The Stub Metapattern

Adila A. Krisnadhi[1,2] and Pascal Hitzler[1]

[1] Wright State University, OH, USA
[2] Universitas Indonesia, Depok, Indonesia

Abstract. We present a minimalistic metapattern which we call the Stub pattern. It acts as a type of placeholder for future extensions of an ontology in cases where a more fine-grained modeling would currently be counterproductive, but future extensions may call for more details. We motivate the Stub pattern, define it, and provide examples.

1 Introduction and Motivation

When modeling an ontology, one of the issues to be addressed is that of granularity: To what detail should the ontology represent the notions it captures? Traditionally, this issue is resolved by looking at a concise definition of the use cases, e.g. by means of competency questions. As a result of this, some parts of the ontology may be modeled in a rather fine-grained manner, while other parts remain relatively coarse. This is not a defect, of course, it is rather very natural and cannot be avoided as such, since a model necessarily remains limited in both detail and scope. However, as we argue in this paper, a straightforward handling of differing granularity requirements in different parts of an ontology can make it more difficult to repurpose or extend the ontology, or to use it in an ontology-driven data integration setting.

Let us discuss an example. The W3C Organization Ontology [3] gives a rather fine-grained account of notions such as roles and positions with respect to an organization, which seems to be central in the context of models capturing organizations. At the same time, however, other parts of the model are not worked out in similar detail.

Let us look at a specific instance of this, depicted in Figure 1 which is taken from the class diagram of the Organization Ontology. Let us for a moment not worry about the question why one would want to assign static locations to persons, since persons are usually thought of as being rather mobile.[1] Rather, let us focus on the choice how location information is represented in this model, namely by means of strings.

Representing locations as strings may sometimes seem the straightforward thing to do, e.g., if the base data has only location names listed for the places it

[1] It seems to us that the location information should rather be attached to the site a person is based at – and in fact it should perhaps not be the person which is based at a site, but rather that the role this person has with respect to the organization be based at a particular site, and thus location. But this discussion leads us astray from the key point we want to make.

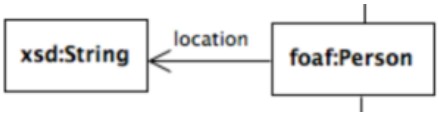

Fig. 1. Snippet from the class diagram of the W3C Organization Ontology [3].

Fig. 2. Places Stub

refers to, such as "Dublin." However, fundamentally, one has to question whether locations are strings; at least to us locations are rather complex entities different (indeed, disjoint) from strings. Locations, of course, can have names (which may be representable as strings), but locations may also have GPS coordinates (with or without indication of precision), bounding boxes, descriptions how to get there or how to best see them,[2] information who discovered them, what type of climate they have, etc. Any of this type of information may be interesting for some use case relevant to locations, or even relevant to organizations and their locations.

A rather obvious and straightforward use case which cannot be realized with strings as locations is that of co-location. Say, we would like to know whether the location "Dublin" of the Online Computer Library Center OCLC[3] is the same as the location of IBM Dublin. Or in other words, we would like to enrich our data with `owl:sameAs` links which identify which of the locations given for some organizations are actually the same. Alas, if the location is given as a string "Dublin" in both cases, we have no way to indicate easily that these refer to the same, or to different, locations.

A solution, of course, lies in the acknowledgement that locations are not strings, but rather – places. And that places in turn may have names. See Figure 2 for a schema diagram. Of course, in this case we still do not provide any more detailed modeling of place or location, i.e., what we have depicted is not a place pattern. However, it has advantages compared with the option depicted in Figure 1, in that it provides a "hook" – the node labelled Place – which can be used to attach additional information, or for providing `owl:sameAs` or `owl:differentFrom` relationships. In fact, the stub can easily be replaced with

[2] To see the Mont Blanc, most people do not actually go to the Mont Blanc, but rather to the top of the Aiguille du Midi, which has a viewing platform with convenient cable car access.

[3] This location name ended up as part of the name of *Dublin Core*.

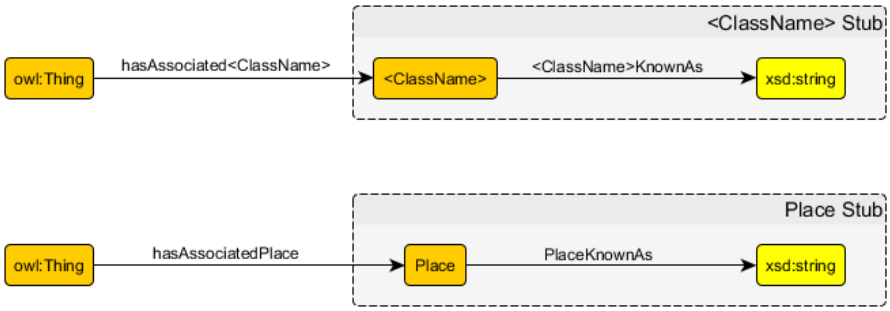

Fig. 3. The Stub metapattern (top) with an instance for Place (bottom)

a more fine-graned model of Place if needed for a new use case which calls for a more fine-grained modeling of locations.

2 The Metapattern

The situation just described does, of course, not only apply to places. It can likewise apply to persons, events, research projects, pets, movie characters, dinosaurs, emotions, particle accelerators and space stations. For each and many more we can use stubs, whenever the type at hand would call for a complex model, but at modeling time it is not (yet) called for to produce a more fine-grained model. Because there is a reasonable "stub" pattern for almost everything, the idea of a "stub" is a type of metapattern which has different instantiations for different class types.

Moreover, rather than concrete competency questions, we would have questions that appear like a template, such as (i) "What X is associated with a given object?"; or (ii) "Given an object and an instance of X associated with it, what is that instance of X also known as?", where X is the notion for which a stub pattern is intended.

In Figure 3, top, we indicate this metapattern by using <ClassName> as a type of variable for the target type; an instantiation for Place as the class type is given underneath, for illustration. A particular stub for type ClassName thus essentially consists of a class ClassName together with a known-as relation to a string. We indicate that it is a stub, by refering to the pattern as a "stub" pattern.

Concrete instantiations of the stub metapattern may of course deviate from this very simple recipe, e.g. by allowing alternatives to the known-as link which may be object properties pointing to controlled vocabularies. We have not really needed this yet, though, in our modeling ventures. Concrete instantiations of the metapattern should most often also have alternative names for the bi-

nary relations, e.g. for the Place Stub with name, one could use "atPlace" and "hasName" instead of the more generic ones we have indicated in Figure 3.

In terms of axioms, there are not many to be listed.[4] Two range restrictions, a scoped domain restriction, and possibly an existential, as given below in this sequence, exhaust what should be said.

$$\top \sqsubseteq \forall \texttt{hasAssociated<ClassName>.<ClassName>}$$
$$\top \sqsubseteq \forall \texttt{<ClassName>KnownAs.xsd:string}$$
$$\exists \texttt{<ClassName>KnownAs.xsd:string} \sqsubseteq \texttt{<ClassName>}$$
$$\texttt{<ClassName>} \sqsubseteq \exists \texttt{<ClassName>KnownAs.xsd:string}$$

Note that the <ClassName> notation should be understood as a place holder for concrete names, and axioms we introduced above should be viewed as a template, not a concrete reusable component. Obviously, this idea of stub metapattern is also not a feature of the OWL as an ontology language. Hence, to use this metapattern, one would need to instantiate it as a stub pattern by replacing <ClassName> in the axioms with a concrete class name, resulting in an actual OWL ontology. This could typically be achieved in ontology editors like Proégé, e.g., by string find and replace command, or more generally, through framework such as PatOMat [5].

Comparison with the Literal Reification Pattern

The stub metapattern is rather closely related with the Literal Reification pattern.[5] The latter allows any literal value to be given context or particular semantics by associating the literal with a proper OWL individual. The association is injective, i.e., different literal values are associated with different individuals. The Literal Reification pattern then introduces the class `litre:Literal`, which contains all such individuals. Also, this class is asserted to be a subclass of the class `region:Region`,[6] which is extracted from the DOLCE. Finally, this pattern has a hook for the object to which the literal is supposedly attached, which is simply represented as `owl:Thing`.

The conceptualization in the stub metapattern is somewhat similar: it contains a hook for the object to which the stub is associated, the main class of the stub, and a link to a literal value. However, the stub metapattern is actually more general and imposes less ontological commitment than the Literal Reification pattern. This is of course expected since the stub metapattern does not intend to lift the literal value as a "first-class object". Rather, it emphasizes on making a place holder for the notion modeled by the instantiation of the stub metapattern. Therefore, it imposes no restriction of one-to-one relationship between the literal value and the main class of the stub. Furthermore, there

[4] ODP Portal submission is at `http://ontologydesignpatterns.org/wiki/Submissions:Stub_Metapattern`

[5] `http://ontologydesignpatterns.org/wiki/Submissions:Literal_Reification`

[6] `http://ontologydesignpatterns.org/wiki/Submissions:Region`

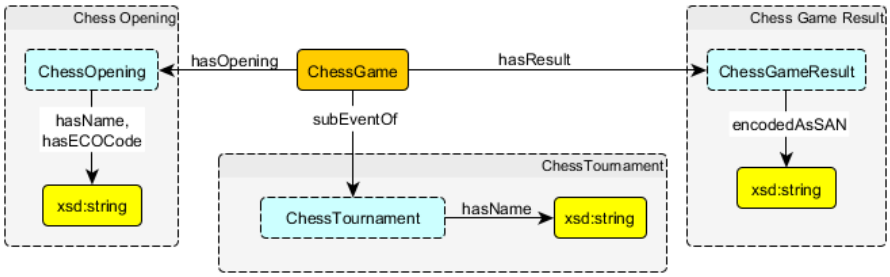

Fig. 4. Three stubs used in the Chess Ontology [2,4]

is no relation with other external class like `region:Region`. As a result, after instantiating the stub metapattern into a stub pattern for a particular notion, one could simply add more details to the stub if desired, including attaching additional literal values, without modifying the axioms originally obtained from that instantiation.

3 Examples

As additional examples, we borrow from the Chess Ontology [2,4]. In this particular case, initial data was obtained from PGN files, which are text files with information about specific chess games, including the move sequences, commentary, and some metadata such as the names of the players and the name of the tournament the game was played at. PGN files are the de-facto file type standard for chess games. Three stubs occur in this ontology; they are depicted in Figure 4 and we will discuss each of them briefly in the following.

Regarding chess tournaments, PGN files list only names of tournaments as strings. As argued in the introduction, in order to preserve the future possibility to expand on tournament information, e.g. by adding further information about specific tournaments which can be obtained from chess websites or Wikipedia, and in order to make it possible to establish same-as links between multiply mentioned tournaments with somewhat different name strings, stubs were our method of choice.

Similarly, in order to represent the result of a chess game, we went with a stub. However, in this case the result would be encoded as a string in Standard Algebraic Notation (SAN), as indicated by the property name. The stub is helpful for future development of the ontology, as the result of a chess game may be more than the simply score which indicates who won. E.g., the second game of the 1972 World Chess Championship was not only won by Boris Spasski, rather the game was remarkable because Bobby Fischer lost by forfeit, which is a rather unexpected occurrence in such a high-profile event. If desired, our stub

could easily be extended to also indicate such additional information as loss by forfeit, or win on time or by adjudication.

For chess openings, PGN files may indicate the opening by name or by the corresponding code according to the Encyclopedia of Chess Openings (ECO); correspondingly out stub bears two properties. Openings are rather central for the game of chess, and there exists a vast literature on opening theory. Openings can also be related to each other in complex ways. Using a stub provides the option of future expansion of the ontology in this direction.

4 Conclusions

We believe that ontology modeling should be modular [1] and based on well-designed, generic ontology design patterns. In particular, we believe that ontological commitments need careful thought in order to improve reusability of an ontology.

Using stubs makes it possible to deal with the seeminly contradictory requirements to avoid oversimplication (locations are strings) on the one hand, and overcomplication (locations modeled as very complex entities) on the other. They provide a middle ground, a compromise, which simplifies future extensibility but avoids modeling effort which is not required yet.

Acknowledgements. This work was supported by the National Science Foundation under award 1017225 *III: Small: TROn – Tractable Reasoning with Ontologies* and award 1440202 *EarthCube Building Blocks: Collaborative Proposal: GeoLink – Leveraging Semantics and Linked Data for Data Sharing and Discovery in the Geosciences.*

References

1. Krisnadhi, A., Hu, Y., Janowicz, K., Hitzler, P., Arko, R.A., Carbotte, S., Chandler, C., Cheatham, M., Fils, D., Finin, T.W., Ji, P., Jones, M.B., Karima, N., Lehnert, K.A., Mickle, A., Narock, T.W., O'Brien, M., Raymond, L., Shepherd, A., Schildhauer, M., Wiebe, P.: The GeoLink Modular Oceanography Ontology. In: Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P.T., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) The Semantic Web – ISWC 2015 – 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9367, pp. 301–309. Springer (2015)
2. Krisnadhi, A., Rodríguez-Doncel, V., Hitzler, P., Cheatham, M., Karima, N., Amini, R., Coleman, A.: An ontology design pattern for chess games. In: Blomqvist, E., Hitzler, P., Krisnadhi, A., Narock, T., Solanki, M. (eds.) Proceedings of the 6th Workshop on Ontology and Semantic Web Patterns (WOP 2015) co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pensylvania, USA, October 11, 2015. CEUR Workshop Proceedings, vol. 1461. CEUR-WS.org (2015)

3. Reynolds, D. (ed.): The Organization Ontology. W3C Recommendation (16 January 2014), http://www.w3.org/TR/vocab-org/
4. Rodríguez-Doncel, V., Krisnadhi, A.A., Hitzler, P., Cheatham, M., Karima, N., Amini, R.: Pattern-based linked data publication: The linked chess dataset case. In: Hartig, O., Sequeda, J., Hogan, A. (eds.) Proceedings of the 6th International Workshop on Consuming Linked Data co-located with 14th International Semantic Web Conference (ISWC 2105), Bethlehem, Pennsylvania, US, October 12th, 2015. CEUR Workshop Proceedings, vol. 1426. CEUR-WS.org (2015)
5. Zamazal, O., Svátek, V.: Patomat - versatile framework for pattern-based ontology transformation. Computing and Informatics 34(2), 305–336 (2015), http://www.cai.sk/ojs/index.php/cai/article/view/1138