for data mining, we evaluated the produced ontologies using the methodology proposed by Grüniger and Fox [8], which is based on an assessment whether the built ontologies answers the competency questions established in the design phase. In addition, we provided a subjective assessment of how the constructed ontology satisfied the design principles established in the design phase.

While these evaluation approaches are reasonable, they do not give sufficient confidence that an ontology is correct. They are mainly based on expert opinions: competency questions are created by experts or potential users, and a selection of the design principles is due to experts choice. There are no procedures like for example in machine learning to compare the performance of two algorithms on the same dataset using objective measures. There is a need for the research community to come up with a better evaluation approach than those currently available.

The development of ontologies is expensive, and therefore it is unlikely to have several ontologies for exactly the same domain to enable an accurate comparison of their performances. However, a comprehensive analyses and evaluation

[3] The ontology is available at `http://www.ontodm.com`
[4] The ontology is available at `http://www.ontodt.com`

of the key design principles could be performed: what works best for what types of problems/ domains: a 4D or a 3D approach; the use of Basic Formal Ontology (BFO) [9] or an ad-hoc upper-level ontology; the use of many or a few relations?

Testing. Testing is an essential part of any engineering project, be it a construction of a bridge or software development. Testing has similarities with evaluation, but it is a different process. On one hand, evaluation aims to identify if a product meets the specified requirement and is closely related with a quality assurance (i.e. a certain level of quality is typically one of the requirements). On the other hand, testing is executing a system in order to identify any errors. Obviously, a system with errors is not a high quality system. Finally, the results of testing are used to improve the system before evaluation.

Standard testing methods for software engineering (e.g. white/grey/black box) can be applied to ontology engineering. However, a development of an ontology has its distinct specificity due to explicit and implicit logical entailments. Unfortunately, there is no methodology available to construct a collection of tests to check if an ontology indeed outputs the expected outputs and does not outputs unexpected ones. Reasoners assist in the detection of logical consistency errors, but they would not detect unexpected outcomes. For example, one can develop a logically consistent pizza ontology where a vegetarian pizza has a meat topping (see the pizza tutorial[5]). Reasoners also do not detect such errors as for example using an `is-a` relation instead of a `part-of` relation. Even if an ontology is error free, and it imports another error free ontology, this does not guarantee that the resulting ontology will be error-free.

The available ontology development tools do not provide sufficient support for the design (manual or automated) and execution of tests. The exception is Tawny-OWL by Lord et al. [10]. The Tawny-OWL library provides a fully-programmatic environment for ontology building; it enables the use of a rich set of tools for ontology development, by recasting development as a form of programming. It is built in Clojure[6] - a modern Lisp dialect, and is backed by the OWL API. It provides a rich and modern programming tool chain, for versioning, distributed development, build, testing and continuous integration.

The barriers for the development and adoption of testing methodologies for ontology engineering are not only technological, but also sociological. There are expectations that a newly developed ontology should be evaluated before a release or a publication. Unfortunately, it is unusual to report on what set of tests an ontology has been tested. Journals do not require an inclusion of information on testing into papers reporting on ontology development. We as a research community need to change this. We also need to agree on a standard for ontology testing, similar to how it was done for software testing (see the IEEE Standard

[5] `http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/`

[6] `http://clojure.org/`

for Software Unit Testing[7]).

Versioning. An ontology, like any other any artifact, has a life cycle and it is changing over time. The changes of ontologies may cause interoperability problems. For example, a good practice is not to delete any of the classes, but to deprecate them. But not all developers do that. Ontologies have specificity regarding versioning (see [11]). Changes in ontologies might occur due to various issues and it is important to capture that.

Ontology development tools do not have inbuilt version control of ontology projects. The developers have to use external tools, like Git[8] to compensate for such shortcomings. While it does address the problem of ontology versioning, it is not a seamless process. In terms of versioning support on the level of languages, the OWL language (1.0) provides some built in versioning attributes[9] (e.g., owl:versionInfo, owl:priorVersion), but this is not enough to fully support versioning of ontologies in a systematic way.

Design patterns. Mature programming techniques rely on the use of patterns. They speed up the development process and reduce number of errors. Some patterns are available for ontology engineering. Ontology design patterns (ODPs) are ready made modelling solutions for creating and maintaining ontologies. ODPs help in creating rich and rigorous ontologies with less effort [13]. There is a public catalog of ODPs focused on the biological knowledge domain[10]. The OBI (the Ontology for Biomedical Investigations) project developed a pattern (*aka* a template) for the key class `assay`. Such an approach has significantly speeded up the submission of hundreds of subclasses. Hoehndorf *et al.* provide a prototype to extract relational patterns from OWL ontologies using automated reasoning [12]. However the described efforts are focusing mainly on biomedical areas. There is a need for an easily accessible collection of more generic ontology patterns.

IT Portal. We deposited our OntoDM and OntoDT ontologies to BioPortal[11]. Currently, it has 450 ontologies and the number is rapidly growing. The portal has an excellent search capabilities and also provides other useful tools, such as: federated querying engine, mapping service, an annotation service, ontology recommender based on an excerpt from a biomedical text or list of keywords and others [14]. However, the portal is supporting biomedical domains, and there is no such a portal for IT ontologies.

The absence of a portal which would provide similar services for IT ontologies contributes to the duplication of the efforts and inhibits the development of the area. For example, we now have several ontologies describing the domains

[7] `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=2599`
[8] `https://git-scm.com/`
[9] `http://www.w3.org/2007/OWL/wiki/Ontology_Versions`
[10] `https://git-scm.com/html/`
[11] `http://bioportal.bioontology.org/`

of machine learning, data mining and knowledge discovery that are not interoperable. Having an IT portal with the same functionalities as the BioPortal (or better), would ease the reuse of the ontologies, their discovery and mapping.

Reausability and Compatibiliy. The most common usage of ontologies remains to be as a controlled vocabulary[12]. However ontologies originally were viewed as 'building blocks' of information systems [15]. We believe that they are under-used in such a capacity and that we will see in the nearest future many interesting use-cases, where ontologies are integral components of complex systems. For example, an ontology LABORS is an integral part of the Robot Scientist system, which is capable of automated discovery of new functional genomics knowledge [16]. The PHenotypic Interpretation of Variants in Exomes (PHIVE) algorithm includes the phenotype manifestations in individuals as well as the signs and symptoms of diseases [17]. This work goes beyond the use of ontologies as controlled vocabularies and exploits hierarchical inheritance properties. It was shown that including phenotype information into the prioritisation of candidate genes leads to an up to 54.1 fold improvement over methods purely based on variant information.

Ontologies as potential information systems 'building blocks' needs to be represented as such to enable their discovery, reuse, integration and functioning as components of complex systems. For example, the service oriented architecture (SOA) approach can be adopted for the description of ontologies as services. SOA is viewed by W3C as a set of components which can be invoked, and whose interface descriptions can be published and discovered. An ontology should be 'wrapped-up' by the specification of what services it can provide, for what domain, what input/output and environment requirements are, provenance, the development stage, quality, etc. A collection of such ontologies would ease the design and implementation of complex information systems.

4 Discussion and Conclusion

We believe that ontology engineering is still far from reaching its full potential. The developers are struggling with multiple issues outlined above. A better support for the development and evaluation processes would speed up the research in this area. Sophisticated and easy to use development tools can low the barriers for the novice ontology designers and help in reducing errors. Dedicated IT portals would promote best practices in ontology engineering and support ontology reuse and their integration into complex intelligent systems. A mature methodology for the testing and evaluation of constructed ontologies would ensure that the deposited ontologies are of high quality. The research community needs to agree on the standards for ontology testing, description and quality assurance. We are confident that ontology engineering would become a true engineering if these objectives would be achieved.

[12] `http://www.w3.org/TR/webont-req/`

References

1. Panov, P., Soldatova, L., Džeroski, S.: Ontology of core data mining entities. Data Mining and Knowledge Discovery **28**(5-6) (2014) 1222–1265
2. Panov, P., Soldatova, L.N., Džeroski, S.: Generic ontology of datatypes. Information Sciences (2015)
3. Panov, P., Soldatova, L., Džeroski, S.: Ontodm-kdd: ontology for representing the knowledge discovery process. In: Discovery Science. Springer Berlin Heidelberg (2013) 126–140
4. Panov, P., Dzeroski, S., Soldatova, L.N.: Ontodm: An ontology of data mining. In: Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on, IEEE (2008) 752–760
5. Keet, C.M., Ławrynowicz, A., dAmato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., Hilario, M.: The data mining optimization ontology. Web Semantics: Science, Services and Agents on the World Wide Web (2015)
6. Diamantini, C., Potena, D., Storti, E.: A virtual mart for knowledge discovery in databases. Information Systems Frontiers **15**(3) (2013) 447–463
7. Vanschoren, J., Soldatova, L.: Exposé: An ontology for data mining experiments. In: International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010). (2010) 31–46
8. Grüninger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies. In: Proceedings of the IJCAI-95 Worshop on Basic Ontological Issues in Knowledge Sharing. (1995)
9. Arp, R., Smith, B., Spear, A.D.: Building ontologies with Basic Formal Ontology. MIT Press (2015)
10. Lord, P.: The semantic web takes wing: Programming ontologies with tawny-owl. arXiv preprint arXiv:1303.0213 (2013)
11. Klein, M.C., Fensel, D.: Ontology versioning on the semantic web. In: SWWS. (2001) 75–91
12. Hoehndorf, R., Oellrich, A., Dumontier, M., Kelso, J., Rebholz-Schuhmann, D., Herre, H.: Relations as patterns: bridging the gap between obo and owl. BMC bioinformatics **11**(1) (2010) 441
13. Aranguren, M.E., Antezana, E., Kuiper, M., Stevens, R.: Ontology design patterns for bio-ontologies: a case study on the cell cycle ontology. BMC bioinformatics **9**(Suppl 5) (2008) S1
14. Whetzel, P.L., Noy, N.F., Shah, N.H., Alexander, P.R., Nyulas, C., Tudorache, T., Musen, M.A.: Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. Nucleic acids research **39**(suppl 2) (2011) W541–W545
15. Mizoguchi, R.: Tutorial on ontological engineering part 1: introduction to ontological engineering. New Generation Computing **21**(4) (2003) 365–384
16. King, R.D., Rowland, J., Oliver, S.G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L.N., et al.: The automation of science. Science **324**(5923) (2009) 85–89
17. Robinson, P.N., Köhler, S., Oellrich, A., Wang, K., Mungall, C.J., Lewis, S.E., Washington, N., Bauer, S., Seelow, D., Krawitz, P., et al.: Improved exome prioritization of disease genes through cross-species phenotype comparison. Genome research **24**(2) (2014) 340–348