Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

A Core Ontology on Events for Representing Occurrences in the Real World

Ansgar Scherp · Thomas Franz · Carsten Saathoff · Steffen Staab

Accepted: November 18, 2010

Abstract Events are central aspect of many semantic ambient media applications such as surveillance, smart homes, automobiles, and others. Existing models for events typically do not follow a systematic development approach, are conceptually narrow with respect to event features, and their semantics is often ambiguous. This makes the communication between and integration of different event-based components and event-based semantic ambient media applications a challenging task. In this paper, we present the Event-Model-F, a formal model of events based on the foundational ontology DOLCE+DnS Ultralite (DUL). The Event-Model-F provides comprehensive support to represent time and space, objects and persons, mereological, causal, and correlative relationships between events, and different interpretations of the same event. It is developed following a pattern-oriented ontology design approach and can be easily extended by domain specific ontologies. We introduce the design and implementation of an application programming interface that allows for easy integration of the Event-Model-F in arbitrary applications. The use of the Event-Model-F is demonstrated at the example of a socio-technical system of emergency response and implemented in the SemaPlorer++ application for creating and sharing event descriptions.

Keywords Events · Models · Principles · Pattern-oriented Ontology Design · Core Ontologies

1 Introduction

The explicit modeling of events and the development of event-based systems are increasingly gaining widespread attention by research and industry [1, 44]. This is due to a couple of reasons: Firstly, we find an increasing number of semantic ambient media applications that are treating events, e.g., surveillance, smart homes and automobiles, ambient assisted living, and situation management such as in the example of emergency

Ansgar Scherp (primary contact), Thomas Franz, Carsten Saathoff, and Steffen Staab
University of Koblenz-Landau, Universittsstr. 1, 56070 Koblenz, Germany
Tel.: +49-261-287-2717
Fax: +49-261-287-100-2717
E-mail: {scherp,franz,saathoff,staab}@uni-koblenz.de

response. Secondly, a fastly growing number of intelligence-collecting devices such as sensors, CCTV, upload facilities, and others lead to an ubiquity of events being recognized and communicated. Thirdly, event detection, clustering, and annotation is and will be realized in many different software components and proprietary solutions using a large variety of internal data models. Thus, multiple systems are connected for managing events resulting in a complex, distributed event-based system [34]. Such a distributed event-based system consists of several components that are characterized by taking events as input and providing events as output. Existing semantic ambient media systems and applications typically focus on processing low-level signals and actions such as [9, 58, 73, 6]. They concentrate on the technical events that are captured by sensory devices and try to make sense out of the captured information.

On the domain-level, events are understood as occurrences in which humans participate. It is essential for semantic ambient media applications to capture and represent these occurrences, i.e., to consider events on domain-level. In contrast to the technical events above, events on domain-level are typically subject to discussions and interpretations by humans [50]. They may be very complex and a variety of aspects need to be considered such as time and space, objects and persons involved, as well as mereological, causal, and correlative relationships between events. Existing models for events like [67, 43, 22, 14, 33, 17, 70] and models for situation-awareness that have a close relation to events and provide event-like structures such as [12, 68, 72, 32, 30, 24, 26] are typically developed in an ad-hoc manner. Thus, they do not follow a systematic development approach. They are conceptually narrow with respect to the event features provided and the semantics is typically ambiguous. This hinders communication between and interoperability of the different event-based components and event-based semantic ambient media applications.

Modeling events is an interesting and challenging task as it involves among others the different aspects of events like objects participating in events, mereological, causal, and correlative relationships between events, and interpretations of the same event by different humans. In philosophical literature, there are different discussions of how to discriminate events from other categories [10]. One of them are objects. Although not undisputed, there are standard differences between events and (physical) objects [10]: Events are said to *occur* or *happen*. They are considered perduring entities that unfold over time, i.e., they take up time. In contrast, material objects such as stones and chairs are said to *exist*. Such enduring entities unfold over space, i.e., they are in time. As said, this metaphysical distinction is not uncontroversial as some philosophers consider objects as four-dimensional entities that extend across time just as they do across space [10].

With the Event-Model-F, we present a formal representation of events that allows for capturing and representing occurrences in the real world [52]. This representation allows easy interchange of event information between different event-based components and event-based systems. For our Event-Model-F, we base on the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [8, 21, 28]. We have aligned the Event-Model-F with the DOLCE+DnS Ultralite (DUL) ontology[1], a lightweight version of DOLCE. The Event-Model-F follows the design decision of DUL and distinguishes events from objects. By this, we can be precise about the relationships that can occur between events and objects [37]. DOLCE and DUL already have proved to be

[1] `http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite` [Last retrieved: August 04, 2010]

a good modeling basis for core ontologies such as [45, 45, 2, 18, 38, 39] and provide a formal modeling basis. For designing the Event-Model-F, we have followed the pattern-oriented ontology design approach of DOLCE and DUL, respectively. This approach provides native support for modularization of our Event-Model-F and extension by domain specific ontologies.

The remainder of this article is organized as follows: The next section motivates the need for a formal model on events at the example of a concrete scenario in the domain of emergency response. Section 3 introduces the SemaPlorer++ application for creating and sharing event descriptions in the emergency response scenario. Section 4 presents the requirements on the Event-Model-F derived from the scenario, SemaPlorer++ application, and an extensive analysis of existing models in various domains. In Section 5, we describe the development of our Event-Model-F. For a semantically precise definition of the Event-Model-F, we have axiomatized it using Description Logics [4] described in Section 6. In order to allow for an easy use of the Event-Model-F in applications such as SemaPlorer++, we have developed an application programming interface discussed in Section 7. The use of our Event-Model-F in the SemaPlorer++ application at the example of the emergency response domain is demonstrated in Section 8. In Section 9, we present an extensive analysis of existing event models and event-based systems and relate them to the Event-Model-F. We also discuss related applications in the domain of managing emergency response, before we conclude the paper.

2 Sharing of Event Descriptions in Emergency Response

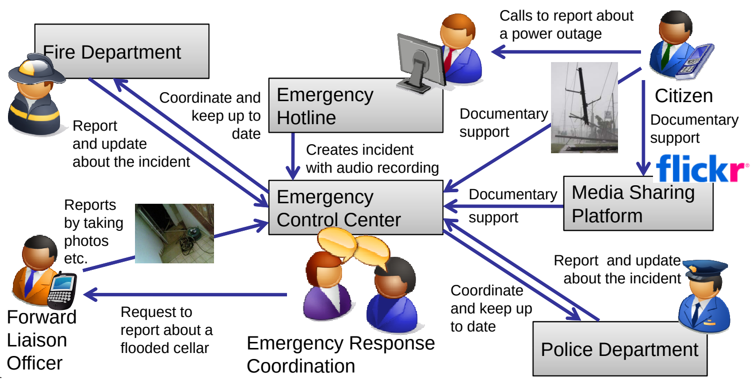
We demonstrate the need for the Event-Model-F at the example of sharing ambient media information in the emergency response use case of the EU project WeKnowIt[2]. We will use the emergency incident of a flooding as a concrete scenario throughout the paper. In the course of such an incident, different professional emergency response entities are involved such as the emergency hotline, police department, fire department, emergency control center, and forward liaison officers. As depicted in Figure 1, the emergency hotline receives calls from citizens and sends event descriptions to the control center. In addition, citizens report about events by using their cell phones and taking images of the incidents. The information provided by the citizens is transmitted from the emergency hotline to the emergency control center. In addition, the citizens can upload content and provide information directly to the emergency control center. Finally, information from social networking platforms are used by the emergency control center such as images shared on Flickr[3]. The emergency control center is in charge of coordinating the emergency response entities. It receives the event descriptions and media associated with it, processes them, and communicates the event descriptions and their documenting media assets with the police department and fire department. Forward liaison officers as part of the emergency control center are out in the field to report about the situation by taking photos and notes.

In the case of a concrete incident this socio-technical system for emergency response becomes very active. Many different descriptions of events and their documenting media assets are created and communicated between the different emergency response

[2] `http://www.weknowit.eu/` [Last retrieved: August 04, 2010]

[3] `http://www.flickr.com/` [Last retrieved: August 04, 2010]

entities. Please note again, the events we are modeling aim at capturing and representing occurrences in the real world where humans participate and thus are different from technical events as discussed above. The different aspects of events are marked with $(\langle number \rangle)$.


Fire Department
Coordinate and keep up to date
Report and update about the incident
Emergency Hotline
Calls to report about a power outage
Citizen
Documentary support
Documentary support
flickr
Creates incident with audio recording
Emergency Control Center
Documentary support
Media Sharing Platform
Reports by taking photos etc.
Forward Liaison Officer
Request to report about a flooded cellar
Emergency Response Coordination
Coordinate and keep up to date
Report and update about the incident
Police Department

sen

Fig. 1 Sharing Event Descriptions in the Socio-technical System of Emergency Response

In an incident of a heavy storm a major flooding may happen. (i) During the flooding a power outage occurs. (ii) Some citizens are lacking power supply and are calling the emergency hotline to report about the outage. The officers at the emergency hotline record these calls and type in an event description for each call to document them in their system. (iii) These events are annotated with information about the call and its recording and are automatically transferred to the system of the emergency control center. In addition, the citizens are reporting about events by taking images of the events, tagging them, and uploading them to the WeKnowIt system. In the course of the flood, (i) further events happen such as pumping out flooded cellars or rescuing people from their flooded homes. (iv) Forward liaison officers drive to the different locations of the events to take photos and report them. Here, the event description is created using an application on their cell phone. (iii) A documentary support for the event is provided by attaching a photo to it and tagging it. The event description is also send to the system of the emergency control center. (v) Although the events of a flooded cellar and rescuing people from their homes might occur at different times and be located farer away from each other, they are probably both caused by the same flooding event. (vi) Thus, the events that happen during an incident correlate to each other. These correlations are important to recognize to gain a full understanding of the emergency situation. Thus, the officers at the control center thoroughly analyze the event descriptions they receive from the forward liaison officers and the emergency hotline. (vii) The emergency control center also receives event descriptions from the systems of the police department and fire department that happen during the incident. (v) Based on the evidence of the event descriptions, the officers in the emergency control center use their system to formulate hypothetical events that might have caused the power outage. To this end, event descriptions are analyzed, (semi-)automatically clustered, visualized, and put into relation. (viii) The officers conclude that there are two possible interpretations that might have caused the power outage, namely a snapped power pole close to the river or a problem with the power plant. The correct assessment of the situation is very crucial in order to most effectively deploy the available emergency response resources. Thus, the different event interpretations modeled in the system need to be verified by the officers as soon as possible in order to confirm or reject the hypothesizes. For this purpose, the officers in the emergency control center may contact the personnel of the power plant. At the same time, the description of the hypothetical event of a snapped power pole together with a task description is sent onto the mobile device of a forward liaison officer. The forward liaison officer drives to the area where the control center suspects that the pole might be snapped, takes images documenting the situation, and reports back the result using his mobile device.

As depicted in Figure 1, several entities are involved in an emergency response using different event-based systems. These systems need to be connected through a common understanding of events into a distributed event-based system. The goal is to efficiently communicate the event descriptions and associated media between the different emergency response entities. Thus, a formal representation of events is useful for the interoperability among the various computer systems as it provides machine accessible semantics. To support this complex, socio-technical system for emergency response, we have developed the SemaPlorer++ application for creating and sharing event-based media information among the emergency response entities.

3 Creating and Sharing Event Descriptions in SemaPlorer++

The SemaPlorer++ application aims at providing a distributed infrastructure for creating and sharing event descriptions and media assets documenting these events in the domain of emergency response. It is an extension of the SemaPlorer application [49] for faceted search and navigation of distributed semantic data sources in real-time. As data sources SemaPlorer uses DBpedia [7], GeoNames[4], WordNet[5], and personal FOAF files[6] (Friend-of-a-Friend). In addition, it connects via a semantic photo query service live to the Flickr photo sharing system. The distributed storage infrastructure of SemaPlorer bases on NetworkedGraphs [48]. It allows to integrate arbitrary further data sources. The user interface of SemaPlorer supports visualizing the semantic data using a map view, media view, and different context views. It bases on the K-Space Annotation Tool (KAT) [46] and can be extended by domain-specific plugins.

To support the specific needs of emergency response to create and share event descriptions, the SemaPlorer++ application provides an event-manager plugin that makes use of the Event-Model-F. Figure 2 shows a screenshot of the SemaPlorer++ application with the event-manager plugin. An ontology browser is located on the left hand side with an emergency response ontology loaded. The emergency ontology is provided by the Sheffield City Council and comprises all emergency incidents that the emergency response team deals with. The emergency events defined in the ontology are listed in a tree-structure of the ontology browser. It enables the user to easily create event descriptions by choosing a concept from the ontology, e.g., the emergency incident concept *Major_Industrial_Fire*. The emergency response concepts can be dragged and dropped from the ontology browser onto the map at the right hand side of Figure 2. Once the user has placed an event concept on the map, an event description

[4] `http://www.geonames.org/` [Last retrieved: August 04, 2010]
[5] `http://wordnet.princeton.edu/` [Last retrieved: August 04, 2010]
[6] `http://www.foaf-project.org/` [Last retrieved: August 04, 2010]

is created. To visualize the just created event description, the event-manager plugin of the SemaPlorer++ application selects an icon for the map view and displays it to the user. The example in Figure 2 shows an icon on the citymap of Sheffield, UK. It represents a major industrial fire that happened in a bakery.

The event-manager plugin uses the system date and time as default for creating the event description. They can be changed by the user. The location of the event is the latitude and longitude of icon's position on the map, which can be moved by the users. Additional information about the event is captured such as a title, location name, description, and images taken during the event. Once the representation of the emergency event is created, it is instantly stored in SemaPlorer++'s storage infrastructure. By this, the event description created by one user of the SemaPlorer++ application will be available to arbitrary other instances of the application in the complex, socio-technical system of emergency response, i.e., other emergency response entities.

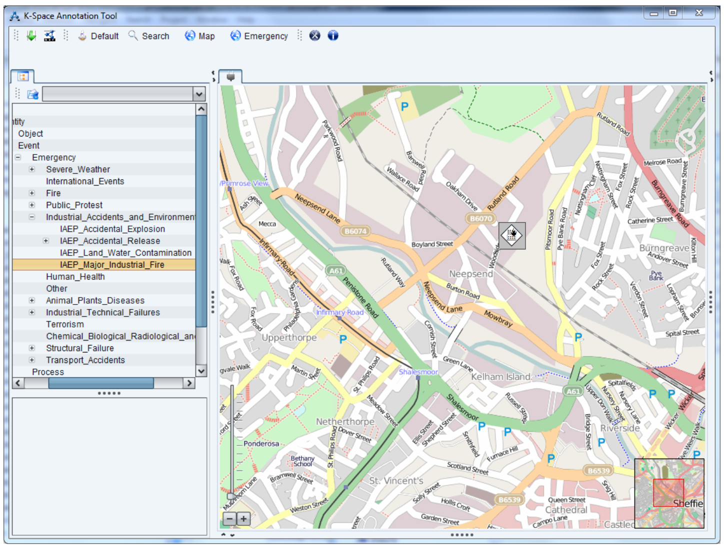
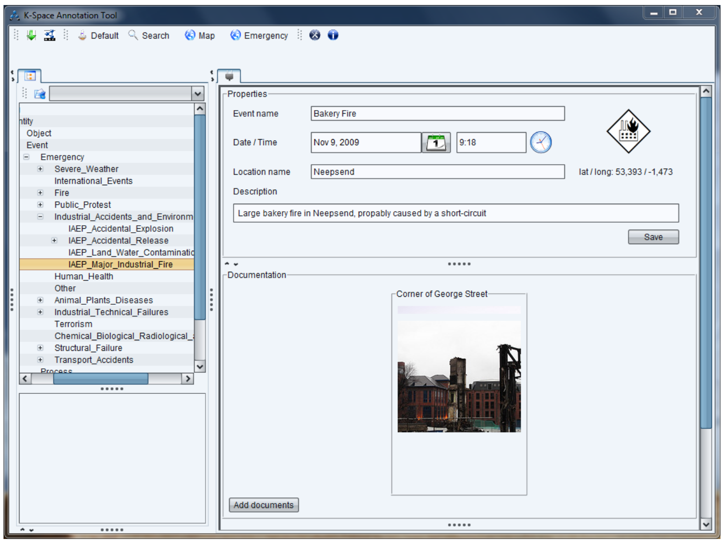


Fig. 2 Screenshot of the SemaPlorer++ Application Showing an Industrial Fire Event that Happened at a Bakery in the City of Sheffield, UK

When the user clicks on the icon, the SemaPlorer++ application changes from the map view to the event view to show the details of the industrial fire event as depicted in Figure 3. The figure shows the event details of the bakery fire in Sheffield that was caused by a short-circuit. In addition, an image from Flickr is shown documenting the event. The users of the SemaPlorer++ application can change the event title, location name, description, time, date, and documenting images. For documenting the event with images, the event-manager plugin allows the users to search for Flickr images that happened in the closeby location and time and associating them to the event as documentary support. To this end, the SemaPlorer++ application calls a semantic photo query service [35] provided by the WeKnowIt system with the parameters of the current time, the location of the event, and the event type, i.e., the emergency incident concept taken from emergency incident ontology. The semantic photo query service searches for appropriate Flickr photos within a specific time range and distance that match the given incident concept. For the SemaPlorer++ application, the time range is set to seven days of when the actual event happened. The radius of the location used to search for Flickr photos is determined by the semantic photo query service.

The photo query service called by the SemaPlorer++ application bases on an approach that combines semantic knowledge and statistical information as described in [35, Section 5.3]. Each concept of the emergency incident ontology is associated with a vector of weighted tags that are associated with the concept. The tags relevant for a concept are identified by exact string match, synonyms extracted from WordNet [16], and tag co-occurrence. When the SemaPlorer++ application calls the semantic photo query service, the Flickr photos of the requested time range and location area are classified by their tags using the vectors of the emergency incident concepts. The semantic photo query service returns a list of ranked photos most relevant to the given concept.



Event name: Bakery Fire
Date / Time: Nov 9, 2009 9:18
Location name: Neepsend lat / long: 53,393 / -1,473
Description: Large bakery fire in Neepsend, propably caused by a short-circuit

Fig. 3 Screenshot of the SemaPlorer++ Application with the Details and Photo Documentation of the Industrial Fire Event at a Bakery in Sheffield, UK

Besides using images from Flickr, the flexible infrastructure of the SemaPlorer++ application also allows to integrate and make use of arbitrary other media data from the Internet such as videos, blog entries, and tweets. Future extensions of the SemaPlorer++ application include providing trigger when new media data about an emergency incident is available on social media sharing platforms. In addition, reasoning on the data could be conducted like determining the buildings that need to be evacuated in a specific incident type such as a larger industrial fire or chemical accident. Here, among others sources like LinkedGeoData [3] could be used.

The emergency response scenario and SemaPlorer++ application have been developed based on the requirements of the professional emergency planning team of our project partner, the City Council of Sheffield, UK. To this end, an extensive requirement analysis has been conducted by visiting different emergency response entities such as the emergency hotline, control center, and fire department in Sheffield. In addition, we have conducted interviews with the emergency planning team of Sheffield to gather further requirements. These requirements have been refined regularly during the project meetings.

What we have learned from the emergency planning team in our project is that in the course of an incident communication is very difficult in the control room due to high noise as many people are gathering in the control room and become very active. During a larger incident, the emergency response coordination team needs to communicate with other emergency response entities such as the police and fire department. The emergency response entities communicate using phones, which is potentially error-prone and inefficient. In specific incidents also dedicated persons of the emergency response entities are in the control room to receive and dispatch messages. For example, during the Sheffield floods in 2007, a police officer was in the control room with responsibility for communicating between the control center and his colleagues out on the ground. Thus, computer support for exchanging emergency response information like with SemaPlorer++ is very welcomed. A scientific evaluation of the SemaPlorer++ application has yet not been conducted and remains part of our future work.

4 Requirements on Event-Model-F

We have derived functional as well as non-functional requirements on our Event-Model-F. These base on an analysis of existing event models and related work in Section 9, the emergency response scenario in Section 2, the SemaPlorer++ application in Section 3, and reported and own experience in designing core ontologies [39, 38, 2, 18].

To derive the functional requirements, we have analyzed existing models in various domains such as music [43], journalism [67], multimedia [15, 17, 70], news [22], cultural heritage [14], and knowledge representation [33]. The most comprehensive list of functional requirements are the six aspects defined for the event model E [70] and the journalism interrogatives of the Eventory system [67]. We have blend the aspects in E and interrogatives of the Eventory system and have synthesized them into our requirements. For each requirement, we also explicitly refer to the scenario in Section 2.

(1) Participation of objects in events. Representing participation of living and non-living objects such as people, animals, and other material objects in events and the roles they play in events. In the scenario, an example of objects participating in events and different object roles are to be found with (ii).

(2) Temporal duration of events and **(3) spatial extension of objects.** As events unfold over time (see Section 1), their temporal duration needs to be modeled. This can be conducted using absolute or relative representations of points in time. Objects unfold over space. Thus, modeling their spatial extension needs to be supported. This can be also modeled using absolute or relative positioning. In the emergency response scenario, events occur and objects exist at different points in time (vii) and at different locations (iv).

(4) Structural relationships between events. We consider three kinds of structural relationships between events, namely **(4a) mereological**, **(4b) causal**, and **(4c) correlation relationships**. The mereological relationship should be supported as events are usually made up of other events [42]. Causal relationships require the modeling of causes and effects and should support the integration and use of different causal theories as discussed, e.g., in [23]. Correlation refers to two events that have a common cause (cf. [55]). It should be supported as it is typically easy to observe, while causality is very difficult to discover and, hence, often unknown. The scenario illustrates the requirement of mereology (i), causality (v), and correlation (vi).

(5) Documentary support for events and objects. This requirement comprises the annotation of events and their participating objects with arbitrary information such as sensor data and media data. It provides documentary support that a particular event happened or object exists. In the scenario, the documentary support is indicated with (iii).

(6) Event interpretations. Relations between events such as causality and correlation can be matter of subjectivity and interpretation. For example, in a law suit the parties involved may each claim that the other one is at fault. Thus, the event model should support such different interpretations of the same event, i.e., provide different contextual points of view onto the same occurrences in the real world. In the scenario, event interpretation is indicated with (viii).

Besides the functional requirements discussed above, there are also a number of non-functional requirements to the Event-Model-F. They are derived from reported and own experience in knowledge-based systems and knowledge representation [39, 38, 2, 18]. The non-functional requirements comprise the extensibility of the Event-Model-F towards new developments and functional requirements. The Event-Model-F needs a sufficient formality and axiomatization so that systems can reason about the represented knowledge and carry out semantic checks on its validity. To decrease the complexity of the Event-Model-F, it needs to be modular. It shall be able to incorporate existing domain ontologies and make use of that existing domain knowledge. However, this domain-specific knowledge needs to be clearly separated from the structural knowledge captured with the Event-Model-F. An extensive discussion of the non-functional requirements has been conducted in [52].

5 Design of the Event-Model-F

For designing the Event-Model-F and implementing the functional requirements, we have carefully aligned it with the DOLCE+DnS Ultralite (DUL) ontology. DUL defines the class `DUL:Event` next to the disjoint upper classes `DUL:Object`, `DUL:Abstract`, and `DUL:Quality`. The definition of `Event` refers to an entity that exists in time (cf. discussion of events and objects in Section 1). The class `Object` stands for entities that exist in space such as living things as well as non-living and abstract things like social and cognitive entities. A `Quality` is a characteristic of an object or an event. It has a value that is represented as a point or area in some `Abstract`. The class `Abstract` represents value spaces, e.g., the space of natural numbers or the time of a day. In the Event-Model-F, we do not prescribe specific `Abstracts` that are to be used. We rather refer to the generic `Abstracts` already defined in DUL such as the regions `DUL:TimeInterval`, `DUL:SpatioTemporalRegion`, and `DUL:SpaceRegion`.

The functional requirements on our Event-Model-F are represented by specialized instantiations of the descriptions and situations (DnS) ontology pattern that is part of DOLCE+DnS Ultralite. We use DnS as it provides formally precise representations of different, contextualized views on events [19, 20]. Thus, with DnS one can reify events and describe the n-ary relation that exists between multiple individuals of events and objects. We use the DnS pattern as the representation of occurrences in the real world (i.e., the events and objects we are modeling) are subject to discussion and interpretation and may not be objectively observable. The DnS pattern allows for representing different opinions about events and their participating objects. This feature is not provided by the DOLCE+DnS Ultralite participation relation, namely the property `isParticipantIn`. The property defines fixed relations between `Event`s and `Object`s. Thus, no further interpretations or discussions of, e.g., the objects participating in an event are possible.[7] This is not desirable for the domains we consider such as emergency response, sports, news, law, and others.

In the following, we introduce the ontology patterns of the Event-Model-F and illustrate them in diagrams. With respect to the functional requirements in Section 4, the participation of objects in events (1) is implemented by the participation pattern. It also provides for modeling the absolute time and location of events (2) and objects (3). The mereology pattern, causality pattern, and correlation pattern implement the structural relationships between events (4a-4c). In addition, the mereology pattern allows for modeling the relative temporal relations and relative spatial relations between events (2) and objects (3). The documentation pattern provides for annotating events (5) and the interpretation pattern supports different event interpretations (6). Classes defined by the Event-Model-F are highlighted in the diagrams to show the alignment with classes of DUL. The Event-Model-F is axiomatized in Description Logics [4] using the Web Ontology Language (OWL) [40]. It is available online at: `http://west.uni-koblenz.de/eventmodel`.

5.1 Participation Pattern

The participation pattern enables to formally express the participation of objects in events. As shown in Figure 4(a), participation is expressed by an `F:EventParticipationSituation` that `satisfies` an `F:EventParticipation-Description`. The situation includes the `Event` being described and the `Objects` participating in this event. The `EventParticipationDescription` classifies the described event and its participants by using the concepts `F:DescribedEvent` (specialized from `DUL:EventType`) and the object role `F:Participant` (specialized from `DUL:Role`). The concept `DescribedEvent classifies` the `Event` that is described by the participation pattern, e.g., the event of a flooded cellar. Likewise, instances of `Participant` classify objects as participants of the event. For example, the citizen calling the emergency hotline to report about the flooded cellar. Instances of `Participant` can be roles defined in some domain ontology as indicated in Figure 4(a). For example, an emergency response ontology may define the role of a person being affected, i.e., the emergency subject, and the role describing the rescue staff such as firemen. Besides the role an object can play in a specific participation pattern, also the described event and its participating objects themselves can be defined in some domain ontology as indicated in Figure 4(a).

The parameter `F:LocationParameter` describes the general spatial region where the objects are located. It `DUL:parametrizes` a `SpaceRegion` and defines a property `DUL:isParameterFor` to the `Participant` role. The `Object` that is classified by the `Participant` has a `Quality` with the property `DUL:hasRegion` of a `SpaceRegion`. Thus, using the `F:LocationParameter` we can define the location(s) represented by `SpaceRegion`s that are relevant for describing the event in a given context. For example, when quenching a house fire all firemen have their specific location within and

[7] DOLCE provides a similar property between endurants and perdurants (objects and events) called `participant-in`.

around the building. The `LocationParameter` can then be used to describe in general that the firemen where at that specific house, e.g., in form of some longitude-latitude rectangular. Thus, we do not need to explicitly state or even know where the individual firemen are. The `F:LocationParameter` is typically some aggregation of the objects' locations. The `F:TimeParameter` describes the general temporal region when the event happened. It `parametrizes` a `TimeInterval` and defines a property `isParameterFor` to the `DescribedEvent` role. For example, one can state that the house fire happened on June 13, 2006.

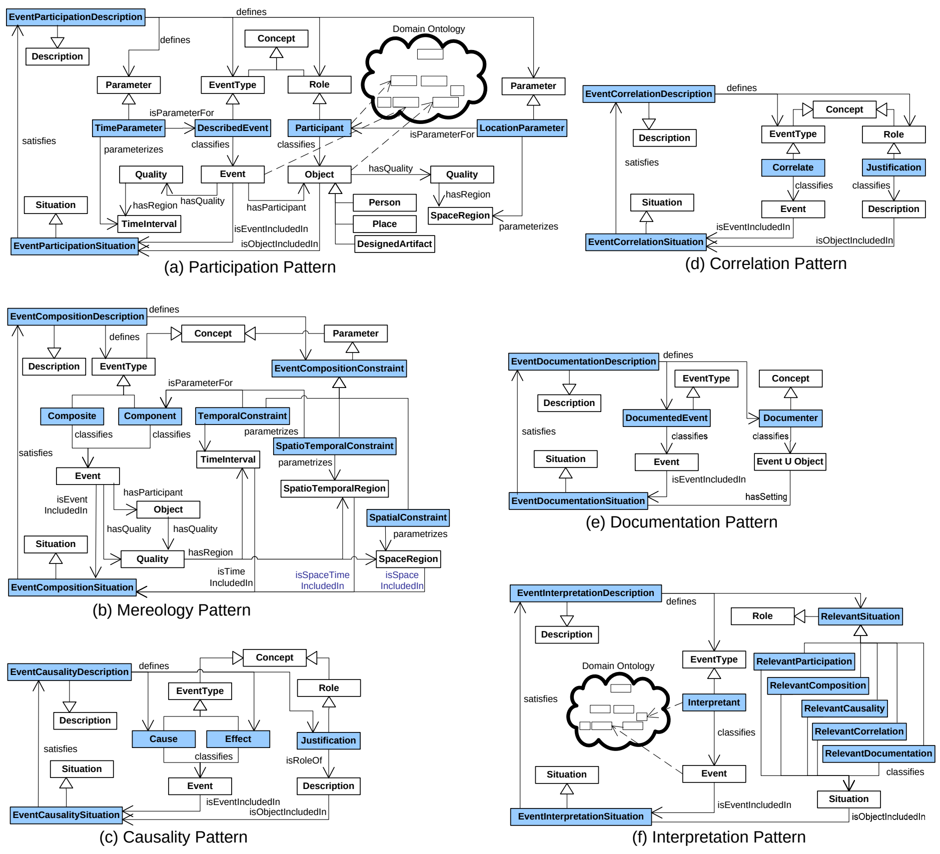


Fig. 4 The patterns of F, namely (a) participation, (b) mereology, (c) causality, (d) correlation, (e) documentation, and (f) interpretation

5.2 Mereology Pattern

Events are commonly considered at different abstraction levels depending on the view and the knowledge of a spectator. For instance, the event of a flooded cellar may be considered as such or as part of the larger event of a flooding in which many other (smaller) incidents occur. The mereology pattern shown in Figure 4(b) enables expressing such mereological relations as composition of events. The composite event is the "whole" and the component events are its "parts". Formally, a `F:EventCompositionSituation` includes one instance of an event that is classified by the concept `F:Composite` and many events classified as its `F:Component`(s). Accordingly, an `EventCompositionSituation` `satisfies` a `F:CompositionDescription` that `defines` the concepts `Composite` and `Component` for classifying the composite event and its component events.

Events that play the `Component` role may be further qualified by temporal, spatial, and spatio-temporal constraints. As events are formally defined as entities that exist in time and not in space (cf. Section 1), constraints including spatial restrictions are expressed through the objects participating in the component event. For instance, a `Component` event may be required to occur within a certain time-interval, e.g., the second week of June 2009. Depending on its objects, a `Component` event may also happen in a certain spatial region. For example, the flooding of a town should be composed of events that have objects associated to it, which have some certain range of longitude and latitude. Finally, events and the objects bond to it may be qualified by a spatio-temporal quality like the progress of a flood that extents over time and space, starting with a high water level located in some area of a river and extending spatially over time into other areas. Any such constraints are formally expressed by one or multiple instances of the `F:EventCompositionConstraint`. They define qualifying values for temporal, spatial, and spatio-temporal qualities of a component event as shown in Figure 4(b). Thus, with the composition pattern, events may be arbitrarily temporally related to each other, i.e., they might be disjoint, overlapping, or otherwise ordered. In order to express such relative temporal relations between events, one can facilitate the provided means of DOLCE such as the formalization of Allen's Time Calculus[8].

5.3 Causality Pattern

Causality is the traditional philosophical problem investigating the existence (or nonexistence) of any special "tie" binding causes and effects together [23]. It can be questioned either as "Why" or "How" [23]. An answer to this question is (or better said claims to be) a causal explanation. What explains, is the cause and that what is explained is the effect [23]. Events are the most natural concept to serve for defining causal relations [42]. In fact, causes and effects are two specific types of events [23, 27] and only events can play the roles cause and effect, respectively [27].[9] Causation is unidirectional [23], i.e., causes are bringing about effects in an orderly succession, and causes are a necessary condition of its effect [23]. A causal relationship is always justified by some (maybe implicit) underlying causal theory.

Based on the analysis of related work and discussion above, we designed a causality pattern as depicted in Figure 4(c). The pattern defines two `EventType`s called `F:Cause` and `F:Effect` which `classify` `Event`s. It further defines a `DUL:Description`, which is classified by a `F:Justification`. By this, the pattern explicitly expresses the causal relationship between the cause and the effect under the justification of some theory. A theory might be an opinion, a scientific law, or not further specified. For example, during a heavy storm, a power outage might occur caused by a snapped power pole.

[8] `http://wiki.loa-cnr.it/index.php/LoaWiki:Ontologies` [Last retrieved: August 04, 2010]

[9] In the case of sociology, causes and effects are states [23].

The `Justification` of this causal relationship is the laws of physics. During the power outage, also some citizens are calling the emergency hotline. This causal relationship would be justified by the social norm, which allows citizens to call the emergency hotline in cases of emergency.

As defined above, causes and effects are events. Thus, we assume that objects inherently involved in causal relationships between events are properly associated to the cause and effect by the participation pattern (see Section 5.1). The causality pattern defines that exactly one cause is tied with one effect. If there are multiple causes that bring a common effect or if a cause implies multiple effects, the mereology pattern is applied to create an aggregated cause and aggregated effect, respectively. Thus, an aggregated cause or aggregated effect is a set of events that are classified as cause or effects, respectively. As a discussion of different causal theories shows [23], not all theories clearly separate objects from events as F does. In cases, where events and objects are merged into one concept, we assume that the causal "tie" modeled in the theory is applied to the event and that the objects are bound to it using the participation pattern. This allows for describing the participation of objects relevant in the context of a given causal relationship by using the causality pattern in combination with the participation pattern from Section 5.1. Finally, according to Shafer [54] there are different kinds of causal relationships. Thus, using a single causality relationship might not be precise enough. Consequently, her logic for causality allows for explicitly modeling the different kinds of causal relationships found in literature. This can be supported with our Event-Model-F by specializing the causality pattern to specific kinds of causal relationships.

5.4 Correlation Pattern

A set of events is called correlated if they have a common cause. However, there exists no causal relationship between the two events [55]. The common cause may originate from a single or a chain of multiple preceding cause-effect relationships. Correlation also differs from co-occurrence where two or more events just (randomly) happen at the same time and do not have a common cause.

Correlation is not of metaphysical interest as it is a property that can be derived from causality, i.e., the common cause. In our ontology, we model correlation explicitly as in many cases the (correlating) effects of some common cause may be known, while the cause itself is not. The correlation pattern depicted in Figure 4(d) defines the role `F:Correlate` to `classify` the events that are correlated. The `Justification` role classifies some `Description`, which explains the correlation in terms of a (mathematical) law or some theory.

5.5 Documentation Pattern

Documentary evidence for an event may be given by arbitrary objects, e.g., some sensor data, media data, or by other events. Formally, this relation is expressed by the documentation pattern depicted in Figure 4(e). It defines the concept `F:DocumentedEvent` that classifies the documented event and the concept `F:Documenter` that classifies the documentary evidence for that event. This evidence can be expressed by any specialization of an `Object`, e.g., a digital photo taken with a cell phone during an incident, or a specialization of `Event`. Thus, the documentation pattern enables the inclusion of evidences described in ontologies other than F. For example, digital media data like images and videos can be classified as documenting `Entity` and precisely described using, e.g., Multimedia Metadata Ontology [45]. Thus, the documentation pattern intriguing connects the representation of the real-world events in which the humans participate with the media assets that were taken during these events. The objects are documented via the events in which they participate (see participation pattern in Section 5.1).

5.6 Interpretation Pattern

The perception of events as occurrences in the real world heavily depends on the context and point of view of the observer. Such different, context-dependent event interpretations can be described formally by instantiating the different Event-Model-F patterns presented so far and binding them together with the interpretation pattern depicted in Figure 4(f). Each pattern models a single, specific interpretation of an event by associating *participations*, *mereological*, *causal*, and *correlative* relationships, as well as *documentations* relevant in the context of a specific *interpretation*. In the emergency use case, two emergency control officers might have differing interpretations of the power outage. One might be convinced that the power outage is due to a snapped power pole, while the other might think of a more serious case of a damaged power plant. Both consider the same event of a power outage, however, consider it from different points of view that involve other events and objects in different patterns.

Formally, the interpretation pattern shown in Figure 4(f) defines a `F:Interpretant` that is specialized from `EventType` and classifies the interpreted `Event`. The `Interpretant` might be defined in some domain ontology and determines how an event is interpreted, e.g., as emergency incident in the case of the emergency control center or as news event described in a news paper. Within each interpretation, we classify the `F:RelevantSituations`, namely the situations satisfying the participation, mereology, causality, correlation, and documentation. These are defined as specializations of `RelevantSituation`, namely the sub-classes `F:RelevantComposition`, `F:RelevantParticipation`, `F:RelevantCorrelation`, and `F:RelevantCausality`.

5.7 Summary

We designed a formal model of events F based on the foundational ontology DOLCE+DnS Ultralite. The functional requirements to the Event-Model-F are fulfilled by introducing ontology design patterns based on the Descriptions and Situations pattern. By this, we can represent arbitrary occurrences in the real world and formally model the different relations and interpretations of events. We fulfill the non-functional requirements stated in Section 4 by the use of this ontology and following a pattern-oriented design approach.

6 Axiomatization of the Event-Model-F

The Event-Model-F is axiomatized in Description Logics using OWL. This axiomatization has been conducted for each ontology design pattern as they have been introduced in Sections 5.1 to 5.6. In the following, we show an excerpt of the axiomatization of the causality pattern. The full axiomatization of the Event-Model-F is presented in Appendix A.

Like all patterns of the Event-Model-F, also the causality pattern bases on the DnS pattern of DUL as introduced in Section 5. Lines (1) to (5) show the axioms of the description part of the causality pattern. The first line is stating that the `EventCausalityDescription` is a `Description` from DOLCE+DnS Ultralite. Line (2) specifies that the `EventCausalityDescription` exclusively defines the three roles `Cause` and `Effect` and `Justification` and only these roles. With lines (3) to (5), the cardinality of the roles `Cause` and `Effect` and `Justification` within a single instantiation of the causality pattern is defined. Finally, line (6) specifies that the `EventCausalityDescription` is `satisfiedBy` exactly one `EventCausalitySituation`.

$$(1) EventCausalityDescription \sqsubseteq Description$$
$$(2) EventCausalityDescription \sqsubseteq \forall defines.(Cause \sqcup Effect \sqcup Justification)$$
$$(3) EventCausalityDescription \sqsubseteq = 1(defines.Cause)$$
$$(4) EventCausalityDescription \sqsubseteq = 1(defines.Effect)$$
$$(5) EventCausalityDescription \sqsubseteq = 1(defines.Justification)$$
$$(6) EventCausalityDescription \sqsubseteq = 1(satisfiedBy.EventCausalitySituation)$$

Continuing with the situation part of the causality pattern, line (7) specifies that the `EventCausalitySituation` is a `Situation` from DOLCE+DnS Ultralite. Lines (8) and (9) specify that each `EventCausalitySituation` in the Event-Model-F includes exactly two events, one `Cause` and one `Effect`. In addition, line (10) specifies that exactly one `Justification` is included. Finally, it is specified in line (11) that the `EventCausalitySituation` satisfies exactly one `EventCausalityDescription`.

$$(7) EventCausalitySituation \sqsubseteq Situation$$
$$(8) EventCausalitySituation \sqsubseteq = 1(includesEvent.(\exists isClassifiedBy.Cause))$$
$$(9) EventCausalitySituation \sqsubseteq = 1(includesEvent.(\exists isClassifiedBy.Effect))$$
$$(10) EventCausalitySituation \sqsubseteq = 1(includesObject.(\exists isClassifiedBy.Justification))$$
$$(11) EventCausalitySituation \sqsubseteq = 1(satisfies.EventCausalityDescription)$$
$$\ldots$$

When specifying ontology design patterns semantically precisely like the causality pattern of the Event-Model-F above, one encounters the question how many axioms are needed in order to sufficiently define the semantics of a pattern. The patterns of the Event-Model-F such as participation, causality, and so on base on the DnS pattern from DOLCE+DnS Ultralite (see Section 5). As we have seen in Sections 5.1 to 5.6 and the example axiomatization of the causality pattern above, these DnS-based patterns are specified in a way that the `Description` defines a number of concepts that classify entities within the context of the pattern, namely `Events` and `Objects`. Each concept might only classify exactly one entity. Thus, the concepts classifying the entities constitute the roles the entities play within the context of a specific pattern.

This is required to ensure a semantically precise specification and use of the patterns. In addition, the `Description` defines that the pattern only comprises this specific set of roles and no further roles. By this, a semantically precise use of the ontology design pattern is ensured.

For example, the axiomatization of the `EventCausalityDescription` in line (2) of the causality pattern exclusively defines the three roles `Cause`, `Effect`, and `Justification` and only these roles. With this so-called *closure* axiom, it is not possible to define and use other roles than `Cause`, `Effect`, and `Justification` within this pattern. For example, an ontology engineer might want to use the pattern in a context in which it is not designed for. He might want to add some concepts that do not fit the intended design of the causality pattern. Thus, a high amount of axiomatization reduces the risk of wrongly applying the patterns. On the other hand, it might be interesting in future to extend the patterns and provide support for additional information like provenance, i.e., meta-information about the method and parameter setting by which the concrete pattern instantiations have been created.

To add provenance information to the ontology design patterns of the Event-Model-F, different options are possible. For example, a new provenance pattern could be created. This provenance pattern would base on DnS and define a provenance role and provenance parameter. To add provenance information to the existing patterns of the Event-Model-F such as the participation pattern and causality pattern, they would need to "piggyback" the provenance pattern. This means that the provenance role and provenance parameter of the provenance pattern would be added to the roles defined by the patterns of the Event-Model-F. One would need to extend the current axiomatization of the Event-Model-F patterns and allow to include the roles of the provenance pattern. To this end, the closure axioms of the ontology patterns need to be modified like the closure axiom of the causality pattern in line (2). Another option to provide support for provenance in the Event-Model-F is to associate the provenance information to the `Situation` concept of the patterns. For example, the provenance role and provenance parameter would be associated with the `EventCausalitySituation` concept to express provenance information with the causality pattern.

It is a principal design decision which option for adding provenance information is chosen and is beyond the scope of the Event-Model-F. However, the discussion shows that the amount of axiomatization is dependent on how much one wants to exactly define and restrict the pattern's context of use and how open it shall be for future extensions. For the Event-Model-F, we have decided to provide a high amount of axiomatization in order to prevent the ontology design patterns from being wrongly applied. The axiomatization also helps us in designing and providing an application programming interface for the ontology, which is described next.

7 Programming Interface for the Event-Model-F

We have developed an application programming interface (API) for the Event-Model-F in order to integrate and use it for the development of concrete applications that make use of events. The API is based on the formal specification of the ontology and provides full functionality of the Event-Model-F through easy to use Java interfaces. It implements the desired behavior of the ontology and its axiomatization and provides a usefull abstraction of the Event-Model-F to the application developer. To this end, the API hides the actual design and axiomatization of the ontology from the application developer in order that they can concentrate on the actual application development tasks. The application developers can directly use the Event-Model-F without knowing the internal details of the ontology or having knowledge in ontology design at all.

Due to the pattern-based design of the Event-Model-F, it is not straightforward to develop an API for it. One of the biggest challenges when implementing an API for a pattern-based ontology is to cope with the nature and complexity of ontology design patterns. Today's tools for ontology API generation[10] such as Elmo[11] and Owl2Java[12] typically map all concepts and properties of the ontology to the API. However, they do not provide explicit support for ontology design patterns. Providing support for pattern-based ontologies raises a couple of challenges to the design and implementation of the API.

The patterns of the Event-Model-F for participation, composition, correlation, causality, documentation, and interpretation base on DnS. In general, the DnS pattern foresees a `Description` that defines `Concepts` (see Section 5). The `Concepts` classify some `Entity`, i.e., `Events` or `Objects`. The entities have the property `hasSetting` that is located in a `Situation`. A loop is established in the DnS pattern by a `satisfies`-property from the `Situation` to the initial `Description` concept. An API for such DnS-based ontologies like the Event-Model-F has to guarantee the consistency of the pattern instantiations for creation, read, update, and delete operations. Without explicit support for ontology design patterns in the API, the consistency of the data cannot be ensured. For example, for the causality pattern the API has to ensure that the individuals for the `EventCausalityDescription` and `EventCausalitySituation` concepts are provided. It has to ensure that there is exactly one individual of type `Cause` and one individual of type `Effect` and so on (see axiomatization of the causality pattern in Section 6). It is important to note that a reasoner would not fail when one of the individuals above is missing. A reasoner would—according to the open world assumption—just assume that it does not know the individual and has no information about it. However, such a behavior is not desired at the API-side when implementing a concrete application using the ontology. When creating an instance of a pattern like the causality pattern it should be ensured that all mandatory information is provided.

In addition, APIs for ontologies using patterns like DnS do not need to provide all concepts of the ontology through distinct classes in the API to the application developer. Not all concepts in the patterns of the Event-Model-F are of importance for the application developer. For example, the `Description` and `Situation` concepts can be automatically generated by the API in the background and thus hidden from the application developers. As a consequence, not all concepts of the ontology design patterns should be visible to the application developers by the API as they are with today's tool support for API generation.

In order to provide an effective and efficient access to the Event-Model-F, we have manually designed and implemented the API in Java. For designing the Event-Model-F API, we have developed and implemented a set of best practices. These best practices are based on the experiences gained with the development of APIs for other core ontologies like COMM [2] and X-COSIMO [18]. The fundamental design decision for the Event-Model-F API is to follow a layer oriented design approach as depicted in

[10] A good overview provides the following website: `http://semanticweb.org/wiki/Tripresso` [Last retrieved: August 04, 2010]
[11] `http://www.openrdf.org/doc/elmo/1.3/` [Last retrieved: August 04, 2010]
[12] `http://www.incunabulum.de/projects/it/owl2java` [Last retrieved: August 04, 2010]

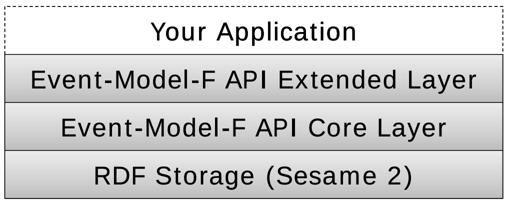

Your Application
Event-Model-F API Extended Layer
Event-Model-F API Core Layer
RDF Storage (Sesame 2)

Fig. 5 The Layered Architecture of the Event-Model-F API

Figure 5. It consists of three layers, the triplestore layer, core layer, and extended layer. On top of the extended layer is the application layer representing the concrete application using the Event-Model-F.

We briefly describe the design of the Event-Model-F API along its layers from bottom to top: The task of the triplestore layer is to provide an efficient and effective persistence support for the Event-Model-F. The triplestore layer is implemented using the Sesame 2 triplestore[13].

The core layer of the API provides a direct mapping of the Event-Model-F ontology to Java objects. It provides a single Java class for each pattern in the ontology such as `EventCausalityPattern.java` for the causality pattern. These pattern classes provide an attribute for all concepts defined in the pattern. For example, attributes for the concepts `Cause` and `Effect` are provided as well as for `EventCausalityDescription` and `EventCausalitySituation`. Thus, the design of the core layer of the Event-Model-F is very close to the actual design of the ontology itself. The core layer of the Event-Model-F API is not designed to be used by the application developers directly. Its purpose is to help dealing with the complexity of the pattern-based ontology and providing an initial API implementation for it. The core layer ensures consistency of the data stored with the API, i.e., ensures that the instantiations of the ontology patterns of the Event-Model-F are consistent with respect to their specification in Sections 5.1 to 5.6.

Finally, the extended layer of the Event-Model-F API provides a specialization of the core layer. The purpose of this layer is to abstract from the concrete design of the pattern-based core ontology and to provide the Java interfaces and classes the actual application developer has to deal with. Thus, the goal of the extended layer is to hide the complexity of the ontology and to provide the application developer the functionality he needs to create, modify, and delete event representations and event relations. To this end, the Java classes implementing the distinct ontology pattern on the core layer are specialized on the extended layer to provide more natural and easy use. The extended layer hides concepts from the application developer he does not need to deal with such as `Description` and `Situation`. In addition, the extended layer provides a set of generic but often required convenience methods. The extended layer is aimed to be specialized to domain-specific requirements of the application.

Our experience in using pattern-based ontologies in concrete applications and designing APIs for such ontologies is that they should not have a fixed programming interface. Although we can assume that the design of a core ontology is very stable with respect to its use in different domains, the API should be redesigned based on the requirements of the concrete domain in which it is used. By this, one can enable

[13] `http://www.openrdf.org/` [Last retrieved: August 04, 2010]

a much more efficient access to the knowledge modeled and represented with the core ontology. Thus, the extended layer should be adapted to the concrete requirements of the application domain. Consequently, methods and tool support are required to provide a rapid design and development of domain-specific APIs for ontologies.

A detailed description of the design and implementation of the Event-Model-F API and the best practices extracted from developing the API have been summarized in [53]. The source code of the Event-Model-F API is released under open source license and is available from `https://launchpad.net/eventmodelf`.

8 Application of the Event-Model-F in the SemaPlorer++ Application

We have introduced the SemaPlorer++ application for creating and sharing event descriptions and associated media data in the context of emergency response in Section 3. The SemaPlorer++ application makes use of the Event-Model-F API introduced above. To this end, the API has been specialized towards the domain-specific requirements of emergency response. For describing events in SemaPlorer++, the different patterns defined in the Event-Model-F are combined, each providing a specific part of the event description. As described in Section 3, the SemaPlorer++ application provides an event-manager plugin designed to deal with events. The communication between the SemaPlorer++ application and the event-manager plugin is handled via the messaging bus of the K-Space Annotation Tool. When dragging and dropping an emergency incident concept from the ontology browser on the map, a message object is send to the event-manager plugin. The message contains an object called `table` and provides all information required by the event-manager plugin. First, the event-manager plugin checks whether the message is an event. If it is a subclass of `DUL:Event`, the latitude and longitude of the event is retrieved from the `table` object. Otherwise the message is ignored. If the concept is of type `DUL:Event`, an instantiation of the participation pattern of the Event-Model-F is created and stored in the Sesame triplestore through the Event-Model-F API. The participation pattern comprises the time and location of the event and a default object participating in the event. The pattern is filled with latitude and longitude information extracted from the `table` object and stored in an instance of the class `GeoLocatedEvent`. The class `GeoLocatedEvent` is part of the extended layer of the Event-Model-F API that has been specialized for SemaPlorer++. The `GeoLocatedEvent` stores also further information such as the date and time when the event happened, a title, location name, description, and images documenting the event.

Figure 6 shows how a fictive industrial fire event that happened in a bakery in Sheffield is captured using the participation pattern. The pattern comprises the individuals for the situation, description, and roles of the participation pattern. The fire event is represented with the individual `fire-event-1` of incident type `Major_Industrial_Fire_Event` (`MIFEvent`). The time and date of the event is modeled using ISO8061. Thus, the `TimeParameter` `parametrizes` the abstract `ISO8061DateTime` that has a property `xsd:dateTime` with the concrete values of the date and time of the fire event, namely `2009-11-01 08:46:00,00-532`. The spatial information of the fire event is added to the participating object using the `LocationParameter`. It `parametrizes` the abstract `WGS84SpaceRegion` that allows modeling the location along the WGS84 vocabulary[14]. The WGS84 vocabulary defines two properties for latitude and longitude. In the concrete example of the fire event, the `geo:lat` is `53,389` and `geo:long` is `-1,453`.

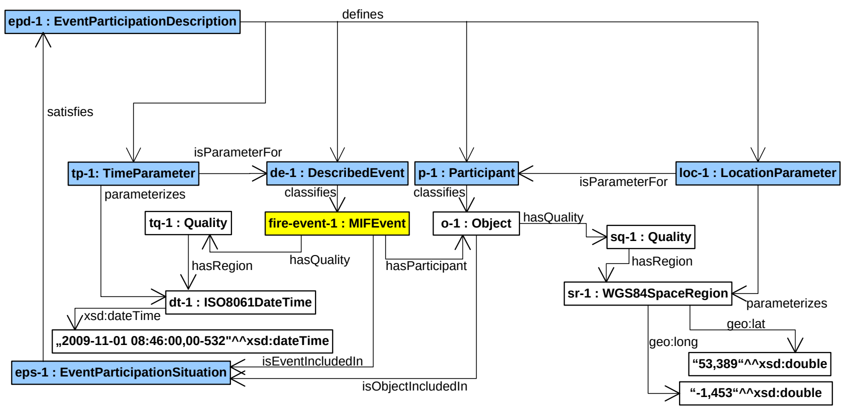


Fig. 6 Instantiation of the Participation Pattern Representing an Industrial Fire Event that happened at a Bakery in Sheffield

In addition, the fire event is documented with a title, written location name, and description. This is represented using a specialization of the documentation pattern shown in Figure 7. It defines the properties `hasEventNameValue`, `hasEventDescriptionValue`, and `hasEventLocationValue` of the `DocumentingEntity`. The connection to the fire that happened in Sheffield and its representation in the participation pattern as shown in Figure 6 is established by using the same individual `fire-event-1` in both patterns.

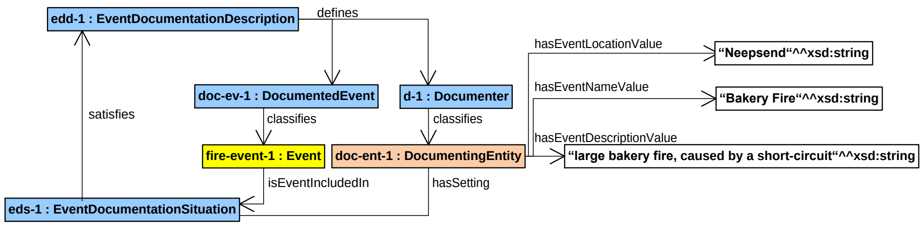

edd-1 : EventDocumentationDescription
defines
hasEventLocationValue
"Neepsend"^^xsd:string
satisfies
doc-ev-1 : DocumentedEvent
d-1 : Documenter
hasEventNameValue
"Bakery Fire"^^xsd:string
classifies
classifies
hasEventDescriptionValue
fire-event-1 : Event
doc-ent-1 : DocumentingEntity
"large bakery fire, caused by a short-circuit"^^xsd:string
isEventIncludedIn
hasSetting
eds-1 : EventDocumentationSituation

Fig. 7 Instantiation of a Specialized Documentation Pattern for Emergency Response Representing the Location, Name, and Description of the Industrial Fire Event

Finally, images documenting the fire event can be associated to the `fire-event-1` by using another specialization of the documentation pattern as shown in Figure 8. This specialization introduces a `PhotoEntity` as `Documenter` and allows to annotate events with a photo URL using the `hasPhotoURL` property and a title describing the image with the `hasPhotoTitleValue` property. In the example of the fire event in Sheffield, an image from Flickr is used to document the event.

[14] `http://www.w3.org/2003/01/geo/` [Last retrieved: August 04, 2010]

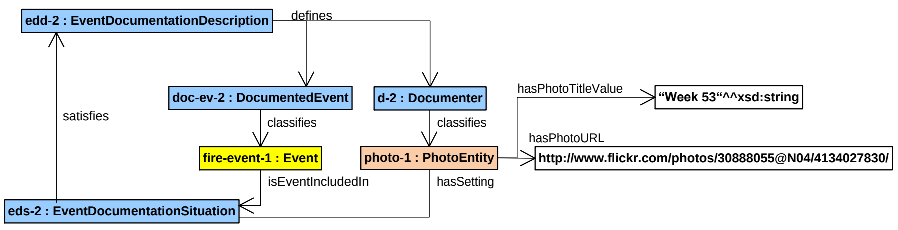


Fig. 8 Instantiation of a Specialized Documentation Pattern Representing the Annotation of the Industrial Fire Event with an Image from Flickr

The Event-Model-F is well suited for implementing the SemaPlorer++ application. By the modular design, i.e., splitting up the functional features of the Event-Model-F into different ontology design patterns, the engineers can easily choose the parts of the ontology, i.e., the patterns they need for their application and can conveniently specialize them towards domain-specific requirements. The formal nature of the Event-Model-F supports this specialization process by reducing the risk to introduce ambiguities (cf. discussion in Section 6). In addition, the Event-Model-F allows to incorporate existing domain knowledge such as the emergency incidents ontology from Sheffield.

Besides the examples above, we have developed further examples demonstrating the use of the Event-Model-F for representing emergency response events. These examples show among others the application of the mereology pattern, causality pattern, and interpretation pattern. They can be downloaded from our website at `http://west.uni-koblenz.de/eventmodel` and are described in detail in [52, 53]. In an extended version of the SemaPlorer++ application, the features for mereology and causality will be available to the end users. In addition, different points of view onto the events will be supported by leveraging the interpretation pattern.

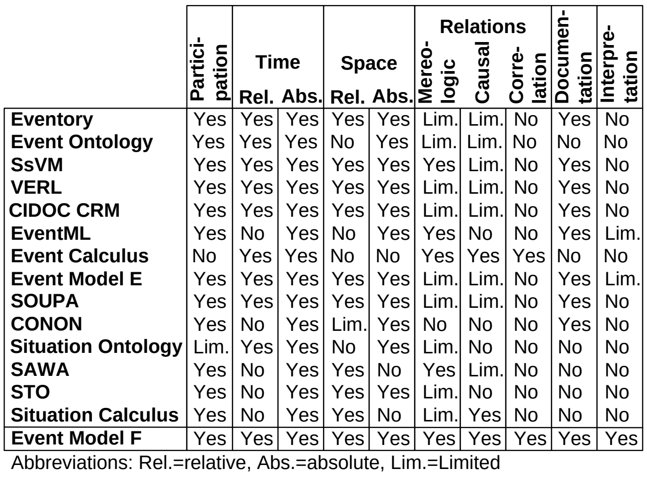
The Event-Model-F can also be applied to arbitrary other domains that need to represent events as occurrences in which humans participate. Thus, in addition to the application of the Event-Model-F for emergency response, the website above provides examples in the domains of tourism and soccer. The tourism example models a two-day weekend trip. It is like the emergency response scenario motivated by the WeKnowIt project. Three people are participating in this trip. On the first day, there is a sub-event of a dinner. On the second day are two sub-events, a visit to a museum and a travel to a sight. The tourism example applies different patterns of the Event-Model-F such as participation and composition. The soccer example models an entire game, i.e., the first halftime and second halftime. Different events happen during the game such as a foul and goal. The soccer example makes full use of all patterns of the Event-Model-F, namely participation, causality, correlation, composition, and interpretation. It further shows how a domain specific ontology can be embedded and used to describe the events happening during a soccer game.

9 Related Work

The discussion of related work is twofold. In Section 9.1, we analyze the existing event-based systems and event models and compare them with the Event-Model-F. We also consider models for situation awareness and situation theory, which have a close relation to events. In Section 9.2, we review commercial and open source command-and-control systems for managing emergency incidents and elaborate on the advantages using the Event-Model-F to enable interoperability between these systems.

9.1 Discussion of Existing Event Models

For designing the Event-Model-F, we have analyzed existing event-based systems and event models. We have also analyzed related models for situation awareness and situation theory that have a close relation to events and provide event-like structures. The existing models and systems have been systematically studied with respect to the functional requirements in Section 4. The event-based system and event models are motivated from different domains such as the Eventory [67] system for journalism, the Event Ontology [43] as part of a music ontology framework, the ISO-standard of the International Committee for Documentation on a Conceptual Reference Model (CIDOC CRM) [14, 56] for cultural heritage, the event markup language EventML [22] for news, the event calculus [33, 11] for knowledge representation, the Semantic-syntactic Video Model (SsVM) [15] and (VERL) [17, 36] for video data, and the event model E [51, 70] for event-based multimedia applications. With respect to situation awareness and situation theory, we find the Standard Ontology for Ubiquituous and Pervasive Computing (SOUPA) [12], which is the core of the Context Broker Architecture (CoBrA) [13], the Context Ontology (CONON) [68] for modeling context in pervasive computing environments, the Situation Ontology [72] for an hierarchical modeling and sharing of situation knowledge, the Situation Awareness Assistant (SAWA) application [32, 30, 31, 29], the Situation Theory Ontology (STO) [24], and the situation calculus [26, 25] for representing changes in the real world. An overview of the analysis results and comparison to the features of our Event-Model-F along the functional requirements is shown in Figure 9.



	Partici-pation	Time		Space		Relations			Documen-tation	Interpre-tation
		Rel.	Abs.	Rel.	Abs.	Mereo-logic	Causal	Corre-lation		
Eventory	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	No
Event Ontology	Yes	Yes	Yes	No	Yes	Lim.	Lim.	No	No	No
SsVM	Yes	Yes	Yes	Yes	Yes	Yes	Lim.	No	Yes	No
VERL	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	No
CIDOC CRM	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	No
EventML	Yes	No	Yes	No	Yes	Yes	No	No	Yes	Lim.
Event Calculus	No	Yes	Yes	No	No	Yes	Yes	Yes	No	No
Event Model E	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	Lim.
SOUPA	Yes	Yes	Yes	Yes	Yes	Lim.	Lim.	No	Yes	No
CONON	Yes	No	Yes	Lim.	Yes	No	No	No	Yes	No
Situation Ontology	Lim.	Yes	Yes	No	Yes	Lim.	No	No	No	No
SAWA	Yes	No	Yes	Yes	No	Yes	Lim.	No	No	No
STO	Yes	No	Yes	Yes	Yes	Lim.	No	No	No	No
Situation Calculus	Yes	No	Yes	Yes	No	Lim.	Yes	No	No	No
Event Model F	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Abbreviations: Rel.=relative, Abs.=absolute, Lim.=Limited

Fig. 9 Comparison of Existing Models for Events Towards Functional Requirements

Our analysis shows that the existing systems and models almost fully support the participative, temporal and spatial aspects. Many of the existing solutions also support the documentary aspect. However, existing models substantially lack in supporting the structural aspect, i.e., mereological, causal, and correlation relationships, and representation of different interpretations of the same event. Here, we find different variations of limitations or even no support by the existing models. With respect to mereological event relationships, the existing models typically provide support for simple part-of relationships such as with the `sub_event` property of the Event Ontology, an informal description of a mereological relationship in Eventory, and hierarchical part-whole relationships in CIDOC CRM [56]. However, no further axiomatizations are provided for refining the mereological relationship by different criteria such as temporal and spatial constraints. Only SsVM allows for describing more complex mereological relationships and the situation-awareness model of SAWA supports spatio-temporal composition [5]. Similar, also the support for causal relationships is limited in most of the existing models. It is typically defined as a cause-effect relationship like the `factor` and `product` properties of the Event Ontology or the "resulted in" property in CIDOC CRM. The same holds true for SsVM and E. Also no further axiomatization of causality is provided by these models such as that causes and effects are events and only events, cardinality between causes and effects, and support for modeling a justification of a causal relationship (see Section 5.3). Only the event calculus and the situation calculus provide well formed support for causality. However, the latter defines causality not between events but on fluents, which are functions on situations. The correlation relationship is only supported by a specific extension of the event calculus [11]. None of the existing models provides for different interpretations of the same event. Only E introduces a similar concept called *constellations* but it remains as future work. Considering the functional requirements, we can state that the Event-Model-F supports all of them.

With respect to the non-functional requirements, one can say that the existing event models are developed "ad hoc", i.e., they do not follow a systematic development approach and mostly do not define a formal semantics. Consequently, they are semantically ambiguous (cf. [59]). For example, the model used in the Eventory [67] system is only verbally described. Ambiguities in E [70] with respect to naming conventions and ontological relationships of key concepts have been removed [51]. However, the principal problem of being an informally defined event model that is developed in an ad-hoc manner remains. For the SAWA model, the axiomatization is conducted only on its domain-specific extensions using rules. The core model itself is not formalized and has some modeling flaws such as introducing a concept for physical objects as specialization of the object concept without introducing a concept for non-physical objects [5]. In the Situation Ontology the concept of situation is ambiguously defined and can be either an object or a property [5]. A formal basis can be provided by using a foundational ontology such as DOLCE [21, 28] or SUMO [41]. Foundational ontologies come with formal definitions of fundamental concepts of the world, i.e., any concepts that are of metaphysical interest such as events and objects, and provide axioms that can be used and extended. However, they do not provide a fully fledged event model as our goal is in this work. Thus, they are a solid modeling basis for core ontologies such as our Event-Model-F and are good design practise. None of the existing event models leverage a foundational ontology like DOLCE for event modeling. Thus, they do not benefit from the formal semantics already defined in such ontologies. They also do not follow a pattern-oriented approach that would allow them to structure the complex problem of an event model into smaller, reusable units. This is very unfortunate as, e.g., DOLCE has already proven to provide a good design basis and modeling approach for different core ontologies such as [45, 2, 18, 38, 39]. Extensibility and separation of concern are typically not in the scope of the existing event models. For example, in SAWA it is difficult to associate the core model with existing domain ontologies [57]. This hinders (re-)use of the SAWA model as it cannot be assumed that for each domain a specific SAWA-aligned ontology will be created. Finally, we should mention that the Event Ontology reuses existing ontologies for modeling the participative, temporal, and spatial aspects. Other ontologies such as SOUPA provide a well elaborated representation of temporal relations and space [5]. However, existing ontologies are not reused. Reuse of existing ontologies is a feature that is also supported by our Event-Model-F.

For designing the Event-Model-F, we have aligned it carefully with the foundational ontology DUL. This choice of DUL as modeling basis specifically reflects all the non-functional requirements as described in Section 4. Foundational ontologies provide a high-level, abstract vocabulary of concepts and relations that are likely to be used in current and future application domains. Thus, a precise alignment of concepts defined in the Event-Model-F with the high-level concepts of a foundational ontology provides a solid basis for future extensions. This alignment also includes the adoption and specialization of the formal semantics of the foundational ontology in our Event-Model-F. Thus, it supports for validating the more specific semantics of the concepts and relations defined in the Event-Model-F. For designing the Event-Model-F, we built upon the foundational ontology DOLCE that supports a pattern-oriented design approach [38]. We have designed the Event-Model-F as a set of patterns that structure the domain into smaller and better manageable ontology modules. The individual events and objects are reified by the different patterns of our ontology. Splitting up the description of events and objects into different parts allows for reusing them among different applications where only parts of the descriptions might be needed and combined with domain-specific knowledge. Finally, the separation of concerns is supported by defining the structural knowledge of events and objects in the Event-Model-F and leaving all domain-specific aspects out of it. By this, the Event-Model-F is independent of any concrete domain that makes use of events and objects.

9.2 Command-and-Control Systems for Emergency Response

The SemaPlorer++ application relates to so-called command-and-control systems for managing emergency incidents and coordinating the emergency response. We find different existing solutions and applications for emergency response such as the commercial Atlas Incident Management System (AIMS) [62] by Ultra Electronics. AIMS provides among others a task management system to allocate, coordinate, and record tasks and logs all incoming and outgoing messages. The Command Support System [66] by VectorCommand is a collaborative platform that aims at creating and sharing a common understanding of an emergency situation through features like messaging, electronic whiteboard, and automatic incident log recording.

Ushahidi [65] is an open-source emergency response system that has initially been developed for reporting violence incidents in Kenya after the post-election fallout in 2008. Since then, it has been used to manage the emergency response in several incidents such as tracking the post-earthquake effort in Haiti and Chile. Goal of the Ushahidi system is to develop an early warning system and visualization of data for emergency response. Another open source emergency response management system is the web-based collaboration platform Sahana [47] for coordinating different emergency response tasks. It provides features such as finding missing people, managing aid and volunteers, and communicating between the professional emergency response organizations and victims.

The 911 program [64] is a national effort of the United States that aims at establishing the Internet as a commonly accepted infrastructure for emergency response [60]. To this end, the traditional emergency hotline system based on wired phones is extended by new channels such as calls from cell phones, text messaging, and the Internet. The focus of the 911 program lies on the network aspects of an Internet-enabled emergency response system such as routing and prioritization protocols [60]. However, it is also concerned with determing the responsible entities in a concrete incident, acquiring and determing the location of the caller, and establishing security controls for system access. Within this context, the 911 Community Response Grid [63, 71] is developed by the University of Maryland. It allows citizens to online receive and submit information about emergency incidents and security problems. The information in the Community Response Grid is not only shared with community officials but also with other citizens.

The existing systems and solutions for managing emergency incidents provide different features such as information sharing and task management to improve the situational awareness in emergency response. Ultra Electronics as well as VectorCommand require installation of their software on all computers of all emergency response entities. Thus, all emergency response entities have to agree on a common software solution. This may be enforceable to some extend. However, it can be assumed that there will be emergency response organizations or civil organizations that use different systems and thus are not integrated. For example, the emergency response entities in rural areas may not have the support to upgrade their systems to the latest technology [64]. Like the commercial systems, Ushahidi and Sahana require that all emergency response entities as well as citizens are using the same platform. In addition, they are vulnerable with respect to the accessibility and performance of the single server that runs the platform. An exchange of information between the applications and platforms of the different commercial and open-source solutions is not possible.

The 911 program defines the Internet as a common infrastructure to enable interoperability of the different emergency response systems and emergency response services. However, it focuses on the network aspects and does yet not tackle the interoperability of domain data. The Emergency Data Exchange Language (EDXL) is an effort of the OASIS[15] emergency management technical committee to gain interoperability and transparency of domain data in emergency response. It defines the EDXL Distribution Element (EDXL-DE) to share event-like information between emergency response entities [69]. However, the EDXL-DE remains very simplistic and only supports information like the time, location, and type of incident. In addition, the specification is semantically ambiguous as it does not provide a formal semantics like the Event-Model-F. For example, it is unclear if the time and date provided with the distribution element refer to the actual time of when the incident happened or when the distributin element has been shared. However, this distinction is very important in emergency response. In addition, the recipients of a distribution element can be specifed, e.g., by keywords taken from a list. For specifying the address of the recipient an email can be

[15] Organization for the Advancement of Structured Information Standards, `http://www.oasis-open.org/` [Last retrieved: August 06, 2010]

provided, but it remains unclear if other channels like instant messaging and SMS can also be used.

To enable the exchange of complex, structured event-based information between the heterogeneous computer systems and applications used by the emergency response entities, we have developed the Event-Model-F. The formal nature of the Event-Model-F provides an unambiguous semantics to the structured event-based information. By this, it facilitates interoperability between different event-based applications such as emergency incident management systems. We allow for integrating heterogeneous client applications and systems in a systematic and formally precise way. To prove the applicability of the Event-Model-F, we have implemented a Java-based API (see Section 7). We have specialized the API towards domain-specific requirements for emergency response and have integrated it with the SemaPlorer++ application by using it in the event-manager plugin (see Sections 3 and 8). As such, the SemaPlorer++ application and the event-manager plugin serve as role model how to integrate the Event-Model-F and leverage it for representing complex event-based knowledge in the domain of emergency response. The API of the Event-Model-F and the event-manager plugin of the SemaPlorer++ application can serve as basis for future extensions of emergency response management applications such as Ushahidi and Sahana to enable interoperability between the different applications.

10 Conclusions

A sophisticated model of events is needed to capture and represent occurrences in the real world for semantic ambient media applications. To fill this need, we have designed the Event-Model-F based on the foundational ontology DOLCE+DnS Ultralite (DUL) and have axiomatized it in Description Logics using OWL. For the functional requirements, we have introduced specific ontology design patterns based on the Descriptions and Situations pattern of DUL. By this, we can represent arbitrary occurrences in the real world and formally model the different relations and interpretations of events. The full support for the structural aspect as well as different event interpretations distinguishes the Event-Model-F very much from existing event models. Separating the model into smaller patterns allows for better managing the complexity of events. By the use of this ontology and following a pattern-oriented design approach, we are able to fulfill the non-functional requirements stated in Section 4. Due to its formal nature, the Event-Model-F allows for integrating different event-based components and event-based systems. We have developed an API for using the Event-Model-F in Java-based applications and have demonstrated the use of the API at the example of the SemaPlorer++ application in the domain of emergency response. The Event-Model-F and its API are available online at: `http://isweb.uni-koblenz.de/eventmodel`.

In a future work, we plan to develop pattern-based core ontologies for further aspects of situation awareness. These core ontologies include the aspects of user context, rules, devices, and others.

Acknowledgements This research has been co-funded by the EU in FP6 in the X-Media project (026978) and FP7 in the WeKnowIt project (215453). We kindly thank Peter Whitwam and Keith Bradley from the Emergency Planning Team of the City Council of Sheffield, UK for the discussions on emergency planning and emergency response and the requirements and feedback on the SemaPlorer++ application. We thank our student Daniel Schmeiß for his support in implementing the Event-Model-F API and SemaPlorer++ application.

References

1. D. Anicic, C. Brelage, O. Etzion, and N. Stojanovic. Complex event processing for the future internet, September 2008. `http://icep-fis08.fzi.de/`.
2. R. Arndt, R. Troncy, S. Staab, L. Hardman, and M. Vacura. COMM: Designing a well-founded multimedia ontology for the web. In *International Semantic Web Conference*. Springer, 2007.
3. S. Auer, J. Lehmann, and S. Hellmann. LinkedGeoData: Adding a spatial dimension to the web of data. In *International Semantic Web Conference*, pages 731–746, Berlin, Heidelberg, 2009. Springer. ISBN 978-3-642-04929-3. doi: http://dx.doi.org/10.1007/978-3-642-04930-9_46.
4. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. ISBN 0-521-78176-0.
5. N. Baumgartner and W. Retschitzegger. A survey of upper ontologies for situation awareness. In *Knowledge Sharing and Collaborative Engineering; St. Thomas, VI, USA*, pages 1–9. ACTA Press, 2006.
6. N. Baumgartner, W. Retschitzegger, and W. Schwinger. A software architecture for ontology-driven situation awareness. In *Applied computing*, pages 2326–2330, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-753-7. doi: http://doi.acm.org/10.1145/1363686.1364237.
7. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia: A crystallization point for the web of data. *Web Semant.*, 7(3): 154–165, 2009. ISSN 1570-8268. doi: http://dx.doi.org/10.1016/j.websem.2009.07.002.
8. S. Borgo and C. Masolo. *Handbook on Ontologies*, chapter Foundational choices in DOLCE. Springer, 2nd edition, 2009.
9. K. Broda, K. Clark, R. M. 0002, and A. Russo. Sage: A logical agent-based environment monitoring and control system. In Tscheligi et al. [61], pages 112–117. ISBN 978-3-642-05407-5.
10. R. Casati and A. Varzi. Events. Stanford Encyclopedia of Philosophy, 2006. `http://plato.stanford.edu/entries/events`.
11. I. Cervesato, M. Franceschet, and A. Montanari. A guided tour through some extensions of the event calculus. *Computational Intelligence*, 16:200–0, 1999.
12. H. Chen and A. Joshi. *The SOUPA Ontology for Pervasive Computing.* Birkhauser Publishing Ltd., April 2004.
13. H. Chen, T. W. Finin, and A. Joshi. Using OWL in a pervasive computing broker. In *Ontologies in Agent Systems; Melbourne, Australia*, volume 73 of *CEUR Workshop Proceedings*, pages 9–16. CEUR-WS.org, 2003.
14. M. Doerr, C.-E. Ore, and S. Stead. The CIDOC conceptual reference model: a new standard for knowledge sharing. In *Conceptual modeling*, pages 51–56. Australian Computer Society, Inc., 2007. ISBN 978-1-920682-64-4.
15. A. Ekin, A. M. Tekalp, and R. Mehrotra. Integrated semantic-syntactic video modeling for search and browsing. *IEEE Transactions on Multimedia*, 6(6):839–851, 2004.
16. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, Massachusetts, 1998.
17. A. R. J. Francois, R. Nevatia, J. Hobbs, and R. C. Bolles. VERL: An ontology framework for representing and annotating video events. *IEEE MultiMedia*, 12(4),

2005.

18. T. Franz, S. Staab, and R. Arndt. The X-COSIM integration framework for a seamless semantic desktop. In *Knowledge capture*, pages 143–150, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-643-1. doi: http://doi.acm.org/10.1145/1298406.1298433.
19. A. Gangemi. Norms and plans as unification criteria for social collectives. *Autonomous Agents and Multi-Agent Systems*, 17(1):70–112, 2008. ISSN 1387-2532. doi: http://dx.doi.org/10.1007/s10458-008-9038-9.
20. A. Gangemi and P. Mika. Understanding the semantic web through descriptions and situations. In *CoopIS/DOA/ODBASE*, pages 689–706, 2003.
21. A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening Ontologies with DOLCE. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 166–181, London, UK, 2002. Springer. ISBN 3-540-44268-5.
22. IPTC International Press Telecommunications Council, London, UK. EventML, 2008. http://iptc.org/.
23. E. Itkonen. *Causality in Linguistic Theory*. Indiana Univ. Press, Bloomington, Indiana, USA, 1983.
24. M. M. Kokar, C. J. Matheus, and K. Baclawski. Ontology-based situation awareness. *Inf. Fusion*, 10(1):83–98, 2009. ISSN 1566-2535. doi: http://dx.doi.org/10.1016/j.inffus.2007.01.004.
25. F. Lin. Embracing causality in specifying the indeterminate effects of actions. In *AAAI/IAAI, Vol. 1*, pages 670–676, 1996.
26. F. Lin. *Handbook of Knowledge Representation*, chapter Situtation Calculus. Elsevier, 2008.
27. L. Lombard. *Events: A metaphysical study*. Routledge & Kegan Paul, 1986.
28. C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. WonderWeb deliverable D18 ontology library; IST WonderWeb project, December 2003. `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf`.
29. C. Matheus, M. Kokar, K. Baclawski, J. Letkowski, C. Call, M. Hinman, J. Salerno, and D. Boulware. Sawa: An assistant for higher-level fusion and situation awareness. In *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications; Orlando, USA*, pages 75–85. SPIE, March 2005.
30. C. J. Matheus, M. M. Kokar, and K. Baclawski. A core ontology for situation awareness; Cairns, Australia. In *Information Fusion*, pages 545–552, July 2003.
31. C. J. Matheus, K. Baclawski, M. M. Kokar, and J. Letkowski. Using SWRL and OWL to capture domain knowledge for a situation awareness application applied to a supply logistics scenario. In *Rules and Rule Markup Languages for the Semantic Web*, volume 3791 of *LNCS*, pages 130–144. Springer, 2005.
32. C. J. Matheus, M. M. Kokar, K. Baclawski, and J. Letkowski. An application of semantic web technologies to situation awareness. In *International Semantic Web Conference*, volume 3729 of *LNCS*, pages 944–958. Springer, 2005.
33. E. T. Mueller. *Handbook of Knowledge Representation*, chapter Event Calculus. Elsevier, 2008.
34. G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer, 2006.
35. P. Mylonas, Y. Avrithis, Y. Kalantidis, E. Spyrou, G. Tolias, E. Giannakidou, N. Ireson, and P. Smrz. D2.1.2 intelligent media analysis tools. Technical report, September 2009. `http://www.weknowit.eu/sites/default/files/D2.1.2.pdf`.

36. R. Nevatia, J. Hobbs, and B. Bolles. An ontology for video event representation. In *Computer Vision and Pattern Recognition*, page 119, Washington, DC, USA, 2004. IEEE. ISBN 0-7695-2158-4.
37. D. Oberle. *Semantic Management of Middleware*. Springer, 2006.
38. D. Oberle, S. Lamparter, S. Grimm, D. Vrandečić, S. Staab, and A. Gangemi. Towards ontologies for formalizing modularization and communication in large software systems. *Applied Ontologies*, 1(2):163–202, 2006. ISSN 1570-5838.
39. D. Oberle, A. Ankolekar, P. Hitzler, P. Cimiano, M. Sintek, M. Kiesel, B. Mougouie, S. Baumann, S. Vembu, M. Romanelli, P. Buitelaar, R. Engel, D. Sonntag, N. Reithinger, B. Loos, H.-P. Zorn, V. Micelli, R. Porzel, C. Schmidt, M. Weiten, F. Burkhardt, and J. Zhou. DOLCE ergo SUMO: On foundational & domain models in the SmartWeb Integrated Ontology. *Web Semantics*, 5(3):156–174, 2007. ISSN 1570-8268. doi: http://dx.doi.org/10.1016/j.websem.2007.06.002.
40. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. Technical report, W3C, 2004. `http://www.w3.org/TR/owl-semantics/`.
41. A. Pease, I. Niles, and J. Li. The suggested upper merged ontology: A large ontology for the semantic web and its applications. In *Ontologies and the Semantic Web*. AAAI, July/August 2002.
42. A. Quinton. Objects and events. *Mind*, 88(350):197–214, April 1979.
43. Y. Raimond and S. Abdallah. The event ontology, October 2007. `http://motools.sf.net/event`.
44. S. Rozsnyai, J. Schiefer, and A. Schatten. Concepts and models for typing events for event-based systems. In *Distributed event-based systems*. ACM, June 2007.
45. C. Saathoff and A. Scherp. Unlocking the semantics of multimedia presentations in the web with the multimedia metadata ontology. In *World Wide Web Conference; Raleigh, NC, USA*, pages 831–840. ACM, 2010.
46. C. Saathoff, S. Schenk, and A. Scherp. KAT: The K-Space Annotation Tool. In *SAMT Demo and Poster Session; Koblenz, Germany*, 2008.
47. Sahana Software Foundation. Sahana: Home of the free and open source disaster management system, 2010. `http://sahanafoundation.org/`.
48. S. Schenk and S. Staab. Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web. In *World Wide Web Conference; Beijing, China*, Bejing, China, 2008.
49. S. Schenk, C. Saathoff, S. Staab, and A. Scherp. Semaplorer: Interactive semantic exploration of data and media based on a federated cloud infrastructure. *Web Semantics*, 7(4):298–304, 2009. ISSN 1570-8268. doi: http://dx.doi.org/10.1016/j.websem.2009.09.006.
50. A. Scherp. Research on events in computer science. In *Semantic Ambient Media Intelligence; Salzburg, Austria*, 2009.
51. A. Scherp, S. Agaram, and R. Jain. Event-centric media management. In *SPIE*, volume 6820, 2008.
52. A. Scherp, T. Franz, C. Saathoff, and S. Staab. F–a model of events based on the foundational ontology DOLCE+DnS Ultralight. In *Conference on Knowledge Capture*, pages 137–144, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-658-8. doi: http://doi.acm.org/10.1145/1597735.1597760.
53. A. Scherp, S. Papadopoulos, A. Kritikos, F. Schwagereit, C. Saathoff, T. Franz, D. Schmeiss, S. Staab, S. Schenk, and M. Bonifacio. D5.2.1 prototypical knowledge management methodology. Technical report, 2009. `http://www.weknowit.eu/`

sites/default/files/D5.2.1.pdf.

54. G. Shafer. Causal logic. In *European conference on artificial intelligence*. Wiley, 1998.
55. B. Shipley. *Cause and Correlation in Biology*. Cambridge Univ. Press, 2002.
56. P. Sinclair, M. Addis, F. Choi, M. Doerr, P. Lewis, and K. Martinez. The use of CRM core in multimedia annotation. In *Semantic Web Annotations for Multimedia*, May 2006.
57. P. R. Smart. Knowledge-intensive fusion for situational awareness: Band sultan dam failure scenario. Technical report, School of Electronics and Computer Science, University of Southampton, 2005. URL http://eprints.ecs.soton.ac.uk/11609/.
58. H. Storf, T. Kleinberger, M. Becker, M. Schmitt, F. Bomarius, and S. Prueckner. An event-driven approach to activity recognition in ambient assisted living. In Tscheligi et al. [61], pages 123–132. ISBN 978-3-642-05407-5.
59. T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Advanced Context Modelling, Reasoning and Management; Nottingham, England*, 2004.
60. The National E9-1-1 Implementation Coordination Office, U.S. Department of Transportation, Washington, DC, USA. A national plan for migrating to ip-enabled 9-1-1 systems, Sept. 2009. http://911.gov/pdf/National_NG911_Migration_Plan_FINAL.pdf.
61. M. Tscheligi, B. de Ruyter, P. Markopoulos, R. Wichert, T. Mirlacher, A. Meschtscherjakov, and W. Reitberger, editors. *Ambient Intelligence; Salzburg, Austria*, volume 5859 of *LNCS*, 2009. Springer. ISBN 978-3-642-05407-5.
62. Ultra Electronics Ltd., UK. Aims: Atlas incident management system, 2010. http://www.atlasops.com/products/aims.php.
63. University of Maryland, College Park, MD, USA. 911.gov: Community response grids, e-government, and emergencies, 2009. http://www.cs.umd.edu/hcil/911gov/.
64. U.S. Department of Transportation. 911.gov - the national 911 office, 2010. http://911.gov/.
65. Ushahidi.com. Ushahidi: Crowdsorcing crisis information, 2010. http://www.ushahidi.com/.
66. VectorCommand Ltd., UK. Emergency command system, 2010. http://www.emergencycommandsystem.com/products/command-support-system/.
67. X. Wang, S. Mamadgi, A. Thekdi, A. Kelliher, and H. Sundaram. Eventory – an event based media repository. In *Semantic Computing*, pages 95–104, Washington, DC, USA, 2007. IEEE. ISBN 0-7695-2997-6.
68. X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology based context modeling and reasoning using OWL. In *Pervasive Computing and Communications Workshops*, page 18, Washington, DC, USA, 2004. IEEE. ISBN 0-7695-2106-1.
69. J. Waters and R. Brooks. The distribution element: The basic steps to package and address your emergency informations [an oasis emergency management technical committee white paper], Aug. 2009. OASIS Emergency Management Technical Committee, http://www.oasis-open.org/committees/download.php/34264/EDXL-DE-Basics-White%20Paper-18Aug09-r2.doc.
70. U. Westermann and R. Jain. Toward a common event model for multimedia applications. *IEEE MultiMedia*, 14(1):19–29, 2007.
71. P. F. Wu, J. Preece, B. Shneiderman, P. T. Jaeger, and Y. Qu. Community response grids for older adults: Motivations, usability, and sociability, 2007.

72. S. S. Yau and J. Liu. Hierarchical situation modeling and reasoning for pervasive computing. In *Software Technologies for Future Embedded and Ubiquitous Systems*, pages 5–10, Washington, DC, USA, 2006. IEEE. ISBN 0-7695-2560-1.
73. S. Yoshioka, Y. Hirano, S. Kajita, K. Mase, and T. Maekawa. Semi-automatic story creation system in ubiquitous sensor environment. In Tscheligi et al. [61], pages 106–111. ISBN 978-3-642-05407-5.

Appendix

A Axiomatization of the Event-Model-F in Description Logic

In the following, we describe the axiomatization of our Event-Model-F in Description Logics [4] along the different patterns for participation, composition, causality, correlation, and interpretation. A discussion on the axiomatization of ontology design patterns in general is conducted at the example of the causality pattern in Section 6.

A.1 Participation Pattern

The participation pattern is discussed in Section 5.1. It describes the set of objects that participate in an event and are relevant in a given context. The participation pattern defines that for one event there has to be at least one participant. This means that there are no events without a participating object. We formalize this with the following set of axioms:

$$\begin{aligned}
EventParticipationDescription &\sqsubseteq Description\\
EventParticipationDescription &\sqsubseteq \forall defines.(Participant \sqcup DescribedEvent)\\
EventParticipationDescription &\sqsubseteq \geq 1(defines.Participant)\\
EventParticipationDescription &\sqsubseteq = 1(defines.DescribedEvent)\\
EventParticipationDescription &\sqsubseteq = 1(satisfiedBy.EventParticipationSituation)\\
EventParticipationSituation &\sqsubseteq Situation\\
EventParticipationSituation &\sqsubseteq \forall includesEvent.(\\
&\qquad \exists isClassifiedBy.DescribedEvent)\\
EventParticipationSituation &\sqsubseteq \forall includesObject.(\\
&\qquad \exists isClassifiedBy.Participant)\\
EventParticipationSituation &\sqsubseteq = 1(satisfies.EventParticipationDescription)\\
DescribedEvent &\sqsubseteq EventType\\
DescribedEvent &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&\qquad EventParticipationSituation)\\
DescribedEvent &\sqsubseteq = 1(isDefinedIn.EventParticipationDescription)\\
Participant &\sqsubseteq Role\\
Participant &\sqsubseteq \forall classifies.(\exists isObjectIncludedIn.\\
&\qquad EventParticipationSituation)\\
Participant &\sqsubseteq = 1(isDefinedIn.EventParticipationDescription)
\end{aligned}$$

A.2 Mereology Pattern

The composition pattern defines how events are composed, i.e., it basically describes a part-whole relationship between events that is valid in a certain context and is possibly subject to a set of constraints (cf. Section 5.2). We require exactly one composite event, i.e., the whole, and at least one component, i.e., the part. The specification of constraints is optional.

$$
\begin{aligned}
EventCompositionDescription &\sqsubseteq Description\\
EventCompositionDescription &\sqsubseteq \forall defines.(Composite \sqcup Component\\
&\quad \sqcup EventCompositionConstraint)\\
EventCompositionDescription &\sqsubseteq = 1(defines.Composite)\\
EventCompositionDescription &\sqsubseteq \geq 1(defines.Component)\\
EventCompositionDescription &\sqsubseteq = 1(satisfiedBy.EventCompositionSituation)\\
EventCompositionSituation &\sqsubseteq Situation\\
EventCompositionSituation &\sqsubseteq \forall includesEvent.(\exists isClassifiedBy.\\
&(Composite \sqcup Component))\\
EventCompositionSituation &\sqsubseteq \forall includesSpace.(\exists isParametrizedBy.SpatialConstraint)\\
EventCompositionSituation &\sqsubseteq \forall includesTime.(\exists isParametrizedBy.TemporalConstraint)\\
EventCompositionSituation &\sqsubseteq \forall includesSpaceTime.\\
&(\exists isParametrizedBy.SpatioTemporalConstraint)\\
EventCompositionSituation &\sqsubseteq = 1(satisfies.EventCompositionDescription)\\
Composite &\sqsubseteq EventType\\
Composite &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&EventCompositionSituation)\\
Composite &\sqsubseteq = 1(isDefinedIn.EventCompositionDescription)\\
Component &\sqsubseteq EventType\\
Component &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&EventCompositionSituation)\\
Component &\sqsubseteq = 1(isDefinedIn.EventCompositionDescription)\\
EventCompositionConstraint &\sqsubseteq Parameter\\
EventCompositionConstraint &\sqsubseteq = 1(isDefinedIn.EventComposition)\\
EventCompositionConstraint &\sqsubseteq \forall parametrizes.(\exists hasSetting.\\
&EventCompositionSituation)\\
SpatialConstraint &\sqsubseteq EventCompositionConstraint\\
SpatialConstraint &\sqsubseteq \forall parametrizes.SpaceRegion\\
TemporalConstraint &\sqsubseteq EventCompositionConstraint\\
TemporalConstraint &\sqsubseteq \forall parametrizes.TimeRegion\\
SpatioTemporalConstraint &\sqsubseteq EventCompositionConstraint\\
SpatioTemporalConstraint &\sqsubseteq \forall parametrizes.SpatioTemporalRegion
\end{aligned}
$$

A.3 Causality Pattern

The causality pattern defines a causal relationship by exactly one cause, exactly one effect, and exactly one justification. The pattern is described in Section 5.3. A formal axiomatization is provided below.

$$\begin{aligned}
EventCausalityDescription &\sqsubseteq Description\\
EventCausalityDescription &\sqsubseteq \forall defines.(Cause \sqcup Effect \sqcup Justification)\\
EventCausalityDescription &\sqsubseteq = 1(defines.Cause)\\
EventCausalityDescription &\sqsubseteq = 1(defines.Effect)\\
EventCausalityDescription &\sqsubseteq = 1(defines.Justification)\\
EventCausalityDescription &\sqsubseteq = 1(satisfiedBy.EventCausalitySituation)\\
EventCausalitySituation &\sqsubseteq Situation\\
EventCausalitySituation &\sqsubseteq = 1(includesEvent.(\exists isClassifiedBy.Cause))\\
EventCausalitySituation &\sqsubseteq = 1(includesEvent.(\exists isClassifiedBy.Effect))\\
EventCausalitySituation &\sqsubseteq = 1(includesObject.(\exists isClassifiedBy.Justification))\\
EventCausalitySituation &\sqsubseteq = 1(satisfies.EventCausalityDescription)\\
Cause &\sqsubseteq EventType\\
Cause &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&\qquad EventCausalitySituation)\\
Cause &\sqsubseteq = 1(isDefinedIn.EventCausalityDescription)\\
Effect &\sqsubseteq EventType\\
Effect &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&\qquad EventCausalitySituation)\\
Effect &\sqsubseteq = 1(isDefinedIn.EventCausalityDescription)\\
Justification &\sqsubseteq Role\\
Justification &\sqsubseteq \forall classifies.(Description \sqcap \exists isObjectIncludedIn.\\
&\quad (EventCausalitySituation \sqcup\\
&\quad EventCorrelationSituation)\\
Justification &\sqsubseteq = 1(isDefinedIn.(EventCausalityDescription \sqcup\\
&EventCorrelationDescription))
\end{aligned}$$

A.4 Correlation Pattern

The correlation pattern describes the correlation of a set of events, as discussed in Section 5.4. It only makes sense to specify a correlation between two or more events. Further, the correlation description also refers to the justification defined for the causality pattern.

$$\begin{aligned}
EventCorrelationDescription &\sqsubseteq Description\\
EventCorrelationDescription &\sqsubseteq \forall defines.(Correlate \sqcup Justification)\\
EventCorrelationDescription &\sqsubseteq \geq 2(defines.Correlate)\\
EventCorrelationDescription &\sqsubseteq = 1(defines.Justification)\\
EventCorrelationDescription &\sqsubseteq = 1(satisfiedBy.EventCorrelationSituation)\\
EventCorrelationSituation &\sqsubseteq Situation\\
EventCorrelationSituation &\sqsubseteq \geq 2(includesEvent.(\exists isClassifiedBy.Correlate))\\
EventCorrelationSituation &\sqsubseteq = 1(includesObject.(\exists isClassifiedBy.Justification))\\
EventCorrelationSituation &\sqsubseteq = 1(satisfies.EventCausalityDescription)\\
Correlate &\sqsubseteq EventType\\
Correlate &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&\qquad EventCorrelationSituation)\\
Correlate &\sqsubseteq = 1(isDefinedIn.EventCorrelationDescription)
\end{aligned}$$

A.5 Documentation Pattern

The documentation pattern provides the documentation of an event by arbitrary sensory data such as images, video, and audio as well as other events. Thus, it allows to specify for an event by which objects and events it is documented. The pattern has been discussed in Section 5.5. A formal axiomatization is provided below.

$$\begin{aligned}
EventDocumentationDescription &\sqsubseteq Description\\
EventDocumentationDescription &\sqsubseteq \forall defines.(DocumentedEvent \sqcup Documenter)\\
EventDocumentationDescription &\sqsubseteq = 1(defines.DocumentedEvent)\\
EventDocumentationDescription &\sqsubseteq \geq 1(defines.Documenter)\\
EventDocumentationDescription &\sqsubseteq = 1(satisfiedBy.EventDocumentationSituation)\\
EventDocumentationSituation &\sqsubseteq Situation\\
EventDocumentationSituation &\sqsubseteq = 1(includesEvent.(\exists isClassifiedBy.DocumentedEvent))\\
EventDocumentationSituation &\sqsubseteq \geq 1(hasSetting.(\exists isClassifiedBy.Documenter))\\
EventDocumentationSituation &\sqsubseteq = 1(satisfies.EventDocumentationDescription)\\
DocumentedEvent &\sqsubseteq EventType\\
DocumentedEvent &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&\qquad EventDocumentationSituation)\\
DocumentedEvent &\sqsubseteq = 1(isDefinedIn.EventDocumentationDescription)\\
Documenter &\sqsubseteq Concept\\
Documenter &\sqsubseteq \forall classifies.(\exists hasSetting.\\
&\qquad EventDocumentationSituation)\\
Documenter &\sqsubseteq = 1(isDefinedIn.EventDocumentationDescription)
\end{aligned}$$

A.6 Interpretation Pattern

The interpretation pattern defines an interpretation of exactly one event. Therefore, it provides the means to specify all those patterns for an event that are relevant for the interpretation. We have discussed the pattern in Section 5.6 and provide the formal axiomatization below.

$$\begin{aligned}
EventInterpretationDescription &\sqsubseteq Description\\
EventInterpretationDescription &\sqsubseteq \forall defines.(Interpretant \sqcup RelevantSituation)\\
EventInterpretationDescription &\sqsubseteq = 1(defines.Interpretant)\\
EventInterpretationDescription &\sqsubseteq \geq 1(defines.RelevantSituation)\\
EventInterpretationDescription &\sqsubseteq = 1(satisfiedBy.EventInterpretationSituation)\\
EventInterpretationSituation &\sqsubseteq Situation\\
EventInterpretationSituation &\sqsubseteq = 1(includesEvent.(\exists isClassifiedBy.Interpretant))\\
EventInterpretationSituation &\sqsubseteq \geq 1(includesObject.(Situation \sqcap \exists\\
&\quad isClassifiedBy.RelevantSituation))\\
EventInterpretationSituation &\sqsubseteq = 1(satisfies.EventInterpretationDescription)\\
Interpretant &\sqsubseteq EventType\\
Interpretant &\sqsubseteq \forall classifies.(\exists isEventIncludedIn.\\
&\quad EventInterpretationSituation)\\
Interpretant &\sqsubseteq = 1(isDefinedIn.EventInterpretationDescription)\\
RelevantSituation &\sqsubseteq Role\\
RelevantSituation &\sqsubseteq \forall classifies.(Situation \sqcap\\
&\exists isObjectIncludedIn.EventInterpretationSituation)\\
RelevantSituation &\sqsubseteq = 1(isDefinedIn.EventInterpretationDescription)
\end{aligned}$$