Ontology Design Patterns for Representing Context in Ontologies using Aspect Orientation *

Ralph Schäfermeier[1,2][0000−0002−4349−6726], Adrian Paschke[2,3][000−0003−3156−9040], and Heinrich Herre[1]

[1] Institute for Medical Informatics, Statistics and Epidemiology, Leipzig University, {ralph.schaefermeier,heinrich.herre}@imise.uni-leipzig.de
[2] Fraunhofer FOKUS, Berlin, Germany
[3] Freie Universitaet Berlin, Berlin, Germany, paschke@inf.fu-berlin

Abstract. Context-sensitive knowledge is ubiquitous. Knowledge may, for example, change over time and from location to location, or it may be dependent on an observer's background knowledge, belief, or opinion. We observe that specifying context leads to recurring modeling problems. For example, the modeling of temporal context involves the specification of one or several time intervals and attaching them to the axioms or facts that are supposed to be valid during that time. The formalism for specifying the temporal context may, however, vary. As a solution, we present a catalog of parametrizable ontology design patterns (ODPs) specific to the problem of modeling context. We base the patterns on the aspect-oriented extension of OWL 2, because it allows nesting of contexts and the usage of the entirety of OWL 2 (DL) as the context description language. We evaluate the adequacy and usefulness of the approach within a real-life research project about anatomical structure recognition in 3D endoscopic imaging, in which a highly contextualized mapping ontology between the qualitative arrangement of 3D shapes, anatomical structures, and surgical situations is developed. We can show that the use of a context specification formalism leads to an adequate representation of the domain at hand and that the proposed context ODPs facilitate the modeling of contextualized knowledge. Nevertheless, the ODPs are sufficiently abstract and general to be reused in different domains and application scenarios.

Keywords: Ontology Design Pattern · Context · OWL · Description Logics · Modal Logics · Aspect-Oriented Ontology Development

* This work has been partially supported by the German Federal Ministry of Education and Research (BMBF) by a grant for the COMPASS project under the "KMU Innovativ" program and the Wachstumskern-Qurator project funded under the BMBF Innovation Initiative for the New German Laender - Unternehmen Region.

1 Introduction

Context-sensitive knowledge is ubiquitous. Knowledge may, for example, change over time and from location to location, or it may be dependent on an observer's background knowledge, belief, or opinion. Knowledge representation formalisms accommodating context can be roughly divided into (i) those that provide means for contextualizing knowledge implicitly (by providing similar concepts, such as entities with a temporal extension or roles) and (ii) those that treat concept as first-class citizens and directly include the concept of context in their language. Focusing on the second kind of approach, we observe that specifying context leads to recurring modeling problems. For example, the modeling of temporal context involves the specification of one or several time intervals and attaching them to the axioms or facts that are supposed to be valid during that time. The formalism for specifying the temporal context may, however, vary.

As a solution, we present a catalog of parametrizable ontology design patterns (ODPs) specific to the problem of modeling context. We base the patterns on the aspect-oriented extension of OWL 2, because it allows nesting of contexts and the usage of the entirety of OWL 2 (DL) as the context description language. We can show that the use of a context specification formalism leads to an adequate representation of the domain at hand and that the proposed context ODPs facilitate the modeling of the context knowledge.

The remainder of the paper is structured as follows: In Section 2 we give an introductional overview of the problem of context representation and different kinds of solutions. We then introduce Aspect OWL, an extension to OWL 2 for representing context in OWL and briefly elaborate on the principles it is built upon. In Section 3 we discuss a set of recurring representation problems involving different kinds of context-sensitive knowledge and present an Ontology Design Pattern for each of them. We provide an implementation of each ODP using the ODP template language OTTR. We draw the different context modeling problems from a real-world project but abstract from them such that the ODPs may be reused in different application scenarios and domains. Finally, in Section 4 we briefly discuss the results.

2 Related Work

2.1 Context and OWL

There exist a number of approaches to the problem of representing context in OWL (or Description Logics, with which OWL 2, being a syntactical variant of the DL $\mathcal{SROIQ}(D)$, shares its semantics).

One kind of approach uses standard OWL constructs for modeling context information. An example of this kind of approach is the *Context Slices* pattern[4], where a context is represented by an OWL individual, which is connected to a

[4] `http://ontologydesignpatterns.org/wiki/Submissions:Context_Slices`

pair of individuals that take part in a context-dependent object property assertion [13]. This context individual may then be referenced by other individuals representing, for example, an agent playing a role in that context (e.g. a person believing that the relation represented by the object property assertion holds).

Upper ontologies that have an OWL axiomatization use similar patterns for representing context. For example, the General Formal Ontology (GFO) [3] defines the abstract category of a role an agent may play in a certain context. Roles may be processual roles, representing memberships in a process that may take part in a temporal context (a time interval), social roles, or relational roles that contextualize participants in a relation and with respect to relators, which represent instances of a relation.

Approaches of this type have in common that the subject of the contextualization are (binary or n-ary) relations between instances of the discourse domain (object property assertions in OWL terms). There exist more general approaches in terms of expressivity, which extend the contextualization facility to ABox and even TBox axioms and are thereby capable of formalizing context not only on the individual but also on the concept level.

The most basic approach of this kind, which is already built into the OWL language, are axiom annotations, which allow ABox and TBox axioms to be annotated with either an anonymous individual, an IRI (which might refer to another OWL entity) or a literal, which, in turn, might carry contextual information. OWL Annotations, however, do not have any formal semantics, which leaves the relation between the contextual information and its subject undefined and makes the development of formal reasoning procedures impossible.

Formal approaches for contexualizing OWL axioms naturally require an extension of the OWL language and an extended semantics since it is not possible to represent arbitrary statements about concepts (classes) in OWL. They usually deploy multidimensional Description Logics, either a Description Logic combined with a domain specific logic, such as temporal logics, or combinations of standard Description Logics, such as $\mathcal{ALC}$ and $\mathcal{SROIQ}$. A significant work in this field is the PhD thesis by Klarman [5] who investigates different combinations of DLs for representing context and delivers important complexity results.

In the following we introduce a formalism for representing context in OWL that builds on the results of Klarman but alleviates some of the shortcomings of multi-dimensional DLs, for example conserving decidability by restricting the interaction between the different logics. The approach is inspired by the Aspect-Oriented Programming paradigm and is hence named Aspect-Oriented Ontology Development.

The ODPs we present later in Section 3 are based on this approach, but (with one exception) are general enough to be used in different context formalisms without change.

2.2 Aspect-Oriented Ontologies

Aspect-Oriented Ontology Development (AOOD) [7] is an extension to the OWL 2 language, which allows context (here called *aspects*) to be added to (TBox and

ABox) axioms in an OWL ontology. It is inspired from the Aspect-Oriented Programming (AOP) paradigm [4]. AOOD uses Modal Logics as a context language and makes use of the fact that Modal Logics are syntactic variants of Description Logics, i.e., (almost) every Modal Logic can be translated to a particular Description Logic and thereby to OWL.

In the remainder of this subsection we will give a brief overview of Aspect-Oriented Programming and Modal Logics and then introduce Aspect-Oriented Ontology Development, which is built on top of these two notions.

Aspect-Oriented Programming AOP is an extension to Object Oriented Programming (OOP). Its primary purpose is to increase modularity in software code by separating so called cross-cutting concerns, such as authentication and logging, from the actual business logic of the application. Cross-cutting concerns lead to scattered and tangled code and cannot be modularized by the decomposition mechanisms of object-oriented or procedural languages. AOP introduces *aspects* as a solution to this problem.

An *aspect* (in the following referred to as *software aspect* in order to distinguish it from an ontology aspect, which we will describe in the next subsection) is an isolated code module representing exactly one concern. In order to connect these isolated modules AOP provides a *join point* model. A join point is a place in the code, described, for example, by the signature of a procedure and additional qualifiers, such as "before", "after", "before return", or "after exception". The join point model also allows the abstract specification of sets of join points, which are called *pointcuts*.

A pointcut specification is a quantified query over the set of potential join points. A pointcut language may allow to select sets of join points on the lexical level (e.g. using regular expressions over method names) or by their signature (by matching parameter types), or both. A pointcut may, for example, consist of all getter methods (all methods whose names start with "get") of all classes in a certain package that take a string and an integer as arguments.

An *advice* is the code belonging to a concern that should be executed at each join point of another concern. Each software aspect consists of a pointcut and an advice. The advice code is said to be *advising* the target code, selected by the pointcut. Due to the fact that the advice code is invisible to its target and that the target is selected by quantification, *quantification* and *obliviousness* were identified as the two main principles of AOP [2].

Since software aspects are classes (in OOP terms) with executable methods, they carry potential join points themselves. Therefore, software aspects may be arbitrarily nested.

Modal Logics Modal logics are propositional logics extended with two additional operators $\Box$ and $\Diamond$. Depending on the type of modal logic, the operators are named differently.

For example, in basic modal logic, $\Box$ is simply named *box*, and $\Diamond$ is named *diamond*. In logics that study necessity and possibility, $\Box$ is named *necessarily*,

and $\Diamond$ is named *possibly*. In epistemic logics, $\Box$ stands for *"it is known that"*, and $\Diamond$ stands for *"it is believed that"*. In temporal logics, $\Box$ may stand for *"it has always been the case that"* or *"it will always be the case that"*, and $\Diamond$ may stand for *"it was the case that"* or *"it will be the case that"*, although it is common practice to use different symbols for the different types of logics.

Modal Logics are a syntactic variant of Description Logics [9]. This makes them easily combinable with DL based languages, since the languages' primitives may be mapped to either logic's features. Yet, the semantics of Modal Logics provide a more intuitive access to the notion of ontology aspects.

Definition 1. *A* Kripke frame *is a tuple* $\mathcal{F} = (W, R_i)$*, where* W *is a set of* possible worlds*, and* $R_i \subseteq W \times W$ *an* accessibility relation *between worlds.*

Furthermore, a Kripke model *is a tuple* (W, R_i, L)*, with* W *again a set of possible worlds,* R_i *an accessibility relation, and* $L : W \rightarrow \mathcal{P}(Prop)$ *a* labeling function*, where* $\mathcal{P}$ *is a* valuation function $Prop \rightarrow \{T, F\}$ *that maps propositional symbols to truth values. Moreover, let* $\Box_i$ *and* $\Diamond_i$ *be modal operators.*

Then, the truth value of the propositional formula ϕ *at a possible world* w : $M, w \models \phi$ *is defined in the following way:*

- $M, w \models T$ *and* $M, w \not\models \bot$
- $M, w \models p$ *iff* $p \in L(x)$
- $M, w \models \neg\phi$ *iff* $M, w \not\models \phi$
- $M, w \models \phi_1 \vee \phi_2$ *iff* $M, w \models \phi_1$ *or* $M, w \models \phi_2$
- $M, w \models \phi_1 \wedge \phi_2$ *iff* $M, w \models \phi_1$ *and* $M, w \models \phi_2$
- $M, w \models \Box_i\phi$ *iff* $\forall w' \in W : wR_iw' \rightarrow M, w' \models \phi$
- $M, w \models \Diamond_i\phi$ *iff* $\exists w' \in W : wR_iw' \rightarrow M, w' \models \phi$

Analogously to Description Logics, there exist different types of Modal Logics. The type is determined by adding certain axioms, and it is possible to alter the behavior of the logic and obtain a particular type of modal logic. According to the *correspondence theorem*, each axiom corresponds to a particular condition which is imposed on the frame. This in turn corresponds to (a combination of) characteristic of the accessibility relation of the frame. For example, temporal modal logics have accessibility relations that represent the notions of *before* and *after* (with the related possible worlds representing instances in time). These relations are *transitive*. Depending of the conceptualization of time they might also be *dense* and *reflexive*.

In summary, it is possible to determine a certain type of Modal Logic by defining an accessibility relation and fixing its characteristics. Table 1 shows the correspondences between logic types, modal, conditions on frames, characteristics of the accessibility relation, and the corresponding DL axiom(s).

Aspect-Oriented Ontology Development and Aspect OWL Aspect-Oriented Ontology Development [7] is a formalism for contextualizing OWL axioms. It comprises a development methodology (that allows for context-sensitive, modular ontology development) and a representation formalism, which consists of an extended version of the OWL 2 language and is called *Aspect OWL*.

Name	Modal Axiom	Condition on Frames	R is...	DL Axiom
(D)	$\Box A \rightarrow \Diamond A$	$\forall w \exists u : wRu$	Serial	$\top \sqsubseteq \exists R.\top$
(M)	$\Box A \rightarrow A$	$\forall w : wRw$	Reflexive	$\top \sqsubseteq \exists R.\mathsf{Self}$
(4)	$\Box A \rightarrow \Box\Box A$	$(wRv \wedge vRu) \Rightarrow wRu$	Transitive	$\mathsf{Trans}(R)$
(B)	$A \rightarrow \Box\Diamond A$	$wRv \Rightarrow vRw$	Symmetric	$\mathsf{Sym}(R)$
(5)	$\Diamond A \rightarrow \Box\Diamond A$	$(wRv \wedge wRu) \Rightarrow vRu$	Euclidean	$R^{-1} \circ R \sqsubseteq R$ [a]
(CD)	$\Diamond A \rightarrow \Box A$	$(wRv \wedge wRu) \Rightarrow v = u$	Functional	$\top \sqsubseteq (\leqslant 1 R.\top)$
$(\Box M)$	$\Box(\Box A \rightarrow A)$	$wRv \Rightarrow vRv$	Shift Reflexive	$\exists R^{-1}.\top \sqsubseteq \exists R.\mathsf{Self}$
$(C4)$	$\Box\Box A \rightarrow \Box A$	$wRv \Rightarrow \exists u(wRu \wedge uRv)$	Dense	$R \circ R \sqsubseteq R \wedge \top \sqsubseteq \nexists R.\mathsf{Self}$ [a]
(C)	$\Diamond\Box A \rightarrow \Box\Diamond A$	$wRv \wedge wRx \Rightarrow \exists u(vRu \wedge xRu)$	Convergent	— [b]

[a] falls under OWL 2 restriction

[b] Not possible in pure DL. See Section 3 for a workaround involving SWRL.

Table 1. Modal Logic axioms and corresponding conditions on frames and OWL axioms

The principle idea behind AOOD is to use ontology pointcuts (as described in subsection 2.2), which define ontology modules (sets of OWL axioms) and aspects in order to attach additional context knowledge (advice) to each of these modules. In Aspect OWL, an advice is an OWL class expression. The semantics of an OWL axiom advised by such a class expression is defined as follows: Each instance of the advice class expression is interpreted as a possible world. The advised axiom is valid for each possible world that is an instance of the advice class expression.

Furthermore, the advised axiom is possibly valid (in terms of modal possibility) if there exists an accessibility relation R that connects at least one individual to an instance of the advice class expression. The advised axiom is necessarily valid (again, in terms of modal neccessity) for all individuals that can access instances of the advice class via R.

As an example, consider the OWL class assertion

$$\mathsf{German\_Chancellor}(\mathsf{Angela\_Merkel})$$

This assertion reflects a part of reality valid at the time of writing this paper, but it obviously is only valid in a certain temporal context (Angela Merkel has not always been the German chancellor, and she will cease to be chancellor at some, yet unknown, point in time).

In order to represent the statement in its temporal context, we define an OWL class Merkels_Chancellorship and an individual MC_beginning representing the point in time of the beginning of Merkel's chancellorship. Furthermore, we need to define an accessibility relation after and declare it transitive. Then we can define Merkels_Chancellorship as:

$$\mathsf{Merkels\_Chancellorship} \equiv \exists \mathsf{after}.\{\mathsf{MC\_beginning}\}$$

This represents the temporal context as a half-open interval that begins at time point MC_beginning (excluding the time point itself) and including every time point that comes after it.

An obvious advantage of this approach is that it is possible to reuse existing vocabulary for representing the context. For instance, the W3C time ontology[5] is fully compatible with the above example and can be used for defining the involved entities in the following manner:

$$\begin{aligned}\mathsf{Merkels\_Chancellorship} \equiv \exists \mathsf{time} : \mathsf{after}.\{\mathsf{MC\_beginning}\},\\ \mathsf{time} : \mathsf{Instant}(\mathsf{MC\_beginning}),\\ \mathsf{time} : \mathsf{inXSDDateTime}(\mathsf{MC\_beginning}, \text{"}2005-11-22\mathsf{T}00:00\text{"}\hat{}\hat{}\,\mathsf{xsd} : \mathsf{dateTime})\end{aligned}$$

What still needs to be done is to connect the advice class representing the context to its pointcut (in this case the single class assertion axiom).

For this purpose, Aspect OWL introduces a new axiom type called *aspect assertion axiom*. An aspect assertion is a binary relation between an OWL axiom and an advice class expression (the context of the axiom). Syntactically, aspect assertion axioms resemble annotation assertion axioms. They differ from the latter in that they have a defined model theoretic semantics, which makes use of combined interpretations, which we call a $\mathcal{SROIQ}_{Kripke}$ interpretation.

Definition 2. *A $\mathcal{SROIQ}_{Kripke}$ interpretation is a tuple $\mathcal{J} := (W, R, L, \cdot^{\mathcal{J}}, \Delta, (\cdot^{\mathcal{I}_w})_{w \in W})$ with W being a nonempty set, called* possible worlds*, and L a Kripke interpretation, assigning truth values to propositional symbols in each world $w \in W$ as described in the subsection about Modal Logics.*

For every $A \subseteq W$, $\mathcal{I}_A$ is a DL interpretation.

The semantics of an aspect of an axiom is then defined as follows:

Definition 3. *Let $\mathcal{J} := (W, R, L, \cdot^{\mathcal{J}}, \Delta, (\cdot^{\mathcal{I}_w})_{w \in W})$ be a possible-world DL interpretation. We interpret an aspect under which an axiom α holds as follows:*

$(\mathsf{hasAspect}(\alpha, A))^{\mathcal{J}} \rightarrow A^{\mathcal{J}} \subseteq C^{\mathcal{J}} := \{w \in W \mid \mathcal{I}_w \models \alpha\}$. Because of the correspondence as described in the subsection about Modal Logics we can set $W = C^{\mathcal{J}}$, such that on the semantic level each individual corresponds to a possible world. Furthermore, we set L such that $L(\alpha)^{\mathcal{J}} := A^{\mathcal{J}}$.

The resulting contextualized axiom is depicted in Figure 1.

A further advantage of this approach is that it keeps OWL's monotonicity intact. If in the future after the end of Angela Merkel's chancellorship someone closes the interval by adding a corresponding individual MC_end and the axiom $\mathsf{Merkels\_Chancellorship} \equiv \exists \mathsf{time} : \mathsf{before}.\{\mathsf{MC\_end}\}$, no information is erased. While the fact alone that Angela Merkel is chancellor becomes invalid the *contextualized* fact that Angela Merkel is chancellor *during a specified time interval* will remain valid independently of the current point in time.

[5] `http://www.w3.org/TR/owl-time/`

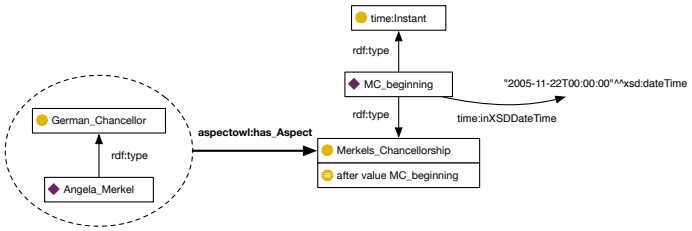

Fig. 1. An example of a temporal aspect. The pointcut of the aspect is the class assertion axiom inside the dashed circle. The advice (contextualizing the axiom) is at the top.

We explicitly restrict the interaction between the logics in such a way that only aspects can "see" their advice. That means that each aspect results in a local interpretation of an axiom. Even if a contextualized axiom has entity names from its advice in its signature, these entities are interpreted differently from the ones on the advice level. This corresponds to the obliviousness principle of Aspect-Oriented Programming.

Aspect OWL has more expressive features. For example, it is possible to define pointcuts using queries over axioms, by either providing a signature (a set of entity names), a DL query or, for selecting axioms on the RDF graph level, a SPARQL construct query. Due to space constraints it is not possible to cover these facilities in detail.

For a more thorough description of the features and semantics of Aspect OWL 2, we refer the reader to [7] and [8][6].

3 ODPs for Different Kinds of Context

In this section, we present a set of Ontology Design Patterns for representing different kinds of context as defined above. The selection of kinds of context is driven by the requirements identified in the research project *COMPASS* (Comprehensive Surgical Landscape Guidance System for Immersive Assistance in Minimally-invasive and Microscopic Interventions)[7].

The aim of the project is to build a markerless system that helps surgeons navigate through a situs during endoscopic surgical interventions by providing useful, context-sensitive real-time information but avoiding information overload that might overwhelm the surgeon. For this purpose, the system requires an internal model of the spatial structure of the situs (e.g. the arrangement of anatomical features), which needs to be mapped to a model of navigational situations of the surgeon and her endoscope. To this end, several ontologies are being developed. Among them are an ontology of geometrical arrangements

[6] For an overview over the complete abstract/functional style syntax of Aspect OWL 2, see `http://www.aspectowl.xyz/syntax/`.

[7] `https://www.iccas.de/projekte/compass-2/?lang=en`

of anatomical structures and an ontology of basic geometric shapes, which are aligned to the OWL version of the GFO ontology. Furthermore, ontologies from a predecessor project are reused, such as the ontology of endoscopic situations and risk structures in FESS interventions [11]. It turns out that a significant amount of concepts captured in our model are context-dependent and may be represented using Aspect OWL.

We use the correspondence of modal axioms and the respective conditions on modal frames in order to represent the modal axioms in OWL. We do this by translating the frame conditions to Description Logic axioms. The column *DL Axiom* in Table 1 shows the corresponding DL axioms.

Note that convergent frames cannot be modeled using pure DL. It is however possible to represent convergence using SWRL and the SWRLX extensions built-in makeOWLThing[8]. This would reduce the entire matter to a single SWRL rule:

$$\mathsf{R(?s,\ ?d1)},\ \mathsf{R(?s,\ ?d2)},\ \mathsf{DifferentFrom\ (?d1,\ ?d2)},\ \mathsf{swrlx{:}makeOWLThing(?c,\ ?s)} \rightarrow \mathsf{R(?d1,\ ?c)},\ \mathsf{R(?d2,\ ?c)}$$

The swrlx:makeOWLThing extension, in turn, is not supported by all OWL reasoners or rule engines. A facility supporting this extension is the SWRL API[8] [6], which comes with a Drools-based forward chaining OWL 2 RL reasoner and SWRL engine.

The formalism we selected for the representation of the ODPs is OTTR [12]. OTTR is a framework for designing parametrizable ontology templates. The selection is based on the nature of Modal Logics and the fact that instances of the same Modal Logic family are structurally very similar and differ only in accessibility relation names and characteristics as well as the number and names of possible worlds. This makes our contexts, which highly depend on these structures, a perfect candidate for a template language.

Furthermore, although our modeling problems are derived from our real world project and cater concrete scenarios tackled in this project, they are sufficiently general to be relevant in other contexts.

3.1 Named Contexts: Endoscopic Perspective

The simplest and most general kind of context the AOOD framework can represent is that of named contexts. This kind of context may be used to represent views on a domain, or any other categorization of axioms.

The underlying Modal Logic is the simple system K. It has an irreflexive frame (which corresponds to the lack of the axiom $\Box A \rightarrow A$). In this simple model, we ignore the existence of any accessibility relation (which is the reason why the lack of the axiom is not harmful).

An advice class is simply defined by a singleton nominal $A \equiv \{I\}$. The intended meaning is that the individual I represents the simple context.

[8] https://github.com/protegeproject/swrlapi/wiki/ExtensionsBuiltInLibrary

Listing 1.1. OTTR template for a simple named aspect. The template takes as input arguments a class IRI for the advice class and an individual IRI and constructs the singleton nominal class equivalence for the advice class.

```
:NamedAspect [ ! owl:Class ?NamedAspectClass, ! owl:Individual ?
    NamedAspectIndividual ] :: {
  o-rdf:Type(?NamedAspectClass, owl:Class),
  o-rdf:Type(?NamedAspectIndividual, owl:Individual),
  o-owl-ax:EquivObjectOneOf(?NamedAspectClass, (?NamedAspectIndividual))
} .
```

Figure 2 shows an example of two named aspects, and Listing 1.1 shows the ODP written in stottr[9].

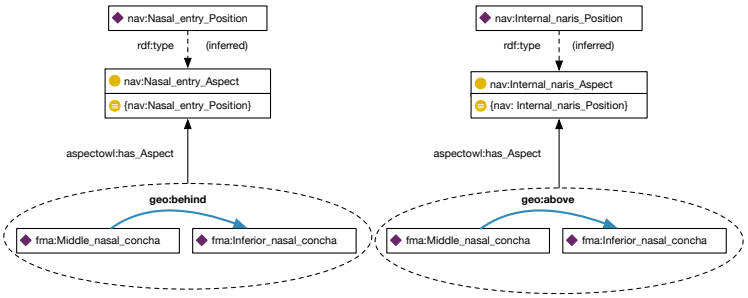

Fig. 2. Two named aspects representing two visual perspectives and the perceived arrangements of anatomic structures in the context of each perspective. Seen from the nasal entry, the inferior nasal concha appears to be behind the middle concha, whereas seen from the internal naris, the former appears to be located above the latter.

3.2 Temporal Context: Changes Over Time

We have already introduced temporal context in Section 2.2 and provided an example. With time being one of the most fundamental concepts pervading every aspect of existence, temporal contexts play a role in the COMPASS system as well. For example, the structure of the situs may change over time (parts may be resected during the surgical intervention) and may hence be present before a certain point in time and absent after that point in time.

It is interesting to note that temporal context may be described as a closed or a half open interval (left or right, depending on whether the beginning or the end are fixed). If the interval is half open, it only necessary to define one of the two accessibility relations that correspond to the notions of *before* and *after*. As a consequence of one being the inverse of the other, it is in fact not necessary to define both at all, since one can always refer to e.g. the property $time : right$ one

[9] Due to space constraints we are not able to provide an introduction to the STOTTR syntax. We ask the reader to refer to `https://dev.spec.ottr.xyz/stOTTR/`.

Listing 1.2. OTTR template for a temporal aspect. The arguments it accepts are a class IRI for the advice class, two object properties for the relations *before* and *after* (one of which is optional) and individual names for the left and right boundaries (start and end points), respectively (one of which is optional as well).

```
:TemporalAspect [ ! owl:Class ?TemporalAspectClass, ! owl:ObjectProperty ?
    BeforeRelation, ! owl:ObjectProperty ?AfterRelation, ? owl:Individual ?
    LeftBoundary, ? owl:Individual ?RightBoundary ] :: {
  o-rdf:Type(?TemporalAspectClass, owl:Class),
  o-owl-ax:SubClassOf(?TemporalAspectClass, aspect-owl:TemporalAspect),
  o-rdf:Type(?BeforeRelation, owl:ObjectProperty),
  o-rdf:Type(?BeforeRelation, owl:TransitiveProperty),
  o-rdf:Type(?AfterRelation, owl:ObjectProperty),
  o-rdf:Type(?AfterRelation, owl:TransitiveProperty),
  util:EqiuvUnionHasValueOneOf(?TemporalAspectClass, ?AfterRelation, ?
      LeftBoundary),
  util:EqiuvUnionHasValueOneOf(?TemporalAspectClass, ?BeforeRelation, ?
      RightBoundary)
} .
```

as $time : left^{-1}$. It is therefore merely for convenience that we included both relations in the pattern. One of the two may, however, be omitted if the interval that the template instance is supposed to represent is half open.

Listing 1.2 shows the ODP written in stottr. Note that in the template, we include the start and end points in the interval, whereas in the example above they were not included for the sake of simplicity. This makes the axioms and the corresponding pattern somewhat more complicated since the advice class is now equivalent to the union of the value restriction and the singleton nominal containing the start (or end) point itself. Also note that the template makes call to an auxiliary template *EqiuvUnionHasValueOneOf*, which we defined for the purpose of instantiating the union and the restrictions/class expressions it comprises.

Note that, depending on the application, it might be necessary to represent time with different properties, for example, dense (between each arbitrary pair of time points fit infinitely many time points). OTTR in its current state, however, does not provide means for parametrization on that level. As a workaround, we provide a template for each possible combination notions of time.

3.3 Epistemic Context: Knowledge about Knowledge

As mentioned in the introduction to this section, one requirement for the COMPASS system is to avoid information overload. In order to capture the information need (and, in turn, knowledge about what information is not needed), we developed a model of knowledge about the knowledge of a group of agents, one agent being the surgeon and the other agents being system components with the capability of providing certain types of information.

We use Epistemic Logic, which is also a type of Modal Logic in order to model this situation. Epistemic Logic may be used to model epistemic states of a group of agents from an outside point of view.

Suppose a navigation component is able to provide the information that the next landmark that will be reached by the endoscope is the uncinate process of

Listing 1.3. OTTR template for an epistemic aspect

```
:EpistemicAspect [ ! owl:Class ?AgentsGlobalEpistemicAspectClass , ! owl:
    ObjectProperty ?AgentsEpistemicCompatibilityRelation , ! owl:Individual ?
    AgentsLocalEpistemicWorld ] :: {
  o-rdf:Type(?AgentsGlobalEpistemicAspectClass , owl:Class),
  o-owl-ax:SubClassOf(?AgentsGlobalEpistemicAspectClass , aspect-owl:
      EpistemicAspect),
  o-rdf:Type(?AgentsEpistemicCompatibilityRelation , owl:ObjectProperty),
  o-rdf:Type(?AgentsEpistemicCompatibilityRelation , owl:ReflexiveProperty),
  o-rdf:Type(?AgentsEpistemicCompatibilityRelation , owl:TransitiveProperty),
  o-rdf:Type(?AgentsEpistemicCompatibilityRelation , owl:SymmetricProperty),
  o-rdf:Type(?AgentsLocalEpistemicWorld , owl:Individual),
  o-owl-ax:EquivHasValue(?AgentsGlobalEpistemicAspectClass , ?
      AgentsEpistemicCompatibilityRelation , ?AgentsLocalEpistemicWorld)
} .
```

ethmoid bone, but due to previous information provided we can assume that the surgeon is aware of that particular fact. Then the system and the surgeon are in compatible epistemic states and the information may be unnecessary.

In Epistemic Logic, possible worlds represent those epistemic states. The accessibility relations (there exists one such relation per agent) express compatibility between epistemic states (by stating that, from the particular agent's point of view, two worlds connected by them are epistemically indistinguishable for that agent). One possible world individual represents the actual epistemic "world" of each agent.

The *indistinguishable from* relations are transitive (intuitively, if A is indistinguishable from B, and B is indistinguishable from C, then A is indistinguishable from C), reflexive (every epistemic state is indistinguishable from itself) and symmetric (if A is indistinguishable from B for an agent, then so is B from A). Listing 1.3 shows the epistemic context ODP for one agent written in stottr. In order to represent multiple agents, it is necessary to instantiate the template multiple times, one time for each agent. The possible epistemic states and their connections must be added manually.

3.4 Dynamic Context: Change of Anatomical Situations

Dynamic context is related to temporal context in that it represents changes of the state of affairs. However, instead of change over defined time points, dynamic context captures change as a consequence of actions.

Dynamic context is represented using Dynamic Logic, which is a multimodal logic. Multimodal logics have multiple (indexed) accessibility relations. In Dynamic Logics, each accessibility relation represents an action that might or might nor result in changes of the state of the world. A possible world represents the world in a certain combination of states.

Figure 4 depicts a dynamic context with two actions (a navigation action that leaves the world unchanged and an action "Resection of orbital plate of ehmoid

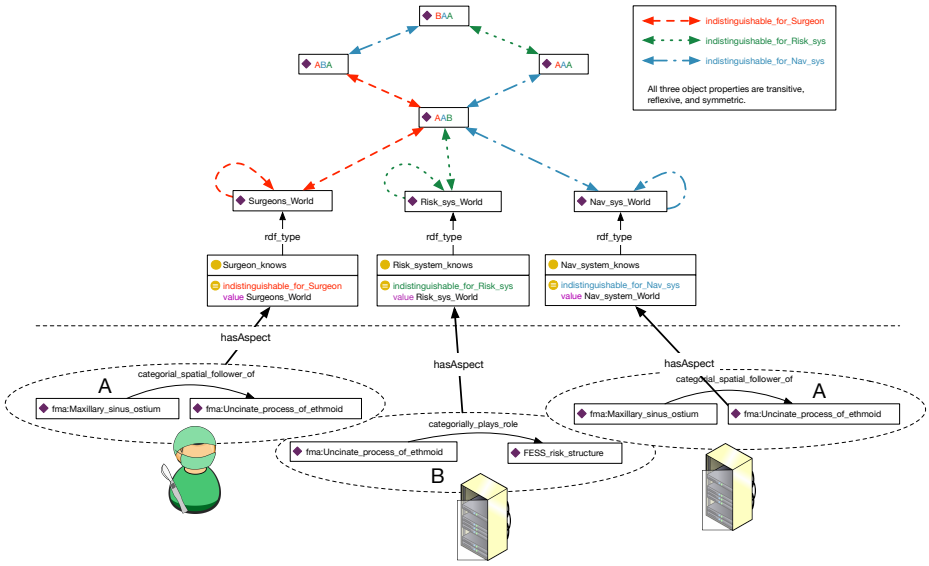

Fig. 3. Epistemic contexts of three agents (a surgeon and two systems). The names of the epistemic state individuals at the top encode possible knowledge of an agent about one of the assertions at the bottom (note that the leftmost and the rightmost assertions are the same. They are named "A". The assertion in the middle is named "B"). The colors and line patterns represent agents (red/dashed: surgeon, green/dotted: risk system, blue/dash dotted: navigation system). A red B represents the possible assumption that the surgeon knows fact B. There exists a blue/dash dotted path from the nav system's world to a state in which the surgeon knows fact A. However, there is no green/dotted path leading from the risk system's world to a state where the surgeon knows fact B. Therefore, fact B (risk structure ahead) might be unknown to the surgeon, and it might be useful to provide her with that information.

bone", which has as a consequence that the orbital plate of ehmoid is not the spatial follower of the sphenoial sinus while navigating the endoscope[10][11].

3.5 Topological Context: Overlapping of Anatomical Structures

The Modal Logic S4 may be interpreted in terms of topological relations between points and their location inside, on the boundary, or outside of a geometric structure, where $\Box P$ means that point P is inside the structure (excluding its boundary), whereas $\Diamond P$ describes the boundary (closure) [1][12].

[10] Due to space constraints we are not able to list this ODP. All ODPs are, however, available at `http://ontologydesignpatterns.org/wiki/Category:LogicalOP` and `http://odp.aspectowl.xyz`

[11] The ODP for dynamic aspects is available at `http://odp.aspectowl.xyz/aspect/0.1/DynamicAspect`

[12] available at `http://odp.aspectowl.xyz/aspect/0.1/TopologicalAspect`

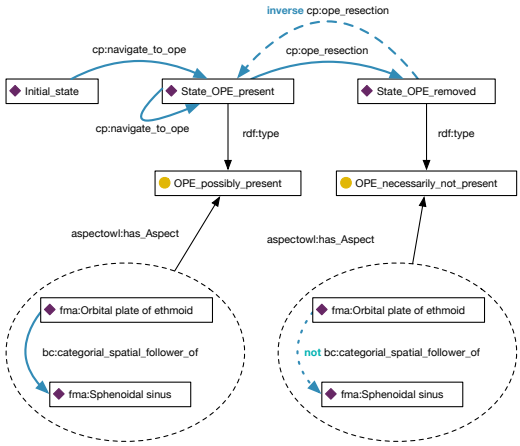

Fig. 4.

4 Discussion

The analysis of the application domain of automatic assistance for endoscopic surgeons revealed that there exist a number of knowledge representation problems where context of different kinds plays a crucial role. Aspect-Oriented Ontology Development which uses different kinds of (multi-)modal logics proves to be an adequate solution to this problem. The recurring nature of context (there is, for example, not just *one* temporal context, but many facts have their own temporal context) makes it a good candidate for an ODP.

We could show that different kinds of context may be represented by an ODP that instantiates a particular context in the form of an aspect. However, there are some caveats. On the representation side, directly interpreting Modal Logics in terms of DL axioms may lead to an inadvertent combination of object property characteristics that violate OWL 2 restrictions and lead to reasoners aborting the reasoning process with an error message. See [10] for an in-depth discussion of the problem (and possible work-arounds).

One shortcoming that we encountered is a lack of expressivity in the ODP template languages. To our knowledge, OTTR is the most advanced species of these formalisms at the time of writing this paper. However, even OTTR has limited support for parametrization of patterns. For example, it is not possible to branch pattern construction depending on the presence of an optional argument. Another issue we had with OTTR is the construction of singleton lists from optional IRI parameters. If a template constructs a list from an argument, and the argument is empty, OTTR will still create a non-empty list, containing the resource $ottr : none$, which is, at least for our purposes, not the desired behavior.

Finally, not everyone might agree to the way we formalize context, and there are different formalisms for context representation. We think, however, that our formalism has the advantage of being general and applicable in many domains

and to many kinds of context. Except for the case of Dynamic Context, the patterns presented in this paper may be decoupled from the aspect-oriented ontology approach in which we used them, and combined with a different approach, as long as the alternative approach provides some mechanism for attaching the context entities to the domain knowledge.

References

1. van Benthem, J., Bezhanishvili, G.: Modal Logics of Space. In: Aiello, M., Pratt-Hartmann, I., Van Benthem, J. (eds.) Handbook of Spatial Logics, pp. 217–298. Springer Netherlands, Dordrecht (2007)
2. Filman, R., Friedman, D.: Aspect-Oriented Programming Is Quantification and Obliviousness. Workshop on Advanced Separation of Concerns, OOPSLA (2000)
3. Herre, H., Heller, B., Burek, P., Hoehndorf, R., Loebe, F., Michalek, H.: General Formal Ontology (GFO): A Foundational Ontology Integrating Objects and Processes. Part I: Basic Principles. Tech. rep., Research Group Ontologies in Medicine (Onto-Med), University of Leipzig (2007)
4. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.: Aspect-Oriented Programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP'97 — Object-Oriented Programming, Lecture Notes in Computer Science, vol. 1241, pp. 220–242. Springer Berlin / Heidelberg (1997)
5. Klarman, S.: Reasoning with Contexts in Description Logics. Ph.D. thesis, Vrije Universiteit Amsterdam (2013)
6. O'Connor, M., D. Shankar, R., Musen, M., Das, A., Nyulas, C.: The SWRLAPI: A Development Environment for Working with SWRL Rules. (01 2008)
7. Schäfermeier, R., Paschke, A.: Aspect-Oriented Ontologies: Dynamic Modularization Using Ontological Metamodeling. In: Garbacz, P., Kutz, O. (eds.) Proceedings of the 8th International Conference on Formal Ontology in Information Systems (FOIS 2014). Frontiers in Artificial Intelligence and Applications, vol. 267, pp. 199 – 212. IOS Press (2014)
8. Schäfermeier, R., Paschke, A.: Aspect-Oriented Ontology Development, pp. 3–30. Springer International Publishing, Cham (2018)
9. Schild, K.: A Correspondence Theory for Terminological Logics: Preliminary Report. In: Mylopoulos, J., Reiter, R. (eds.) Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991. pp. 466–471. Morgan Kaufmann (1991)
10. Schneider, M., Rudolph, S., Sutcliffe, G.: Modeling in OWL 2 without Restrictions. In: OWLED. CEUR Workshop Proceedings, vol. 1080. CEUR-WS.org (2013)
11. Siemoleit, S., Uciteli, A., Bieck, R., Herre, H.: Processual Reasoning over Sequences of Situations in Endoscopic Surgery. In: German Medical Data Sciences: Visions and Bridges, Studies in Health Technology and Informatics, vol. 243, pp. 222–226 (2017)
12. Skjæveland, M.G., Forssell, H., Klüwer, J.W., Lupp, D.P., Thorstensen, E., Waaler, A.: Pattern-Based Ontology Design and Instantiation with Reasonable Ontology Templates. In: WOP@ISWC (2017)
13. Welty, C., Fikes, R.: A Reusable Ontology for Fluents in OWL. In: Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006). pp. 226–236. IOS Press, Amsterdam, The Netherlands, The Netherlands (2006)