Chapter 1

Modeling With Ontology Design Patterns: Chess Games As a Worked Example

Adila Krisnadhi, Data Semantics Laboratory, Wright State University, Dayton, OH, USA; and Faculty of Computer Science, Universitas Indonesia

Pascal Hitzler, Data Semantics Laboratory, Wright State University, Dayton, OH, USA

1.1. Introduction

In this chapter, we provide a worked example for modeling with ontology design patterns (ODPs), as a way to set the stage for subsequent discussions. It should be noted, though, that there is not a single way how to model with ODPs effectively – the approach we follow here is mostly inspired by our own experiences with modeling at GeoVocamps [5, 9] and with scientists with diverse backgrounds [2, 3, 10, 11, 16, 20]. Another approach for designing ODPs, called the eXtreme Design Methodology, is discussed in another chapter of this book [1, 17]. Roughly, our approach here consists of (i) formulating use case(s) for which the ODP is intended; (ii) modeling key notions identified from the use case descriptions, first done visually, followed by writing the appropriate logical axiomatization; and (iii) checking the resulting ODP to ensure its quality, which can be done via checking the logical consistency of its axioms, inspecting some of its logical consequences (ensuring no unintended consequence is entailed), populating it with sample data to expose shortcomings in modeling, and testing the ODP against the use cases to ensure that they can be adequately answered by the ODP.

In practice, developing an ODP to model a notion in a particular domain typically requires a close collaboration with the domain experts. However, in this chapter, our modeling of "Chess Game" is helped by widely available chess game data in a number of chess game online repositories. Hence, we do not explicilty involve the domain experts.

We assume that the reader is familiar with basic knowledge about OWL and RDF, as conveyed in brief form in an appendix of this book [15], as well as with

Linked Data [4]. For a thorough introduction, see [6]. The chess example itself is based on the papers by the co-authors [13, 19].

1.2. Use Case

Chess is one of the most popular games worldwide, e.g., a 2012 study found that 605 million adults play chess regularly,[1] with millions playing online.[2] This significant interest in the game, and the large number of chess games played online, led to huge repositories of chess games that people can access in the form of Portable Game Notation (PGN) files. This chess game data does not just contain details about the moves of a game, but also about player identities, chess tournament names, and spatiotemporal information relevant to the games.

The use case that shall drive our modeling is that of enhanced searchability of chess games by potentially taking into account additional information from the Web, e.g. using player pages or pages on tournaments or openings on Wikipedia or on chess sites, using location information available through gazeteers or geo-related linked datasets, etc. This perspective precludes, for example, a straightforward conversion of PGN into linked data: In order to be able to merge information from other data sources, it will not suffice to represent players by strings consisting of names, as done in PGN.

Our modeling in fact shall be informed by the desire to remain as general as possible so that it would be possible, and any time later, to add information from additional sources, and such that we do not impose any significant restrictions on the parameters that can be used during the search. This means, it is a use case scenario where it is prudent to apply ontology design patterns and good ontology modeling techniques in order to arrive at an underlying schema which is robust, extendable, and easy to maintain and update, before creating the actual shared data as linked data based on this schema or ontology.

After settling on a use case, it is advisable to formulate a number of competency questions which shall be answerable over the data once it is published. Some typical competency questions are listed below.

(i) Who played against Kasparov in the round 1994 Linares tournament? Did (s)he play as a white or black player?

(ii) What is the first move taken by the black player in the Sicilian Defence opening?

(iii) Find all games in which Bobby Fischer, playing black, lost in the poisoned pawn variation of the Sicilian Defence opening.

(iv) Are there any recorded games using the Grünfeld Defence from before the 20th century?

[1]`http://en.chessbase.com/post/che-redux-how-many-people-play-che-`

[2]`https://www.chess.com/` lists over 13 million members as of February 2016, with tens of thousands concurrently online.

(v) What did Kasparov say about his opponent's first two moves in his commentary about his game against Topalov in the 1999 Tournament in Wijk aan Zee?

(vi) Who was the first non-Russian world champion after Fischer?

(vii) Did Bobby Fischer ever play against a grandmaster in Germany?

(viii) List all world championship games won by forfeit.

1.3. Data Sources and Scoping

The use case definition above of course already needs to be informed by some knowledge about available datasets that can be used for the intended purpose. Once we have a set of competency questions as listed above, we take a somewhat closer look at the possible data sources.

Central to our endeavor is the fact that there exists a de-facto standard for representing chess games, namely the above mentioned PGN format[3] which is supported by many chess programs and onling playing sites, and countless games in PGN format are available for download from the Web. Each game description in a PGN file contains a tag-pair section and a movetext section. The latter describes the actual moves in the game using the Standard Algebraic Notation (SAN), possibly with additional annotation, while the former provides information in the form of name-value pairs expressed as a tag enclosed with square brackets. The tag-pair section contains seven mandatory pieces information, namely the name of tournament or match event, the location of the event, the starting date of the game, the playing round for the game in a tournament, the white player's name, the black player's name, and the result of the game. Additionally, it may also contain other information such as players' Elo ratings, opening variation, alternative starting position, or description of the game conclusion. The PGN format thus makes it possible to record the most relevant information pertaining to a particular game of chess. Figure 1.1 contains a complete example of a simple PGN file.

PGN files will make it possible to address some of the competency questions, e.g., questions (i) through (iv), or even (v) if a PGN file with Kasparov's annotations on this game is available. However, PGN files alone will not make it possible to address competency questions like number (vi), which would require biographical information about players, (vii), which would require more fine-grained information on game locations, or (viii), which would require more fine-grained information about the cause for a game score, i.e., whether it was by check-mate, by resignation, or by forfeit.

Possible sources for such additional information are plentiful. Most convenient, of course, would be to incorporate sources that are already available as linked data, e.g., DBpedia [14] for additional information on players, openings, tournaments, or GeoNames[4] for location-information to address questions such as

[3]http://www.thechessdrum.net/PGN_Reference.txt
[4]http://www.geonames.org/ontology/documentation.html

```
[Event "World Championship 31th-KK1"]
[Site "Moscow"]
[Date "1984.11.23"]
[Round "27"]
[White "Karpov, Anatoly"]
[Black "Kasparov, Gary"]
[Result "1-0"]
[WhiteElo "2705"]
[BlackElo "2715"]
[ECO "D55"]

1.Nf3 d5 2.d4 Nf6 3.c4 e6 4.Nc3 Be7 5.Bg5 h6 6.Bxf6 Bxf6 7.e3 O-O 8.Qc2 c5
9.dxc5 dxc4 10.Bxc4 Qa5 11.O-O Bxc3 12.Qxc3 Qxc3 13.bxc3 Nd7 14.c6 bxc6
15.Rab1 Nb6 16.Be2 c5 17.Rfc1 Bb7 18.Kf1 Bd5 19.Rb5 Nd7 20.Ra5 Rfb8 21.c4 Bc6
22.Ne1 Rb4 23.Bd1 Rb7 24.f3 Rd8 25.Nd3 g5 26.Bb3 Kf8 27.Nxc5 Nxc5 28.Rxc5 Rd6
29.Ke2 Ke7 30.Rd1 Rxd1 31.Kxd1 Kd6 32.Ra5 f5 33.Ke2 h5 34.e4 fxe4 35.fxe4 Bxe4
36.Rxg5 Bf5 37.Ke3 h4 38.Kd4 e5+ 39.Kc3 Bb1 40.a3 Re7 41.Rg4 h3 42.g3 Re8
43.Rg7 Rf8 44.Rxa7 Rf2 45.Kb4 Rxh2 46.c5+ Kc6 47.Ba4+ Kd5 48.Rd7+ Ke4
49.c6 Rb2+ 50.Ka5 Rb8 51.c7 Rc8 52.Kb6 Ke3 53.Bc6 h2 54.g4 Rh8 55.Rd1 Ba2
56.Re1+ Kf4 57.Re4+ Kg3 58.Rxe5 Kxg4 59.Re2  1-0
```

Figure 1.1. Example PGN file.

question (vii). Additional relevant data could e.g. be scraped from chess websites, to potentially further expand search capabilities for games.[5]

Of course, at this stage the scope, in terms of data coverage and search capabilities, needs to be considered. We will take a rather conservative perspective and intend to model essentially what is available from PGN files, in order to republish them as linked data. However at the same time we want to keep in mind that we may want the possibility to expand to other data sources later, i.e., our modeling should already reflect this, such that a broadening of scope is possible without the need to redo the modeling.

1.4. Modeling Key Notions

Since our main focus is on PGN files, it is tempting to let the ontology – and thus the shape of the resulting linked data RDF graph – be informed by the way PGN files encode information. However, this straightforward way of modeling would not account for the possible expansion of scope and thus required data integration which is part of our use case: We have already discussed problems arising, for example, out of identifying players by name-string only, as done in the PGN format. There is also another, perhaps less obvious reason to not go this route: modeling an ontology closely following a data format may make data publication simpler, but at the same time the resulting graph structures may end up being rather not intuitive, i.e., they may be far from capturing human conceptualization of this information, thus making the reuse by third parties much more difficult and costly [19]. We will thus follow a different approach: the PGN format as our

[5]Licensing issues of course need to be reflected [8, 18], but we will ignore this aspect since it is not directly relevant for our modeling example.

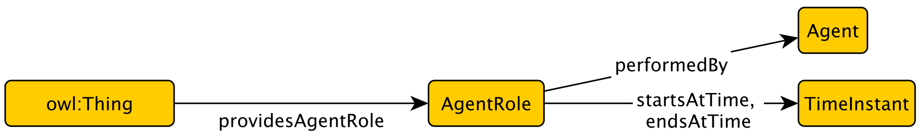

Figure 1.2. The Agent Role ODP.

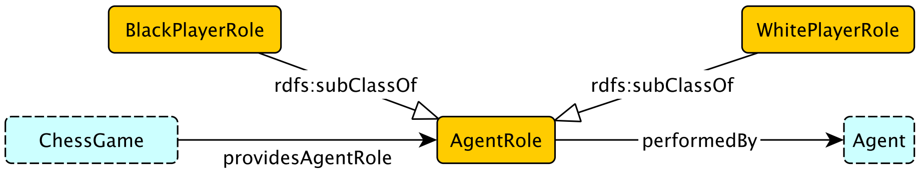

Figure 1.3. Adapted Agent Role ODP for chess player roles.

starting point will merely inform us of key notions that should be captured in our ontology. However, the structure of our ontology will be informed not by the PGN format, but rather by general ontology design patterns which much more faithfully capture human conceptualizations.

In this particular case, based on the required fields of the PGN format discussed earlier, and based on our competency questions, we can identify several key notions which will be in need of modeling, say *chess game*, *move* (or *half-move*[6]), *player*, *opening*, *result*, *commentary*, and *tournament*. In a next step, we look at each of these key notions in turn, and attempt to determine what general pattern it should be modeled after.

Let us start with the notion of *player*. In order to be able to attach additional information to a player, such as the player's ELO score or citizenship information, we need of course a dereferenceable URI. However, in addition to this, we want to borrow from best practices and realize that "being a player" is actually a *role* which an agent can take, e.g., for a certain time period. We thus reuse the very common ontology design pattern for this purpose, which is depicted in Figure 1.2 in exactly the form in which it is used, e.g., in [11]. In this ODP, something (e.g., a chess game) provides roles (such as white player or referee), which an agent (e.g., Magnus Carlsen or Deep Blue) holds for some time interval.

For the chess game, we want to adapt this pattern for our specific case. We prefer to leave most things untouched, while avoiding overgeneralization (e.g., we want to get rid of the owl:Thing type) and while pruning unnecessary details. The resulting pattern is depicted in Figure 1.3. Note in particular that we have removed the temporal extent, which should rather be attached to the chess game itself. We also introduced two subclasses to distinguish the two different player roles, white player and black player. The dashed frames indicate that a more complex entity (a pattern in its own right) could stand in place of the frame:

[6]In chess, one player's turn is considered a *half-move*, while a *move* consists of two half-moves: one by the white player followed by one by the black player.

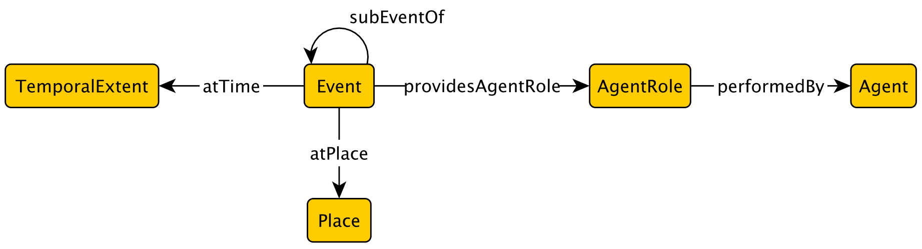

Figure 1.4. The Event ODP.

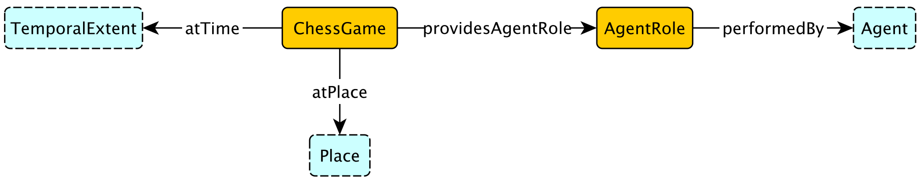

Figure 1.5. Chess game as event.

ChessGame will indeed be modeled below, and Agent could also be expanded, if desired, with a more fine-grained model; or alternatively, an existing Agent ODP could be directly used instead.

In our experience, agent roles occur very often when modeling, i.e. the Agent Role ODP is particularly useful at many occasions.

We now turn our attention to *chess game*. There are different possible perspectives from which one could understand the notion of chess game. It could, e.g., be understood as an abstract artifact similar to a piece of art, with the players being the creators. One could also view it as an event, or alternatively as a record of an event. Each of these perspectives has merit and it is not always straightforward to find the most suitable one; in tricky cases it is in our experience helpful to have a group of people discuss the different viewpoints in light of the competency questions and the use case.

In our specific case, our competency questions indicate that time and space related to a chess game play an identifying role. Furthermore, agents take particular roles (e.g., as players) relative to a chess game. These observations suggest to reuse a generic and rather straightforward *event* pattern, such as the one depicted in Figure 1.4, which is a variant of the form used by the co-authors in the past [11]. It essentially lists only spatial and temporal extent, indicates that events may provide agent roles, and allows an event to have a sub-event.

Like before, we specialize the pattern for our specific purpose, the result of which is depicted in Figure 1.5. Of course we reuse the Agent Role pattern. We do at this stage not address the question how exactly we want to represent temporal and spatial information; others have indeed alrady provided solutions to this, e.g., the OWL Time Ontology [7]. Moreover, the subevent relationship is not employed at this stage.

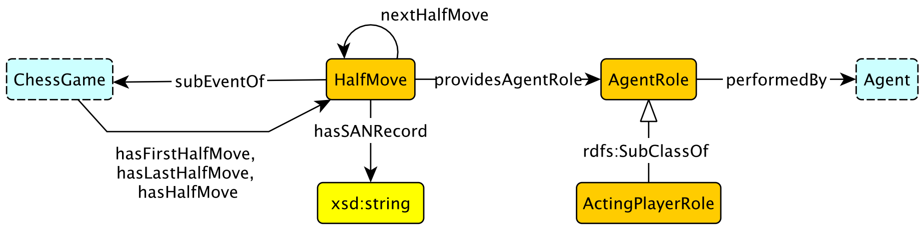

Figure 1.6. Half-moves.

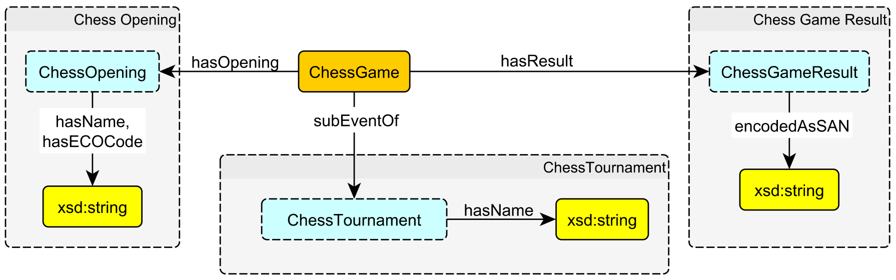

Figure 1.7. Three *stubs*, for ChessOpening, ChessTournament, and ChessGameResult.

Let us next look at *move*, or rather at *half-move* which is the technical term in chess for one player's action. Since we have settled for viewing chess games as events, half-moves are naturally subevents. This also aligns with the idea that half-moves may have their own temporal extent, which may be recorded (and very relevant for the game outcome) by means of the chess clock used during the game. Half-moves provide an agent role, namely that of acting player, and each half-move has a record, which can be given in Standard Algebraic Notation (SAN),[7] the undisputed norm for recording half-moves which is also incorporated in the PGN format. Each half-move furthermore refers to the next half-move, so that the sequence is conveyed,[8] and we need to identify first and last half-moves of the game.[9] The resulting pattern is depicted in Figure 1.6.

Let us next look at *tournament*. Some of our competency questions indicate that we may eventually want a fine-grained modeling of chess tournaments. However, let us say that we agree to not model this aspect in detail at this stage, but that we rather intend to provide the possibility to make this extension later, without having to change anything already modeled.

At this stage, say we want only the ability to record the name of a tournament as a string. For extendability we also require a *hook*, in form of the central node for a chess tournament, to which additional information can later be attached.

[7]`http://www.fide.com/component/handbook/?id=171&view=article` Appendix C

[8]We can also understand this as a derivative of a generic *sequence* ODP of course.

[9]Some of this, e.g. which player's turn it is, is really redundant information, but it seems convenient to nevertheless record it explicitly.

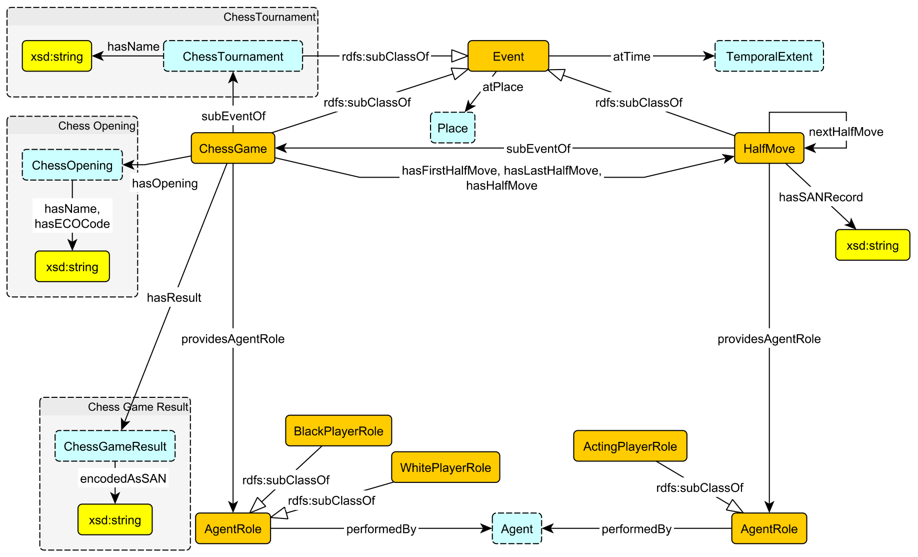

Figure 1.8. Putting the previously developed pieces together. Orange boxes are atomic classes. Blue boxes with dashed frames are classes with details to be developed or hidden in a separate pattern/ontology. Grouping boxes are sub-patterns that can be modeled separately.

This can be realized best with a *stub*, as depicted in the lower part of Figure 1.7. Note that this way of modeling chess tournaments is rather uncommital regarding future extensions. In contrast, if we had directly linked chess games to the string containing the name of the tournament, we would not have provided an obvious hook for the future extension.

Let us currently decide to do the same with *opening*, however adding a way to record the ECO (Encyclopedia of Chess Openings) code for the opening. And let's also do the same for now with the result of a chess game, recorded using SAN, and again this could be extended later to add the ability to answer competency questions like number viii.

The only key notion missing from our initial list is *commentary*, but before we do this, it seems time to start putting things together.

1.5. Putting Things Together

It is straightforward to assemble the pieces we already developed, and the result is depicted in Figure 1.8. Note that we attached the temporal and spatial extents to Event: due to the subclassing of ChessGame (and ChessTournament) this means that temporal and spatial extents can also be specified for these. The picture shows two nodes for AgentRole simply because it makes the graph look nicer, and because it is then easier to see which agent roles are available in each case.

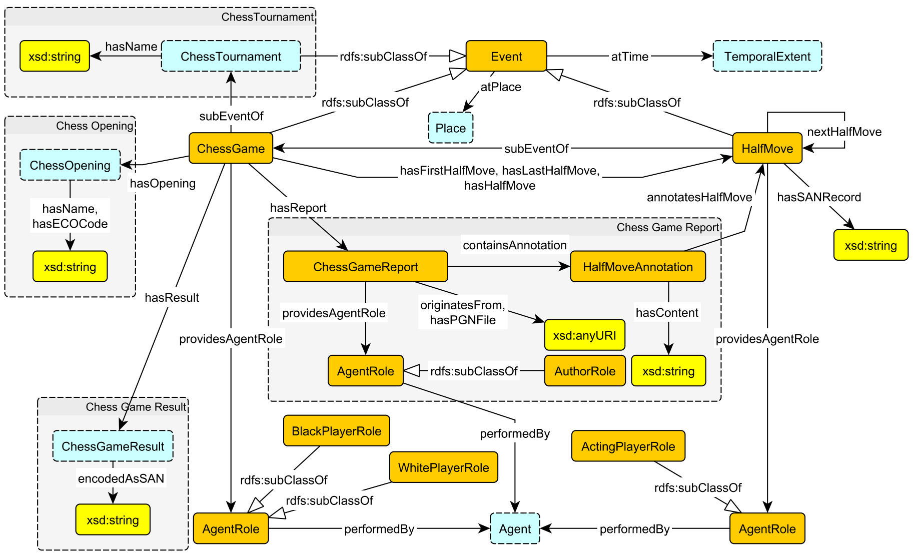

Figure 1.9. Depiction of the complete model.

Note also that the previously developed pieces can essentially be understood as a type of *modules* [11], each derived from ODPs, which are put together to form the ontology outline.

Now let's go back to *commentary*, which is arguably a bit more tricky do deal with. A commentary in chess usually consists of a set of comments, each of which is attached to a half-move. A comment thus comes attached to the move sequence, e.g. within a PGN file. We thus may have different commentaries for one and the same game, just as we may have different PGN files for the same game, which may or may not differ in some respects. There are different ways how this can be viewed. One could consider the PGN files to be *manifestations* of a game – in the sense in which, say, two different hardcopies of a book are two concrete manifestations of one and the same book. This perspective was taken in [13, 19]. A different perspective would be to consider the PGNs, and also commentaries, as *reports* on the chess game events. We will take this latter perspective herein.

Once we have settled on this perspective, the remaining pieces fall into place relatively easily. Reports provide author roles (modeled using the Agent Role ODP), and carry Annotations (i.e., comments) which are attached to half-moves and have strings as content. We furthermore want to record that a report may originate from some URI, or has an associated PGN file we would want to point to. The resulting ontology (we could also call it *Chess Game ODP*) is depicted in Figure 1.9.

1.6. Axiomatization

The diagram of course is not the ontology, it is merely a preliminary depiction of main relationships, which is appropriate at this stage because so far the discussion has not dived into details. We have so far avoided issues such as specifying that a chess game can only have one first half move, for example, and while they may be intuitively understood by a human reader familiar with chess, they may not be so clear for others, and in any case we would like to put as many of these conditions into the formal model as possible.

For this purpose, we describe the pattern using *axioms* in some logic. Below, we use description logics, which in this case can be translated seamlessly into OWL. From the earlier sections, we know that the chess game pattern consists of the following components:

1. the (re)use of the Agent Role pattern, which reoccurs when modeling players and authors of chess game reports;
2. the (re)use of the Event pattern to model chess game, half move, and chess tournament;
3. stubs for representing chess opening and chess game result; and
4. simple pattern for chess game report.

One way to realize the reuse of Agent Role and Event patterns in practice is by importing the OWL serialization of both patterns into the OWL serialization of the Chess Game pattern. This means that all axioms and ontological commitments imposed by both Agent Role and Event patterns will be employed by the Chess Game pattern. In general, however, the use cases and modeling requirements may necessitate some adjustments and modifications in the axioms during pattern reuse, which cannot be done solely through OWL import statements. To simplify our discussion in this chapter, the Agent Role and Event patterns are modeled so that it is not necessary to modify them when importing into the Chess Game pattern. We begin with the Agent Role pattern, follow it with the Event pattern, and conclude the discussion with the Chess Game pattern itself.

1.6.1. Agent Role

Axioms for the Agent Role pattern are depicted in Figure 1.2. The axioms for Agent Role pattern are given in Figure 1.10. Further discussion on the Agent Role pattern can be found in Chapter 16. In the meantime, we provide a brief explanation of the axioms here.

Axiom (1.1) and (1.2) asserts that every agent role is always performed by exactly one agent, starts at exactly one time instant, and ends at exactly one time instant. Note that these axioms only states the existence of an agent, starting time instant, and ending time instant for an agent role; it does not imply that they have to be *known* nor they have to be included in the data.

The other axioms in the pattern, except the last one, capture the domain and range of each property in the pattern. Roughly, for every triple of class A, (object

$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{performedBy.Agent}) \tag{1.1}$$
$$\mathsf{AgentRole} \sqsubseteq (=1\ \mathsf{startsAtTime.TimeInstant}) \sqcap (=1\ \mathsf{endsAtTime.TimeInstant}) \tag{1.2}$$
$$\exists \mathsf{performedBy.Agent} \sqsubseteq \mathsf{AgentRole} \tag{1.3}$$
$$\exists \mathsf{startsAtTime.TimeInstant} \sqcup \exists \mathsf{endsAtTime.TimeInstant} \sqsubseteq \mathsf{AgentRole} \tag{1.4}$$
$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{performedBy.Agent} \tag{1.5}$$
$$\mathsf{AgentRole} \sqsubseteq \forall \mathsf{startsAtTime.TimeInstant} \sqcap \forall \mathsf{endsAtTime.TimeInstant} \tag{1.6}$$
$$\top \sqsubseteq \forall \mathsf{providesAgentRole.AgentRole} \tag{1.7}$$
$$\mathtt{DisjointClasses}(\mathsf{AgentRole}, \mathsf{Agent}, \mathsf{TimeInstant}) \tag{1.8}$$

Figure 1.10. Axiomatization for Agent Role ODP

or data) property P, and class or data type B, we want to express that the domain of P is A and its range is B. We achieve this using the so-called scoped domain and range restrictions, which we shall mostly follow throughout this chapter. We begin with axiom (1.3), which states that if something is performed by an agent, then it is an agent role. This is a form of scoped domain restriction for the property performedBy where the domain is AgentRole and the scope is Agent. More precisely, given two individuals x and y such that x is performed by y, we infer that x is an agent role (i.e., belongs to the domain) if y is an agent (i.e., is known to belong to the scope). If y is not known to be an agent, then we do not infer that x is an agent role. This differs from directly asserting that the domain of performedBy is AgentRole. The latter means that if x is performed by y, then x is an agent role regardless whether y is an agent or not. Axiom (1.4) is likewise scoped domain restriction for both startsAtTime and endsAtTime.[10]

Axiom (1.5) is a scoped range restriction for the property performedBy with Agent as the range and AgentRole as the corresponding scope. This axiom states that for two individuals x and y, given two individuals x and y such that x is performed by y, if x is an agent role, then we infer that y is an agent. If x is not known to be an agent role, we do not infer y to be an agent here. Axiom (1.6) expresses the scoped range restriction for both startsAtTime and endsAtTime property.[11].

Next, anything may provide an agent role, i.e., it provides zero or more agent roles. For this requirement, we do not write an axiom involving a number restriction because for every property R and class C, $(\geq 0\ R.C)$ is equivalent to $\top$. Instead, we simply express the domain and range restrictions for the property. Specific for the Agent Role pattern, the domain of providesAgentRole property is owl:Thing, so there is no need to explicitly state a domain restriction for this property, and we simply assert the (unscoped) range restriction according to axiom (1.7).

[10] $C \sqcup D \sqsubseteq E$ is equivalent to two axioms $C \sqsubseteq E$ and $D \sqsubseteq E$, as an easy consequence of the OWL 2 semantics.

[11] Again, OWL 2 semantics implies that $C \sqsubseteq D \sqcap E$ is equivalent to two axioms $C \sqsubseteq D$ and $C \sqsubseteq E$.

$$\begin{aligned}
&\mathsf{import}\colon (1.1), (1.2), (1.3), (1.4), (1.5), (1.6), (1.7), (1.8) \\
&\mathsf{Event} \sqsubseteq \exists \mathsf{atPlace}.\mathsf{Place} \sqcap \exists \mathsf{atTime}.\mathsf{TemporalExtent} && (1.9) \\
&\exists \mathsf{atTime}.\mathsf{TemporalExtent} \sqcup \exists \mathsf{atPlace}.\mathsf{Place} \sqcup \exists \mathsf{subEventOf}.\mathsf{Event} \sqsubseteq \mathsf{Event} && (1.10) \\
&\mathsf{Event} \sqsubseteq \forall \mathsf{atTime}.\mathsf{TemporalExtent} \sqcap \forall \mathsf{atPlace}.\mathsf{Place} \sqcap \forall \mathsf{subEventOf}.\mathsf{Event} && (1.11) \\
&\mathtt{DisjointClasses}(\mathsf{Event}, \mathsf{TemporalExtent}, \mathsf{Place}, \mathsf{AgentRole}, \mathsf{Agent}) && (1.12)
\end{aligned}$$

Figure 1.11. Axiomatization for Event ODP

Finally, axiom (1.8) asserts that every pair of classes amongst AgentRole, Agent, and TimeInstant are pairwise disjoint. That is, for example, nothing can be simultaneously an agent and an agent role, or nothing can be both an agent role and a time instant.

1.6.2. Event

Axioms for the Event pattern are depicted in Figure 1.11. Note that the Event pattern imports the Agent Role pattern.

Axiom (1.9) asserts that an event occurs at some temporal extent and at some place. Again, the temporal extent and place may not necessarily be known.

Axiom (1.10) expresses scoped domain restrictions for the property atTime, atPlace, and subEventOf where the domain of all three properties is Event while the scope are TemporalExtent, Place, and Event, respectively. Meanwhile, axiom (1.11) expresses the corresponding scoped range restrictions where the range are respectively TemporalExtent, Place, and Event, while the class Event is the scope for range restrictions for all those three properties. Since the Agent Role pattern is imported to the Event pattern, we directly use their axioms and do not repeat their assertion here, including domain and range restrictions for providesAgentRole and performedBy properity.

Finally, axiom (1.12) asserts any two different classes amongst Event, TemporalExtent, Place, AgentRole, and Agent are pairwise disjoint.

1.6.3. Chess Game

Axioms for Chess Game pattern are given in Figure 1.12 and 1.13. Note that axioms from the Agent Role and Event pattern are imported into the Chess Game pattern.

Axiom (1.13) asserts that a chess game is an event and it provides a black player role and a white player role. Since every chess game is an event, axioms for event apply and thus, a chess game happens at some time and some place. Next, every black player role and white player role themselves are agent roles, and furthermore, there exists exactly one chess game that provides them. This was asserted in axiom (1.14).

$$\text{import: } (1.1), (1.2), (1.3), (1.4), (1.5), (1.6), (1.7), (1.8), (1.9), (1.10), (1.11), (1.12)$$

$$\mathsf{ChessGame} \sqsubseteq \mathsf{Event} \sqcap \exists \mathsf{pAR.BlackPlayerRole} \sqcap \exists \mathsf{pAR.WhitePlayerRole} \tag{1.13}$$

$$\mathsf{BlackPlayerRole} \sqcup \mathsf{WhitePlayerRole} \sqsubseteq \mathsf{AgentRole} \sqcap ({=}1\ \mathsf{pAR}^{-}.\mathsf{ChessGame}) \tag{1.14}$$

$$\mathsf{ChessGame} \sqsubseteq ({=}1\ \mathsf{hasFirstHalfMove.HalfMove}) \tag{1.15}$$

$$\mathsf{ChessGame} \sqsubseteq ({=}1\ \mathsf{hasLastHalfMove.HalfMove}) \tag{1.16}$$

$$\mathsf{hasHalfMove} \sqsubseteq \mathsf{subEventOf}^{-} \tag{1.17}$$

$$\mathsf{hasFirstHalfMove} \sqsubseteq \mathsf{hasHalfMove} \tag{1.18}$$

$$\mathsf{hasLastHalfMove} \sqsubseteq \mathsf{hasHalfMove} \tag{1.19}$$

$$\exists \mathsf{subEventOf.ChessTournament} \sqcup \exists \mathsf{hasOpening.ChessOpening} \sqsubseteq \mathsf{ChessGame} \tag{1.20}$$

$$\exists \mathsf{hasResult.ChessGameResult} \sqcup \exists \mathsf{hasReport.ChessGameReport} \sqsubseteq \mathsf{ChessGame} \tag{1.21}$$

$$\mathsf{ChessGame} \sqsubseteq \forall \mathsf{subEventOf.ChessTournament} \sqcap \forall \mathsf{hasOpening.ChessOpening} \tag{1.22}$$

$$\mathsf{ChessGame} \sqsubseteq \forall \mathsf{hasResult.ChessGameResult} \sqcap \forall \mathsf{hasReport.ChessGameReport} \tag{1.23}$$

$$\mathsf{HalfMove} \sqsubseteq \mathsf{Event} \sqcap \exists \mathsf{pAR.ActingPlayerRole} \sqcap ({=}1\ \mathsf{hasHalfMove}^{-}.\mathsf{ChessGame}) \tag{1.24}$$

$$\mathsf{ActingPlayerRole} \sqsubseteq \mathsf{AgentRole} \sqcap ({=}1\ \mathsf{pAR}^{-}.\mathsf{HalfMove}) \tag{1.25}$$

$$\mathsf{HalfMove} \sqsubseteq ({\leq}1\ \mathsf{nextHalfMove.HalfMove}) \sqcap \neg \exists \mathsf{nextHalfMove.Self} \tag{1.26}$$

$$\exists \mathsf{subEventOf.ChessGame} \sqcup \exists \mathsf{nextHalfMove.HalfMove} \sqsubseteq \mathsf{HalfMove} \tag{1.27}$$

$$\exists \mathsf{hasSANRecord.xsd{:}string} \sqsubseteq \mathsf{HalfMove} \tag{1.28}$$

$$\mathsf{HalfMove} \sqsubseteq \forall \mathsf{subEventOf.ChessGame} \sqcap \forall \mathsf{nextHalfMove.HalfMove} \tag{1.29}$$

$$\mathsf{HalfMove} \sqsubseteq \forall \mathsf{hasSANRecord.xsd{:}string} \tag{1.30}$$

Figure 1.12. Chess Game pattern (part 1) with some axioms imported from the Agent Role (Fig. 1.10) and Event pattern (Fig. 1.11); pAR stands for providesAgentRole. The rest of the axioms can be found in Fig. 1.13

Axiom (1.15) and (1.16) state that every chess game has exactly one half move in the beginning (first half move) and one at the end (last half move). Axiom (1.17) indicates that the relation between half-move and chess game is that of a sub-event. This is important in case one may want to impose further axioms coming from an event pattern, e.g., one could specify that sub-events of an event must fall within the spatial and temporal scope of the larger event. We have not done this here. Furthermore, axiom (1.18) and (1.19) assert that first half-move and last half-move are actually half-moves.

Next, a chess game may be a sub-event of a chess tournament, may have a chess opening, and may have a chess game result, and may have a chess game report. The reader may notice that stating that a chess game may be a sub-event of a chess tournament is actually equivalent to saying that a chess game is a sub-event of zero or more chess tournaments. This can be modeled as the axiom $\mathsf{ChessGame} \sqsubseteq ({\geq}0\ \mathsf{subEventOf.ChessTournament})$. As discussed earlier when discussing the Agent Role pattern, the right-hand side of this axiom is equivalent to $\top$, hence the axiom is equivalent to asserting $\mathsf{ChessGame} \sqsubseteq \top$, which gives us

$$\begin{aligned}
\mathsf{ChessTournament} &\sqsubseteq \mathsf{Event} && (1.31)\\
\mathsf{ChessTournament} \sqcup \mathsf{ChessOpening} &\sqsubseteq \forall \mathsf{hasName.xsd{:}string} && (1.32)\\
\exists \mathsf{hasECOCode.xsd{:}string} &\sqsubseteq \mathsf{ChessOpening} && (1.33)\\
\mathsf{ChessOpening} &\sqsubseteq \forall \mathsf{hasECOCode.xsd{:}string} && (1.34)\\
\exists \mathsf{encodedAsSAN.xsd{:}string} &\sqsubseteq \mathsf{ChessGameResult} && (1.35)\\
\mathsf{ChessGameResult} &\sqsubseteq \forall \mathsf{encodedAsSAN.xsd{:}string} && (1.36)
\end{aligned}$$

$$\begin{aligned}
\mathsf{ChessGameReport} &\sqsubseteq \exists \mathsf{pAR.AuthorRole} && (1.37)\\
\mathsf{AuthorRole} &\sqsubseteq \mathsf{AgentRole} \sqcap ({=}1\ \mathsf{pAR}^{-}.\mathsf{ChessGameReport}) && (1.38)\\
\exists \mathsf{containsAnnotation.HalfMoveAnnotation} &\sqsubseteq \mathsf{ChessGameReport} && (1.39)\\
\mathsf{ChessGameReport} &\sqsubseteq \forall \mathsf{containsAnnotation.HalfMoveAnnotation} && (1.40)\\
\exists \mathsf{originatesFrom.xsd{:}anyURI} \sqcup \exists \mathsf{hasPGNFile.xsd{:}anyURI} &\sqsubseteq \mathsf{ChessGameReport} && (1.41)\\
\mathsf{ChessGameReport} &\sqsubseteq \forall \mathsf{originatesFrom.xsd{:}anyURI} \sqcap \forall \mathsf{hasPGNFile.xsd{:}anyURI} && (1.42)
\end{aligned}$$

$$\begin{aligned}
\mathsf{HalfMoveAnnotation} &\sqsubseteq ({=}1\ \mathsf{annotatesHalfMove.HalfMove}) && (1.43)\\
\mathsf{HalfMoveAnnotation} &\sqsubseteq \exists \mathsf{hasContent.xsd{:}string} && (1.44)\\
\exists \mathsf{hasContent.xsd{:}string} &\sqsubseteq \mathsf{HalfMoveAnnotation} && (1.45)\\
\mathsf{HalfMoveAnnotation} &\sqsubseteq \forall \mathsf{hasContent.xsd{:}string} && (1.46)
\end{aligned}$$

$$\begin{aligned}
&\texttt{DisjointClasses}(\mathsf{AgentRole}, \mathsf{Agent}, \mathsf{Event}, \mathsf{Place}, \mathsf{TemporalExtent},\\
&\qquad \mathsf{ChessOpening}, \mathsf{ChessGameResult}, \mathsf{ChessGameReport}, && (1.47)\\
&\qquad \mathsf{HalfMoveAnnotation})\\
&\texttt{DisjointClasses}(\mathsf{ChessGame}, \mathsf{ChessTournament}, \mathsf{HalfMove}) && (1.48)\\
&\texttt{DisjointClasses}(\mathsf{BlackPlayerRole}, \mathsf{WhitePlayerRole}, \mathsf{ActingPlayerRole},\\
&\qquad \mathsf{AuthorRole}) && (1.49)
\end{aligned}$$

Figure 1.13. Chess Game pattern (part 2; continued from Fig. 1.12); pAR stands for providesAgentRole.

nothing in terms of inference because it always holds by definition of the semantics of OWL. We thus do not include this in the axiomatization. The same thing holds for the relationships with chess opening, chess game result, and chess game report.

Nevertheless, the corresponding object and data properties that capture such relationships shall still be included in the axiomatization in the form of domain and range restrictions. Axiom (1.20) and (1.21) simultaneously express scoped domain restrictions for subEventOf, hasOpening, hasResult, and hasReport. The class ChessGame is the domain of all four properties, while the scope are respectively ChessTournament, ChessOpening, ChessGameResult, and ChessGameReport.

Axiom (1.22) and (1.23) assert scoped range restrictions for subEventOf, hasOpening, hasResult, and hasReport. Here, the range are ChessTournament, ChessOpening, ChessGameResult, and ChessGameReport, respectively, while the class ChessGame acts as the scope for range restrictions of those four properties.

Axiom (1.24) states that every half-move is an event that provides an acting player role and furthermore, it must be a half-move of exactly one chess game. Here, an acting player role refers to the role of the chess player that performs the half-move. Also, axiom (1.25) expresses that an acting player role is an agent role, which is provided by exactly one half-move.

Axiom (1.26) asserts that a half-move cannot be followed by more than one half-move and cannot precede itself. This ensures that using the property nextHalfMove, we obtain a linear chain of half-moves.

Axiom (1.27) simultaneously expresses scoped domain restrictions for the property subEventOf and nextHalfMove where the domain is HalfMove for both properties and the scope are ChessGame and HalfMove, respectively. Notice that axiom (1.27) is a different domain restriction for subEventOf than the one given in axiom (1.20): the former asserts the domain of subEventOf to be the class HalfMove, provided that the scope is ChessGame, whereas the latter asserts the domain of subEventOf to be the class ChessGame if the scope is ChessTournament. These two restrictions can be asserted together because we use scoped domain restriction, i.e., this cannot be achieved using only non-scoped domain restrictions. In addition, axiom (1.28) is a scoped domain restriction for the data property hasSANRecord with HalfMove as the domain and xsd:string as the scope.

Axiom (1.29) asserts scoped range restrictions for the property subEventOf and nextHalfMove simultaneously where the scope is HalfMove for both properties and the range are ChessGame and HalfMove, respectively. Similar to the case for scoped domain restrictions explained earlier, this axiom asserts the second range restriction for subEventOf. Also, axiom (1.30) is a scoped range restriction for the data property hasSANRecord with xsd:string as the range and HalfMove as the scope.

Next, chess tournaments are events, as asserted in axiom (1.31). Axiom (1.32) asserts a scoped range restriction for the data property hasName with xsd:string as the range and the class union ChessTournament $\sqcup$ ChessOpening as the scope. Next, axiom (1.33) asserts a scoped domain restriction for hasECOCode with ChessOpening as the domain and xsd:string as the scope, while axiom (1.34) expresses a scoped range restriction for the same property with xsd:string as the range and the class ChessOpening as the scope. Axiom (1.35) is a scoped domain restriction for encodedAsSAN with ChessGameResult as the domain and xsd:string as the scope, and axiom (1.36) is a scoped range restriction for the same property with xsd:string as the range and ChessGameResult as the scope.

The next few axioms are intended for the chess game report part of the pattern. Recall that the relationship between chess games and chess game reports is provided by the property hasReport whose domain and range were axiomatized in axiom (1.21) and (1.23). Now, a chess game report always provides at least one author role as given by axiom (1.37). Meanwhile, an author role is an agent role and it is provided by exactly one chess game report as asserted by axiom (1.38).

Axiom (1.39) asserts a scoped domain restriction for containsAnnotation property with ChessGameReport as the domain and HalfMoveAnnotation as the scope. Axiom (1.40) asserts a scoped range restriction for the same property with HalfMoveAnnotation as the range and ChessGameReport as the scope.

Axiom (1.41) asserts scoped domain restrictions for both originatesFrom and hasPGNFile, respectively. Here, the domain for both is ChessGameReport while the scope for both is xsd:anyURI. Axiom (1.42) asserts scoped range restrictions for both data properties with xsd:anyURI as the range and ChessGameReport as the scope for both.

Axiom (1.43) states that every half-move annotation annotates exactly one half-move. Axiom (1.44) asserts that every half-move annotation must have some string as content. Axiom (1.46) expresses a scoped domain restriction with HalfMoveAnnotation as the domain and xsd:string as the scope, while axiom (1.46) expresses a scoped range restriction for hasContent with xsd:string as the range and HalfMoveAnnotation as the scope.

Finally, axiom (1.47), (1.48), and (1.49) indicate pairwise disjointness between classes. The first one asserts that any two classes from AgentRole, Agent, Event, Place, TemporalExtent, ChessOpening, ChessGameResult, ChessGameReport, and HalfMoveAnnotation are pairwise disjoint. The second asserts pairwise disjointness between subclasses of Event, while the third asserts pairwise disjointness between subclasses of AgentRole in the Chess Game pattern.

1.7. Final Assessment

Quality assurance regarding the model can now be done the usual way, e.g., it would be prudent to check the set of axioms for consistency, and compute some logical consequences and inspect them as a type of sanity check. And of course, the model should be populated using real data, which sometimes exposes shortcomings. An example of a step-by-step process to populate the model with linked data can be found in a separate chapter of this book [12].

We will wrap up this chapter by returning to the competency questions stated earlier. Question (i) is answerable if the tournament can be identified, which may even be possible with the stub given. Question (ii) again is answerable by identifying the opening using name or ECO code, and then navigating to the first black half-move. Question (iii) again is identifiable e.g. by using name or ECO code for this variation. Question (iv) is possible by looking at the temporal extent given for games or their associated tournaments. Question (v) is answerable if the comment has been captured in some of the source data. Question (vi) is currently not answerable, but could be answerable by an appropriate extension of the Agent pattern in the model, which could, e.g., encompass nationalities. Question (vii) is answerable if player roles be enhanced by stating grandmaster status, and if enhanced location information is provided for tournaments and games. Question (viii) is currently not answerable, but our game result stub provides a hook for indicating whether a win was by forfeit, i.e. our model can also be extended in this direction.

Acknowledgements. This work was partially supported by the National Science Foundation under award 1440202 *EarthCube Building Blocks: Collaborative Proposal: GeoLink – Leveraging Semantics and Linked Data for Data Sharing and Discovery in the Geosciences.*

Bibliography

[1] E. Blomqvist, K. Hammar, and V. Presutti. Engineering ontologies with patterns – the eXtreme design methodology. In P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors, *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, Studies on the Semantic Web. IOS Press, 2016. In this volume.

[2] D. Carral. An ontology design pattern for detector final states. In P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors, *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, Studies on the Semantic Web. IOS Press, 2016. In this volume.

[3] D. Carral, M. Cheatham, S. Dallmeier-Tiessen, P. Herterich, M. D. Hildreth, P. Hitzler, A. Krisnadhi, K. Lassila-Perini, E. Sexton-Kennedy, C. Vardeman, and G. Watts. An ontology design pattern for particle physics analysis. In E. Blomqvist, P. Hitzler, A. Krisnadhi, T. Narock, and M. Solanki, editors, *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.*, volume 1461 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[4] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.

[5] P. Hitzler, K. Janowicz, and A. A. Krisnadhi. Ontology modeling with domain experts: The GeoVocamp experience. In C. d'Amato, F. Lécué, R. Mutharaju, T. Narock, and F. Wirth, editors, *Proceedings of the 1st International Diversity++ Workshop co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pennsylvania, USA, October 12, 2015.*, volume 1501 of *CEUR Workshop Proceedings*, pages 31–36. CEUR-WS.org, 2015.

[6] P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[7] J. R. Hobbs and F. Pan, editors. *Time Ontology in OWL*. W3C Working Draft, 27 September 2006. Available from http://www.w3.org/TR/owl-time.

[8] P. Jain, P. Hitzler, K. Janowicz, and C. Venkatramani. There's no money in linked data. Available from `http://daselab.org/sites/default/files/No-Money-LOD-TechReport2013.pdf`, May 2013.

[9] K. Janowicz. Modeling ontology design patterns with domain experts – a view from the trenches. In P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors, *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, Studies on the Semantic Web. IOS Press, 2016. In this volume.

[10] A. Krisnadhi. *Ontology Pattern-Based Data Integration*. PhD thesis, Wright State University, 2015.

[11] A. Krisnadhi, Y. Hu, K. Janowicz, P. Hitzler, R. A. Arko, S. Carbotte, C. Chandler, M. Cheatham, D. Fils, T. W. Finin, P. Ji, M. B. Jones, N. Karima, K. Lehnert, A. Mickle, T. W. Narock, M. O'Brien, L. Raymond, A. Shepherd, M. Schildhauer, and P. Wiebe. The GeoLink Mod-

ular Oceanography Ontology. In M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, editors, *The Semantic Web – ISWC 2015 – 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 301–309. Springer, 2015.

[12] A. Krisnadhi, N. Karima, P. Hitzler, R. Amini, V. Rodríguez-Doncel, and K. Janowicz. Ontology design patterns for linked data publishing. In P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors, *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, Studies on the Semantic Web. IOS Press, 2016. In this volume.

[13] A. Krisnadhi, V. Rodríguez-Doncel, P. Hitzler, M. Cheatham, N. Karima, R. Amini, and A. Coleman. An ontology design pattern for chess games. In E. Blomqvist, P. Hitzler, A. Krisnadhi, T. Narock, and M. Solanki, editors, *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.*, volume 1461 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[14] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia – A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[15] F. Maier. A primer on RDF and OWL. In P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, and V. Presutti, editors, *Ontology Engineering with Ontology Design Patterns: Foundations and Applications*, Studies on the Semantic Web. IOS Press, 2016. In this volume.

[16] P. O'Brien, D. Carral, J. Mixter, and P. Hitzler. An ontology design pattern for data integration in the library domain. In E. Blomqvist, P. Hitzler, A. Krisnadhi, T. Narock, and M. Solanki, editors, *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.*, volume 1461 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[17] V. Presutti et al. eXtreme Design with content ontology design patterns. In E. Blomqvist et al., editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009), Washington D.C., USA, 25 October 2009.*, volume 516 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[18] V. Rodríguez-Doncel, A. Gómez-Pérez, and N. Mihindukulasooriya. Rights declaration in linked data. In O. Hartig, J. Sequeda, A. Hogan, and T. Matsutsuka, editors, *Proceedings of the Fourth International Workshop on Consuming Linked Data, COLD 2013, Sydney, Australia, October 22, 2013*, volume 1034 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[19] V. Rodríguez-Doncel, A. A. Krisnadhi, P. Hitzler, M. Cheatham, N. Karima, and R. Amini. Pattern-based linked data publication: The linked chess dataset case. In O. Hartig, J. Sequeda, and A. Hogan, editors, *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., volume 1426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[20] B. Yan, Y. Hu, B. Kuczenski, K. Janowicz, A. Ballatore, A. A. Krisnadhi, Y. Ju, P. Hitzler, S. Suh, and W. Ingwersen. An ontology for specifying spatiotemporal scopes in life cycle assessment. In C. d'Amato, F. Lécué, R. Mutharaju, T. Narock, and F. Wirth, editors, *Proceedings of the 1st International Diversity++ Workshop co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pennsylvania, USA, October 12, 2015.*, volume 1501 of *CEUR Workshop Proceedings*, pages 25–30. CEUR-WS.org, 2015.