Design and use of the Simple Event Model (SEM)

Willem Robert van Hage, Véronique Malaisé, Roxane Segers,
Laura Hollink, and Guus Schreiber

Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands

Abstract. Events have become central elements in the representation of data from domains such as history, cultural heritage, multimedia and geography. The Simple Event Model (SEM) is created to model events in these various domains, without making assumptions about the domain-specific vocabularies used. SEM is designed with a minimum of semantic commitment to guarantee maximal interoperability. In this paper, we discuss the general requirements of an event model for web data and give examples from two use cases: historic events and events in the maritime safety and security domain. The advantages and disadvantages of several existing event models are discussed in the context of the historic example. We discuss the design decisions underlying SEM. SEM is coupled with a Prolog API that enables users to create instances of events without going into the details of the implementation of the model. By a tight coupling to existing Prolog packages, the API facilitates easy integration of event instances to Linked Open Data. We illustrate use of the API with examples from the maritime domain.

1 Introduction

Events have become central elements in the representation of data from domains such as history[1], cultural heritage [2,5], multimedia [10] and geography [12]. Event-centered modeling captures the dynamic aspects of a domain. In addition, events provide a natural way to explicate complicated relations between people, places, actions and objects. In this paper we investigate the representation of events on the Web. We propose an event model called the Simple Event Model (SEM), which is designed with a minimum of semantic commitment to guarantee maximal interoperability with data sets from various domains.

The diversity of Web data poses a challenge on the design of an event model. First, each domain has a different data structure and poses therefore different requirements on a model. Defining separate domain-specific models would be a solution. However, to connect events across domains we aim to create a non-domain specific model that makes minimal assumptions on the structure of the data, and that can be extended for domain specific requirements. That being said, the existence of one unifying event model is myth: the general model we propose is not the first of its kind nor will it be the last. To benefit from them, we create mappings from our model to existing event models. The second challenge is the openness of the Web. The fact that everyone can contribute means that we cannot assume that future data will follow the same rules as the data

[1] The AGORA project `http://www.cs.vu.nl/~schreiber/projects/agora.html`

we currently have. New vocabularies keep popping up. A model that is tied to specific vocabularies for its classes (*e.g.* by means of domain and range restrictions), such as GeoNames for places or DBpedia for events, is unusable to model data sets that use other vocabularies. We minimally commit our model by assuming nothing about the used vocabularies.

We have selected examples from two use cases from which we draw further requirements for an event model: historic events and events in the maritime safety and security domain. These domains have commonalities: In both these domains, it becomes apparent that the model needs to represent not only the description of who did what, when and where, but also the roles that each actor played, the time that a role was valid and the authority according to whom this role is assigned. However, they also differ a lot: In the history domain missing knowledge is common (there are only accounts of varying soundness and completeness) and differing opinions. In the maritime domain data are often in a raw format (*e.g.* radar tracks) and the important features are often not predefined (*e.g.* time and place are often indicated with coordinate values instead of names). We take a historical example to describe SEM and a selection of other event models. We take a maritime example to illustrate the use of the SEM API and to show the applicability of SEM in a very different domain.

The maritime examples will be mentioned in section 4. The event description from our historic dataset is the following[2]:

> The Dutch launched the first *police action* in the Dutch East Indies in 1947; the Dutch presented themselves as liberators but were seen as occupiers by the Indonesian people.

This example is interesting for a number of reasons: (1) it contains conflicting views on the role of the actor: were the Dutch liberators or occupiers? (2) it makes explicit according to which authority the roles hold (the Dutch / the Indonesian people); (3) it presents a challenge for modeling the type of the place involved: the Dutch East Indies were at that time an independent Republic, according to the Indonesians, but were a 'controlled region', according to the Dutch. We will use this as a running example throughout the paper to illustrate the development of SEM.

Several event models or ontologies have been published over the past years [2] [3] [4] [5] [6] [7] [10]. They differ in their focus (class or property centered), domain specificness, size and level of formalization. We review and compare the design choices of existing models and SEM using the running historical example above.

Learning to properly use an event model is hard and populating it with instances is a time consuming and error prone task. Therefore, we designed a Prolog API for SEM. It facilitates the creation of SEM instances and is integrated with the SWI-Prolog space [8] and semweb packages [11]. The combination of these three provides a fast and efficient indexing for geopositions, RDF and literals (strings, numbers, dates, etc.), along with RDFS, OWL, and rule reasoning.

[2] The original text is in Dutch. This sentence is extracted from the Netherlands Institute for Sound and Vision's catalogue description of the TV episode of Andere Tijden broadcasted on the 26/10/2004. The serie Andere Tijden consists of documentaries on historical topics.

The remainder of this paper is organised as follows. We present the general requirements that underly SEM's design and how events are modeled in SEM in section 2. We propose a short overview of event models in section 3 and review three models as representative examples. The API and its coupling with the other Prolog packages is demonstrated in section 4, on examples from the maritime domain. We conclude with a discussion and future work in section 5.

2 Simple Event Model

In this section we motivate the modeling decisions we took in the development of SEM and describe its structure.

2.1 Modeling Decisions

The primary consideration for designing SEM is that it should on the one hand be forgiving for the inherent messiness of the Semantic Web, while on the other hand still allowing you to derive useful facts. On the Web, vocabulary owners can choose different options to classify the same domain, because different situations merit different distinctions. It can be hard to decide in advance which way will prove to be the most useful, especially because the Web allows reuse across domains, in applications that were not predicted beforehand. The more constraining a model is, the harder it is to reuse. To profit the most from what the semantic web has to offer it pays off to model with relatively weak semantics. To compensate for the lack of formal inference you can make with a weak model, you have to rely more on graph patterns to do reasoning.

The greatest implication of our decision to tailor an event model for data on the web is that we **can not** commit to a specific definition of an event. Events, according to SEM, encompass everything that happens, even fictional events. Whether there is a specific place or time or whether these are known is optional. It does not matter whether there are specific actors involved. Neither does it matter whether there is consensus about the characteristics of the event. For example, King Arthur's quest or the hunt for the Red October, are valid events in SEM.

An important corollary of this loose definition of event and multitude of possible sources is that handling different viewpoints is crucial. In particular three aspects of viewpoints: (1) Event bounded roles, (2) time bounded validity of facts (*e.g.* time dependent type or role), (3) attribution of the authoritative source of a statement.

In order to query events at a relevant level of abstraction for any given application we need a good typing system. We would like to be able to reuse any vocabulary on the Web to pick our types from, regardless of how the concepts in these vocabularies are modeled.

The concrete implications of the web context for the RDF model of SEM are the following.

- We allow types to be both individuals or classes. This way we can borrow type identifiers from any vocabulary. It should not matter whether the type has been modeled as an individual or a class by the foreign vocabulary. (*cf.* OWL 2 punning)

– We use as few disjointness statements as possible, even where they would seem obvious. For example, SEM does not enforce places to be disjoint with actors. This allows reuse of vocabularies that do not make the distinction between a geographical region and its governing body.
– We only use rdf:domain and range to non-restricting classes. This way we do not inherit any constraints from these classes through property semantics.
– We map to other event models with the SKOS vocabulary[3], instead of using OWL constructs, to avoid overcommitment.
– Every class and property is optional and can be duplicated, *i.e.* without cardinality restriction. Specifically we do not enforce the use of types. One of the advantages of optionality is that we are not dependent on the existence of an event to state facts about an actor. For example, Napoleon Bonaparte was emperor, regardless of whether he participated in an event. Therefore, in SEM you do not have to explicitly model a coronation event.
– We do not declare properties functional, even if that seems appropriate. This avoids conflicts when aggregating data from different sources. For example, you might gather various birth dates for a single person. Even though the person was only born once–and thus inconsistency is appropriate–we do not want this to break our system. When reasoning over the web, debugging someone else's data is not always possible.

2.2 SEM Specification

In this section we will describe how these modeling decisions are implemented in SEM. First we will discuss the classes and properties that make up SEM; how to model views, roles and temporary validity as constraints on properties; then how to model time and space with symbols (*c.q.* URIs) or values (*c.q.* coordinates). Finally we will give a simple and more elaborate example of how the historical running example can be modeled in SEM.

The RDF code of SEM is at `http://semanticweb.cs.vu.nl/2009/11/sem/`.

Classes SEM's classes are divided in three groups: Core classes, Types, and Constraints. This is illustrated in Figure 1. There are four core classes: sem:Event (what happens), sem:Actor (who or what participated), sem:Place (where), sem:Time (when). Each core class has an associated sem:Type class, which contains resources that indicate the type of a core individual. Individuals and their types are usually borrowed from other vocabularies. For example, the sem:Place "Indonesia" (tgn:1000116) from our running example and its sem:PlaceType "republic" (tgn:82171) are borrowed from the Getty Institute's Thesaurus of Geographical Names (TGN)[4].

The sem:Type classes exist to aggregate the various implementations of type systems in any vocabulary. Some vocabularies do not have properties that exactly correspond to the sem:type property, even though type can be derived from the value of

[3] `http://www.w3.org/2004/02/skos/`
[4] `http://www.getty.edu/research/conducting_research/vocabularies/tgn/`

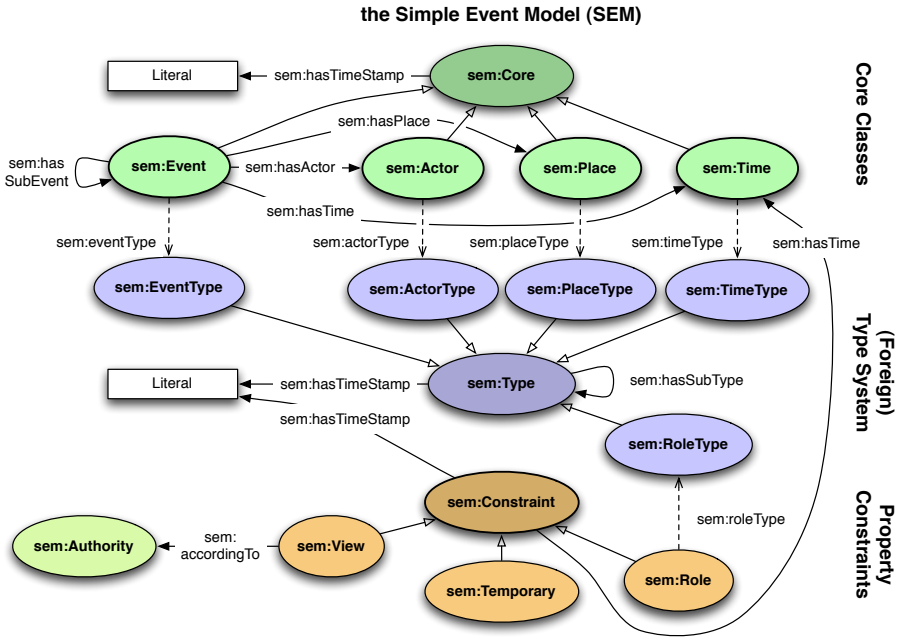

Fig. 1. The classes of the Simple Event Model. Arrows with open arrow heads symbolize rdfs:subClassOf properties. Dashed arrows symbolize subproperties of rdf:type; regular arrows represent other properties.

other properties. This can be done by using Alan Rector's Value Sets and Value Partition patterns.[5] Having explicit sem:Type classes provides a placeholder to define these patterns.

Besides the sem:Actor class, a class sem:Object has been defined as a rdfs:subClassOf sem:Actor, for the cases where it is necessary to specify a distinction between these two concepts. For example, the finding of gold during the Gold Rush might necessitate to model the class Gold as a simple object rather than as an Actor.

The class sem:Authority is used to indicate according to whom a statement is valid. Individuals of sem:Authority can be, but are not necessarily sem:Actors. They can also symbolize data sources. The sem:Authority class is meant as a hook for provenance and trust reasoning, even though SEM itself does not explicitly provides this.

Properties SEM's properties are divided in three kinds: sem:eventProperties, sem:type properties and a few miscellaneous properties like sem:accordingTo and sem:hasTimeStamp's subproperties. The sem:eventProperties relate sem:Events to other individuals. A sem:type relates individuals of the sem:Core class to individuals of sem:Type. There are specific subproperties of sem:type for each of the core classes, for example sem:eventType, to facilitate querying. This reduces the strain on reasoners, because you

[5] http://www.w3.org/TR/swbp-specified-values/

know that it points to an individual of sem:EventType without doing any subsumption reasoning. sem:accordingTo relates a sem:View to a sem:Authority and is used to represent opinions. There are seven sem:hasTimeStamp properties. One for single time values, sem:hasTimeStamp; two for time intervals, sem:hasBeginTimeStamp and sem:hasEndTimeStamp; and four for uncertain time intervals, sem:hasEarliestBeginTimeStamp, sem:hasLatestBeginTimeStamp, sem:hasEarliestEndTimeStamp, and sem:hasLatestEndTimeStamp. The latter kind of intervals is used to describe any kind of uncertainty about the begin or end of a period. It does not imply, for example, a fuzzy interpretation of time. Open-ended intervals can be expressed by omitting begin or end timestamps.

There are two aggregation relations amongst the sem:eventProperty and sem:type properties: sem:hasSubEvent and sem:hasSubType. These can be used to indicate that respectively a sem:Event or sem:Type is related to another more generic sem:Event or sem:Type, without any further commitments. More specific relations between events and types are not part of SEM and should be taken from other ontologies, like GEM [12].

Constraints Property constraints can be applied to any property. They constrain the validity of the property and are expressed as either a reification of the property or by adding attributes to the property and turning it into an n-ary relation. There are three permissible ways to represent sem:Constraints[6] that are illustrated in figure 2. The default representation is the rdf:value pattern, which is often used when representing the unit of measure of a value.[7]

There are three kinds of sem:Constraints: sem:Role, sem:Temporary and sem:View. sem:Role defines the role that an individual of a class is playing in the context of a specific event (*i.e.* to which it is linked with a sem:eventProperty). Roles can be specified for all sem:Core individuals, for example, Actors ("occupier") as well as places ("capital city", dbpedia:Colony). It is not meant to model roles in the sense of temporary or dependent types, like "mother". This can be done by putting a sem:Temporary constraint on a sem:type property. Instead, sem:Role explicitly models the event-bounded role: an "occupier", "liberator". These roles do not depend on external conditions (like the fact to have a child definies a "mother"), but only on the way they are related to the ongoing sem:Event. sem:Temporary defines the temporal boundary within which a property holds, for example, the type of the Place "Indonedia" as a "republic" holds from 1945 on. sem:View defines points of view, opinions: Indonesia, in 1947, has either the type "republic" or "controlled region", depending on who you are asking. This is modeled as a sem:View constraint on the property (sem:placeType in this case) that holds sem:accordingTo a sem:Authority (in the case of the example, respectively the Indonesian People or the Netherlands).

Multiple kinds of sem:Constraints can be used in combination to create conjunctive statements. Figure 4 shows two ways in which the combination can be done: a single RDF node that stands for multiple Constraints; and multiple chained Constraints. An example of the former is the blank node to the right of ex:FirstPoliceAction that

[6] These are all supported by the SEM API.

[7] *cf.* the MUO ontology `https://forge.morfeo-project.org/wiki_en/index.php/How_to_use_MUO`

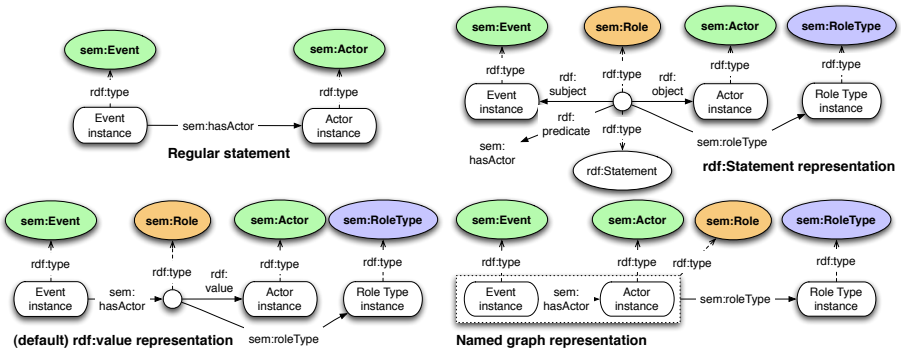

Fig. 2. Three alternative representations of property constraints in SEM.

stands for the constrained sem:hasActor property. This represents both a sem:Role and a sem:View. This expresses the fact that according to both authorities the actor is dbpedia:Netherlands, but they differ on which sem:roleType it has. This is expressed with the latter type of combination, by chaining constraints. The sem:roleType property of the sem:Role constraint is constrained further with a sem:View constraint.

The API, which will be discussed in section 4, automatically processes constraints.

Symbols versus Values to denote Place and Time The individuals of all of the Classes can be timestamped. To be compliant with other event models and Web data, which can use diverse formats, the expression of time in SEM can be symbolic (*i.e.* by referring an individual of the sem:Time class with the sem:hasTime property) or concrete: by attaching time points, two-value intervals, or four-value intervals with sem:hasTimeStamp. Symbolic representation of time can be used to represent relations between time indications, for example, that one thing happened after another, without having to say when something happened exactly. For time values we recommend using a literal of type xsd:dateTime, or a rdf:XMLLiteral containing a TIMEX time element[8], both of which support the ISO 8601[9] time format.

A similar difference between symbols and values exists when expressing places. There are symbolic places and coordinates. In SEM the individuals of the sem:Place class are symbolic places. Their location can be attached by using various constructs, like georss:point[10], or wgs84:lat and wgs84:long.[11] Complex geometries like polygons can be encoded in GML[12] in an rdf:XMLLiteral pointed at by georss:where.

Example The running example can be expressed in SEM as shown in Figure 3 and 4. The former shows a simple example, which disregards the differences of opinion

[8] http://timex2.mitre.org/
[9] http://en.wikipedia.org/wiki/ISO_8601/
[10] http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/
[11] http://www.w3.org/2003/01/geo/
[12] http://www.opengeospatial.org/standards/gml

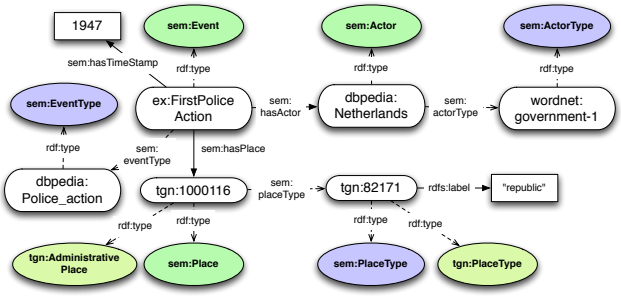

Fig. 3. A simple representation of the historical example event in SEM.

between the various parties. The latter takes into account all views and roles of the sentence.

The Scope of SEM SEM provides classes and properties to model the basic constituants of an event, their types, roles, temporary validity and the view according to which these constraints hold. However, SEM does not model relationships between events like causality, or specific properties about the semantics of they way an actor participates in an event. Other knowledge patterns like the D&S from DUL[1], or detailed models like CIDOC-CRM have modeled such distinctions and properties. In order to benefit from the specificities and advantages of these other models, we have created a set of mappings between the SEM and some other models. Because of our need for minimal commitment we use the SKOS vocabulary rather than with OWL to map to other ontologies. The mappings are included in the SEM RDF file. We describe in more details the characteristics of the main event models the SEM is mapped to in the next section, where the description of the models is centered around the historical example.

3 Related Work

Various models have been proposed for representing events on the Semantic Web, such as the Event Ontology (EO) [4][13], Linking Open Descriptions of Events (LODE) [7][14] and the F-Model (F) [6][15] and the event ontology used in CultureSampo [5]. Some more general models for semantic data organization also include event models like CIDOC-CRM[16] and the ABC ontology[3].

These event models differ significantly since they were created for diverse purposes and therefore show different design choices. Event models can be typed on basis of four main design choices: domain (in)dependency, focus on classes or properties, scope

[13] http://motools.sourceforge.net/event/event.html
[14] http://linkedevents.org/ontology/
[15] http://isweb.uni-koblenz.de/eventmodel/
[16] http://cidoc.ics.forth.gr/official_release_cidoc.html

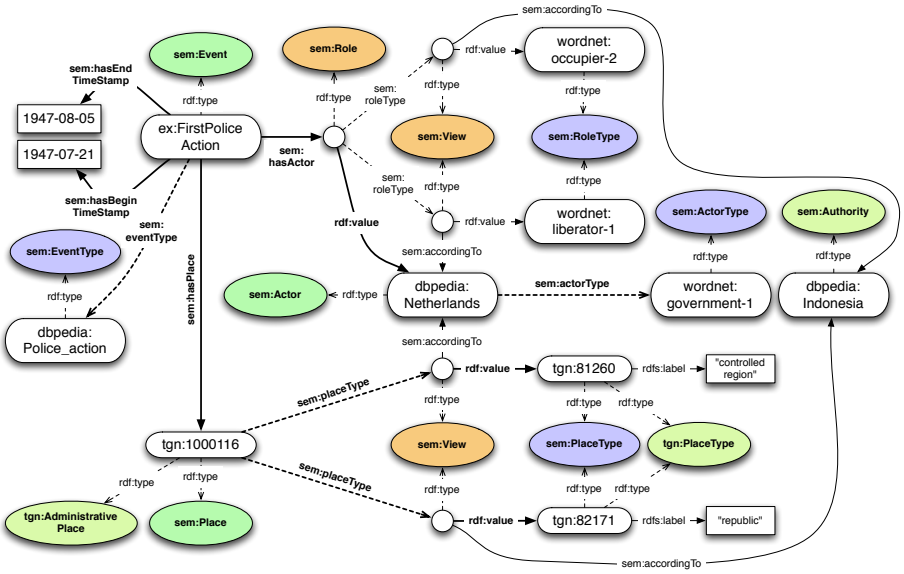

Fig. 4. A more elaborate representation, including Role and View property constraints of the historical example event in SEM.

(minimal or complex model) and the level of formalization. Models that can be classified as domain independent are EO, LODE and F, while CIDOC-CRM and ABC are domain dependent. This distinction is not strict, as domain specific models can be applied to other domains than the one they were intended for, and domain independent models are always developed in a given context. For example, EO was created in the context of music events and LODE for historic and news events. Another distinction is that some models are class-based (F, CIDOC-CRM, ABC), while others are more property-based, such as, EO, LODE and CultureSampo. The former define classes for the event constituants while the latter mostly define properties between classes in external ontologies. The models also differ in scope: some aim for a minimal representation of events (EO, LODE) while others aim at representing a broader range of event constituants and their relations like CIDOC-CRM and F. Some models are constrained by the use of OWL constraints (LODE: rdf:domain, rdf:range, rdfs:subPropertyOf, owl:sameAs equivalence to DUL and EO Event definitions, F: rdfs:subPropertyOf, rdf:domain, rdf:range, owl:inverseOf, cardinality restrictions), while others opt for a minimal commitment (EO, CIDOC-CRM). Finally, models can be highly formalized by either the partial use of classes and properties in external ontologies like F that borrows classes and properties from DUL [1] or by strict mappings of the properties to other (event) ontologies as is done in LODE. Within this group of models, SEM can be characterized as a domain independent, class based event model of average size (in number of classes and properties), that contains only few constraints.

To illustrate the benefits and possible drawbacks of the different models, we will take a more detailed look at three event models that together form a representative cross-section of the different design choices:

EO as a domain independent, property-based model with few classes and constraints.
LODE as a domain independent and property based model, still with few classes, some restrictions and a higher level of formalization.
CIDOC-CRM contains an event model that is domain-specific and class-based, with lots of classes and few constraints.

We discuss these models on basis of how they model (or not) the notions of Role, Type, View and Temporary. These notions go beyond the most common components (event, participant, time and place) and are part of our requirements. We illustrate them with the event of the *police action* (see section 1) as a running example, to show some of their differences and similarities with SEM.

3.1 Event Ontology

The Event Ontology (EO)[17] follows a very simple design and consists of four classes (eo:Event and three defined classes: Agent, Factor and Product) and seventeen properties. EO defines a minimal event, and relies on vocabularies defined externally to refine the knowledge expressed. For example, no Agent class is defined per se, but their eo:agent property has foaf:Agent as a range: EO benefits therefore from the richness of the FOAF vocabulary.[18]

Roles, Types, Views and Temporary are not defined in EO. Place, Time and Agent are defined via range restrictions on EO's properties. The explicit linking to vocabularies brings EO its richness, but also constrains the possible values for these properties. SEM is compatible with more Place, Time and Actor representations. The main common point between SEM and EO is the modularity in the design: most classes are optional in EO; In SEM, event the sem:Event classe is optional.

3.2 LODE

LODE [7] also aims at a minimal modeling of events. It contains one class (Event) and six properties: lode:atTime, lode:circa, lode:inSpace, lode:atPlace, lode:involved and lode:involvedAgent. Both the class and the properties are formally mapped to other event models like the CIDOC-CRM, EO and DUL by the use of owl:sameAs and rdfs:subPropertyOf. In this way, interoperability is enabled and a user can benefit from existing more complex vocabularies, while LODE itself keeps its own classes and properties at the lowest possible number.

Role, Type and View can be expressed via their mapping to DUL, by using the Description and Situation patterns, or via the interpretation and mereology patterns of F.[19] In SEM, we also adopt the principle of using external vocabularies for modeling

[17] http://motools.sourceforge.net/event/event.html
[18] http://www.foaf-project.org/
[19] F specializes D&S patterns from DUL.

properties that are beyond the model's scope, like the causality. But to the difference with LODE, we do not make formal mappings, functional property restrictions and do not conform to one single vocabulary for our properties. We do not benefit from the other models or vocabularies directly, but stay open to more diversity. The other vocabularies can be connected to SEM via our placeholders for Role and Type.

3.3 CIDOC-CRM

CIDOC-CRM [2] was created for describing museum artifacts, for enhancing their exchange across musea. The whole model is quite large, it contains 140 classes and 144 properties. A subset of these can be used to represent events.

Roles are represented in the same fashion as in SEM: as constraints on a property. But unlike SEM, the Role can only be assigned to the Actor. Types can apply to all entities of CIDOC-CRM, but time-stamps (modeled with a two-position pattern) can only apply to TemporalEntities: Roles, Types and other event constituants cannot be time-stamped. We generalize the CIDOC-CRM's model with SEM, and add the representation of View.

4 SEM Prolog API

In this section we describe the SEM API and illustrate its use with a scenario.[20] The code of the API can be downloaded from the GIT repository at `http://eculture.cs.vu.nl/poseidon/sem.git`. We developed an open source SEM SWI-Prolog API for two reasons. 1) To ease the creation of SEM instances. 2) To make it easy to perform complex queries on SEM instances. In general, the use of an RDF repository for querying or adding new instances requires a user to know the structure of the model and the names of the properties and classes used. This can be a bottleneck, especially for composite or complex queries. The SEM API alleviates this problem by proving a number of predefined, simple functions for frequent operations. Also, it takes care of property constraints, allowing transparent queries over the various representations, *i.e.* without having to query for each of the three representations separately. The API provides two types of interactions with SEM: assertions and queries. When asserting instances through the API, a complete SEM RDF graph is generated from minimal user input. For example:

```
assert_event_actor(ex:'FirstPoliceAction',   % event
                   dbpedia:'Netherlands',    % actor
                   wordnet:government-1).    % actor type
```

asserts a sem:hasActor property between ex:FirstPoliceAction and dbpedia:Netherlands, states that the former is of rdf:type sem:Event and the latter sem:Actor, that dbpedia:Netherlands has sem:actorType wordnet:government-1 and that this is of rdf:type sem:ActorType. The following query:

```
event_actor(Event, Actor, ActorType).
```

[20] Since this API is a Prolog module, we write about 'variables' and 'instantiation' instead of 'classes' and 'individuals'. Words that start with a capital in code examples are variables.

finds all possible instantiations of the variables Event, Actor and ActorType, such that the Actor has the actor type ActorType and Event sem:hasActor Actor. Filling in a value for any of these variables will constrain the results. Missing values are indicated with a hyphen. Variables that are irrelevant can be left out by prefixing them with an underscore. For example, if you want to find all the names of places where someone acted as a attacker and who is victim in a different event, ordered by distance[21] to the middle of the Gulf of Aden, you could formulate the following query:

```
space_nearest(linestring([point(11.46,44.18),
                          point(13.67,52.03)]), Place),
event_place(Event, Place, _PlaceType),
% reusing the Actor variable binds it to the same value
role(Event, sem:hasActor, Actor, ex:attacker),
role(Event2, sem:hasActor, Actor, ex:victim),
rdfs_label(Place, PlaceName).
```

The API is integrated with the SWI-Prolog space [8] and semweb packages [11], which together, with built in Prolog predicates, provide: indexing for geopositions and literal values (strings, numbers, dates, etc.), and reasoning. More specifically, the integration with the SWI-Prolog semweb package provides backward chaining RDFS++ reasoning and access to OWL reasoners through a DIG interface and to OWL and SWRL via Thea [9]. The space package includes import facilities for various ways to encode location in RDF, such as the W3C WGS84 vocabulary[22] and GeoRSS[23]. It provides common query types like nearest neighbor, intersection, containment, and within range. The geometries encountered in RDF are converted to a common shape format (*e.g.* `point`, `linestring`, `polygon` terms), which makes it easier to query and integrate places that come from different sources, like DBpedia, GeoNames, and Freebase. The space package supports KML[24] output to display results, and can crawl the Linked Data[25] graph along owl:sameAs, skos:exactMatch, or skos:closeMatch properties, to acquire more information about the context of events. The SEM API normalizes various time representations that can be encountered in RDF, like ISO 8601[26] as literals, or TIDES TIMEX[27] expressions as XML literals.

4.1 Example Scenario: Spatial Queries over Piracy Events

In this section we illustrate how to use the SEM API in conjunction with the space package to do spatial queries on event data from various sources on the web. We focus on our second use case: maritime safety and security events, specifically piracy events in the Gulf of Aden.

[21] the `space_nearest` predicate comes from the space package and provides a nearest neighbor query facility

[22] `http://www.w3.org/2003/01/geo/`

[23] `http://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/`

[24] `http://code.google.com/apis/kml/`

[25] `http://linkeddata.org/`

[26] `http://en.wikipedia.org/wiki/ISO_8601/`

[27] `http://timex2.mitre.org/`

In the past years pirate attacks in the Gulf of Aden have intensified. These attacks are reported by the crew of the victim ships to the International Chamber of Commerce's ICC-CCS. The reports are listed on their website.[28]. We used the SEM API to generate event instances from the reports of the years 2006 to 2009. These description of the events contain an actor type (type of the victimized ship) and event type (type of the situation, *e.g.* hijacking), and place and time (as timestamps) of the event. An example of an event assertion, which demonstrates anonymous actors that do have a type, and anonymous places with coordinates, is listed below. In this way all the events can be asserted.

```
% event is of type hijacking
assert_event(ex:event_2008_164, ex:hijacked),

% an actor of the event is an anonymous ship of type yacht
assert_event_actor(ex:event_2008_164, -, ex:yacht),

% at unnamed place of type "out at sea" at given coordinates
assert_event_location(ex:event_2008_164,
                      -, ex:out_at_sea, point(9.5899,51.635)),

% event at ISO 8601 date time in UTC
assert_event_time(ex:event_2008_164,
                  literal(type(xsd:dateTime,
                               '2008-08-04T03:00+0400Z'))).
```

As a measure of protection against the piracy attacks, the Internationally Recommended Transit Corridor (IRTC) was created.[29] Later, the location of this corridor was changed. Using Google Earth we created a KML shape of the coordinates of the transit corridor, and converted it to RDF with GeoRSS with the space package. Combining the events with the timestamped representation of the two transit corridors, it is possible to query the number of piracy events that happened inside or within the safety corridors while they are in effect or at other periods of time. These counts are shown in table 1. Our KML export of piracy events from all over the world can be downloaded from `http://semanticweb.cs.vu.nl/poseidon/piracy.kmz` for viewing in Google Earth. From these counts we can see that the pirate attacks mainly happen at the location where a safety corridor is in effect. A possible reason for this could be that the pirates, knowing that the ships would mostly transit through the corridor, focused their attacks on these zones. Another reason could be that ships traveling through the corridors are more inclined to report events to the ICC-CCS than ships traveling elsewhere. As we have all the information represented in RDF, we can also compare the numbers for the cases where the event was of type hijacked: when a hijacking was succesful. This could be done specifically for any of the ship types, for various periods in time, or for different places, because all of these facets are represented in SEM. The number of actual hijackings are shown between parentheses in table 1, next to the total number of events. There does not seem to be a large difference in the percentage of successful hijackings

[28] `http://www.icc-ccs.org/`

[29] `http://gcaptain.com/maritime/blog/ukmto-transit-corridor/`

between the different areas. However, there is a large difference with respect to the types of events that happen in the Gulf of Aden and in the rest of the world, for example, in the Malacca Strait. The former are mainly attempted and successful hijackings, while the latter are mainly boardings.

date		**area**				
begin	end	MSPA	IRTC west	IRTC east	entire gulf	
2008-05-30	2008-08-18	0	0	0	9 (2)	no safety corridor
2008-11-13	2009-02-01	19 (3)	1 (0)	0	40 (9)	MSPA in effect
2009-02-01	2009-04-22	1 (0)	8 (2)	12 (2)	36 (7)	IRTC in effect

Table 1. Number of pirate attacks and successful hijackings (between parentheses) in the areas of the safety corridors during three equally long periods in time.

5 Discussion and Future Work

We have presented SEM, a model to represent data from the Web as events, in and across various domains. Because it is not possible to control or redesign data sources on the Web, SEM's design choices are driven by minimal commitment. For example, we use no cardinality restrictions, no functional or inverse functional properties and SKOS links to other vocabularies and event models to avoid inheriting their constraints. By use of the OWL 2 features such a punning we allow types to be instances as well as classes.

From two use cases, it became apparent that representing viewpoints as property constraints such as opinions or the role of participants in an event, are crucial for modeling data from different sources. We have shown three alternative ways to model these property constraints. A Prolog API translates between these representations, and facilitates easy access to SEM. The API enables a user to create individuals without having to remember the name of all of SEM's classes, properties or SEM's structure. The API is integrated with the existing Prolog semweb and space packages, which facilitates easy prototyping and connecting to Linked Data.

Several features of SEM have been inspired by a thorough review of other event models. In this context, SEM fulfills the need for a model that is on one side open to the variety of data on the Web and on the other side capable of modeling the complex aspects of events such as conflicting viewpoints and time bounded validity of facts.

A few things that fall outside the scope of SEM are the following. We do not define an explicit vocabulary of types (*e.g.* of events, roles etc.), but we provide placeholder classes for using other vocabularies instead. We do not deal with relations between events other than sem:hasSubEvent. These are deferred to other ontologies. We do not have strong mappings to other models. This has both a positive side and negative side: on the one hand, SEM stays independent from the formalizations and restrictions that other models or vocabularies have defined, but on the other hand, SEM cannot use their expressivity for direct inferences. However, since it is easier to add statements than to remove them from an ontology we choose to specify less, rather than more.

In the future we will investigate how other models, like GEM or F, can be used in combination with SEM to model other event properties than sem:hasSubEvent, for example, causality and correlation.

Acknowledgements

Part of this work has been carried out as a part of the Poseidon project in cooperation with Thales Nederland under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK03021 program. We would like to thank Chiel van den Akker and Anna Tordai.

References

1. Stefano Borgo Claudio Masolo, Aldo Gangemi, Nicola Guarino, Alessandro Oltramari, and Luc Schneider. Wonderweb deliverable d18. ontology library library. Technical report, ISTC-CNR WonderWeb project, 2003.
2. Nick Crofts, Martin Doerr, Tony Gill, Stephen Stead, and Matthew Stiff (editors). Definition of the cidoc conceptual reference model. online, November 2009.
3. Carl Lagoze and Jane Hunter. The abc ontology and model. In *Proceedings of the International Conference on Dublin Core and Metadata Applications (DMCI 2001)*, pages 160–176. National Institute of Informatics, Tokyo, Japan, 2001.
4. Yves Raimond and Samer Abdallah. The event ontology. online, 2007. http://purl.org/NET/c4dm/event.owl.
5. Tuukka Ruotsalo and Eero Hyvönen. An event-based approach for semantic metadata interoperability. In *6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*, pages 407–420, November 2007.
6. Ansgar Scherp, Thomas Franz, Carsten Saathoff, and Steffen Staab. F—a model of events based on the foundational ontology dolce+dns ultralight. In *International Conference on Knowledge Capturing (K-CAP)*, Redondo Beach, CA, USA,, September 2009.
7. Ryan Shaw, Raphaël Troncy, and Lynda Hardman. Lode: Linking open descriptions of events. In *4th Annual Asian Semantic Web Conference (ASWC'09)*, pages 153–167, Shanghai, China, December 6-9 2009.
8. Willem Robert van Hage, Jan Wielemaker, and Guus Schreiber. The space package: Tight integration of space and semantics. In *Proceedings of the 8th International Semantic Web Conference Workshop: TerraCognita*, 2009.
9. Vangelis Vassiliadis, Jan Wielemaker, and Chris Mungall. Processing owl2 ontologies using thea: An application of logic programming. In *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, 2009.
10. Utz Westermann and Ramesh Jain. Toward a common event model for multimedia applications. *IEEE MultiMedia*, 14(1):19–29, 2007.
11. Jan Wielemaker, Zhisheng Huang, and Lourens van der Meij. Swi-prolog and the web. In A. Bossi, editor, *Theory and Practice of Logic Programming*, volume 8, pages 363–392. Cambridge University Press, 2008.
12. Michael F. Worboys and Kathleen Hornsby. From objects to events: Gem, the geospatial event model. In Max J. Egenhofer, Christian Freksa, and Harvey J. Miller, editors, *Third International Conference on Geographic Information Science, GIScience 2004*, volume 3234 of *Lecture Notes in Computer Science*, pages 327–344. Springer, 2004.