Design and use of the Simple Event Model (SEM)

Willem Robert van Hage [a,*] Véronique Malaisé [a] Roxane Segers [a] Laura Hollink [b]
Guus Schreiber [a]

[a] *Web & Media Group, Vrije Universiteit Amsterdam, de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands*
[b] *Web Information Systems, Technical University Delft, Mekelweg 4, 2628 CD Delft, 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 are central elements in the representation of data from domains such as history, [1] cultural heritage [3,6], multimedia [12] and geography [14]. 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 achieve interoperability with data sets from various domains.

The diversity and openness of the Web poses challenges on the design of an event model. We show how we minimally commit our model by assuming nothing about the used vocabularies or the structure of the data. We draw further requirements from two domains: historic events and events in the maritime safety and security domain. Although very different, these domains do have commonalities: from both 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 during which a role is valid and the authority according to whom this role is assigned.

Several event models or ontologies have been published over the past years [3,4,5,6,7,8,12]. They differ in their focus (class or property centered), domain specificity, size and level of formalization. We review and compare the design choices of existing models to explicate what SEM contributes to the existing literature. In addition, we show how we create mappings from our model to existing event models.

Learning to properly use an event model is hard and

* Corresponding author. Tel: +31 20 598 8785
Email addresses: W.R.van.Hage@vu.nl (Willem Robert van Hage), vmalaise@few.vu.nl (Véronique Malaisé), rh.segers@cs.vu.nl (Roxane Segers), l.hollink@tudelft.nl (Laura Hollink), Guus.Schreiber@vu.nl (Guus Schreiber).

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

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 by hiding some details of the model to a user and it is integrated with the SWI-Prolog space [10] and semweb packages [13]. The combination of these three provides a fast and efficient indexing for geopositions as well as RDF and literals (strings, numbers, dates, etc.), along with spatial, RDFS, OWL, and rule reasoning.

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 provide a short overview of event models in Section 5; we review in particular three models as representative examples and compare them to SEM. The API and its coupling with the other Prolog packages is demonstrated in Section 3, on examples from the maritime domain. We conclude with a discussion and future work in Section 6. The RDF code of SEM can be found online [2] .

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 a user 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 (*e.g.* with SPARQL) 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, the landing of UFO in Roswell or the elections of G.W. Bush, 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 rdfs:domain and rdfs:range to non-restricting classes (*i.e.* that can not be proved to be disjoint to any other SEM classes and hence do not restrict the domain or range of the property). 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. In principle these mappings can be treated as documentation of the meaning of the SEM constructs and not as parts of their formal definition. SKOS mapping properties do not transfer OWL consequences. If stronger mappings are necessary it is possible to decide to momentarily replace the appropriate mappings relations with stronger versions, like owl:sameAs or rdfs:subClassOf.
- Every class and property is optional and can be duplicated, *i.e.* we do not model cardinality restrictions.

[2] http://semanticweb.cs.vu.nl/2009/11/sem/

[3] http://www.w3.org/2004/02/skos/

Specifically we do not enforce the use of sem:types (not rdf:types, which *are* necessary).
- 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.
- We pay a special attention to graph patterns for efficient reasoning, to compensate for the limited formal constraints of SEM.

These implications are in line with the view of the Web outlined in chapter 1 of Allemang and Hendler [1].

2.2. *SEM Specification*

In this section we describe how these modeling decisions are implemented in SEM. First we discuss the core classes and the properties that make up SEM; then how to model views, roles and temporary validity as constraints on properties; and finally how to model time and space with symbols (*c.q.* URIs) or values (*c.q.* coordinates). We give a simple and more elaborate example of how an event from the historical domain can be modeled in SEM in Figure 3 and 4. These examples represent information from the following sentence, [4] which represents a typical sentence from the historical domain:

> *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 / 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. The next subsections describe the classes, properties and constraints of 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 [5] 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 Figure 4 and its sem:PlaceType "republic" (tgn:82171) are borrowed from the Getty Institute's Thesaurus of Geographical Names (TGN). [6]

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 other properties. This can be done by using Alan Rector's Value Sets and Value Partition patterns. [7] Having explicit sem:Type classes provides a placeholder to define these patterns. An example of Rector's pattens applied to SEM can be found in Section 2 of [9]. Also, having an explicit sem:Type class makes it easier to define applications like facet browsers on top of SEM. You can simply define your interface to show all sem:Type instances, instead of having to query for all top classes from the domain vocabularies.

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 [8] 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. [9]

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 dependent types, like "mother", or temporary types. The latter can

[4] 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.

[5] The sem:Constraint class sem:Role has also an associated sem:RoleType.

[6] `http://www.getty.edu/research/conducting_research/vocabularies/tgn/`

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

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

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

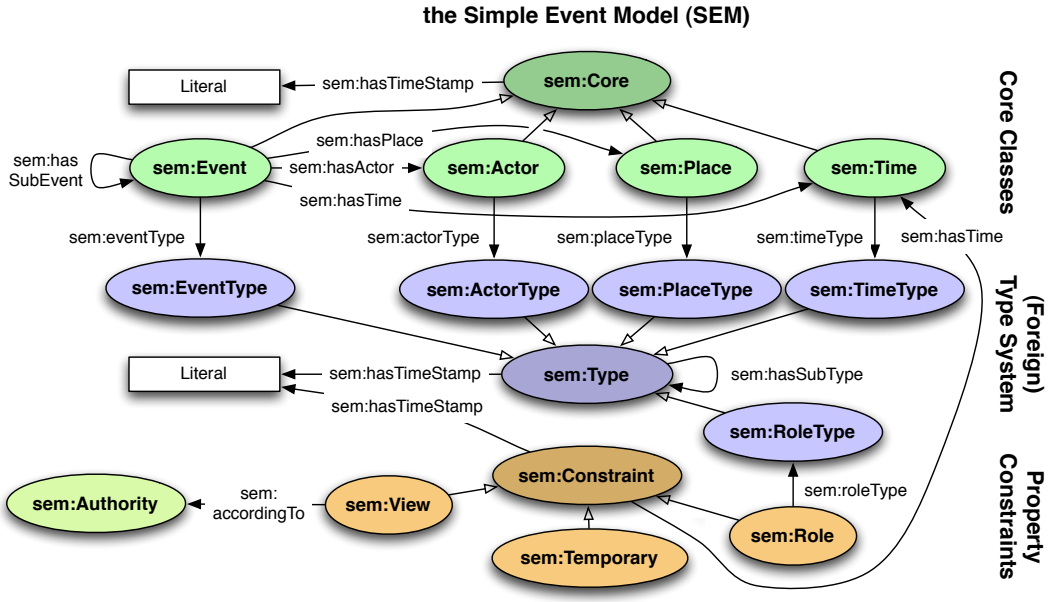

Figure 1. The classes of the Simple Event Model. Arrows with open arrow heads symbolize rdfs:subClassOf properties between the classes. Regular arrows visualize rdfs:domain and rdfs:range restrictions on properties between instances of the classes.

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", "landing area". These roles do not depend on external conditions (like the fact to have a child defines 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 "Indonesia" as a "republic" holds from 1945 on, at least according to the Indonesians. This notion can be modeled with sem:View, which is used to define points of view and opinions. Indonesia, in 1947, has either the type "republic" or "controlled region", depending on the source of information. 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). 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 provide this.

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 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 3, automatically processes constraints. In Figure 4 one actor plays two different roles in one event. Modeling the case where one actor plays different roles in two different events can be accomplished by having two separate blank nodes indicating the different roles that both point to the same URI indicating the actor with the rdf:value property.

Properties SEM's properties are divided in three kinds: sem:eventProperty, sem:type properties and a few miscellaneous properties like sem:accordingTo and sem:hasTimeStamp's subproperties. The sem:eventProperty relates sem:Events to other individuals. A sem:type relates individuals of the sem:Core class [10] 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

[10] They also relate sem:Role to its sem:RoleType.

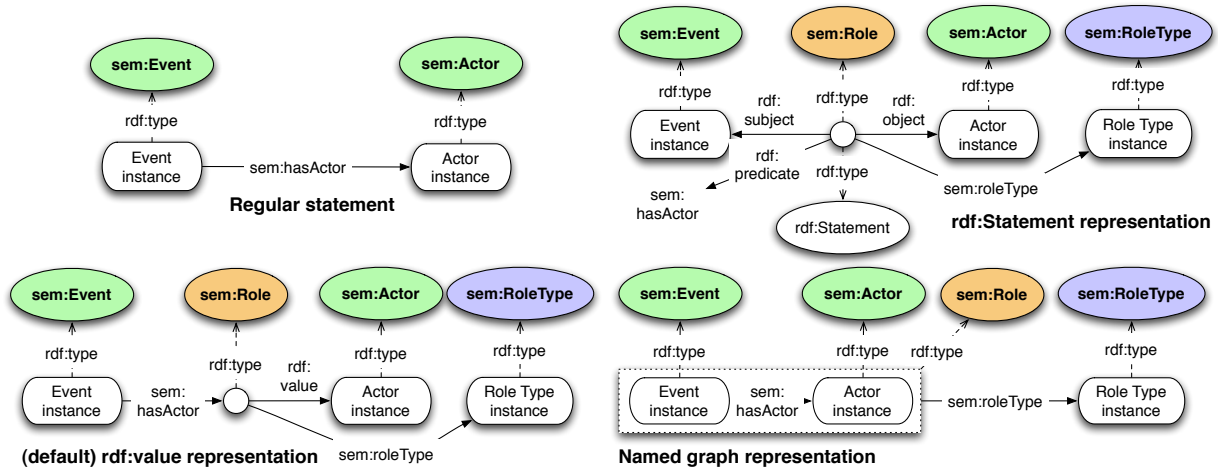

Figure 2. Three alternative representations of property constraints in SEM.

reduces the strain on reasoners, because the property points directly 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 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. We decided not to model subtypes as subproperties of skos:broader/narrower, because we do not want to inherit the disjointness of skos:broader with skos:related. More specific relations between events and types are not part of SEM and should be taken from other ontologies, like GEM [14].

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.

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, [11] both of which support the ISO 8601 [12] 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, [13] or wgs84:lat and wgs84:long. [14] Complex geometries like polygons can be encoded in GML [15] in an rdf:XMLLiteral pointed at by georss:where.

Example The historical example mentioned in the introduction of this section can be expressed in SEM as shown in Figure 3 and 4. The former shows a simple example, which disregards the differences of opinion between the various parties. The latter takes into ac-

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

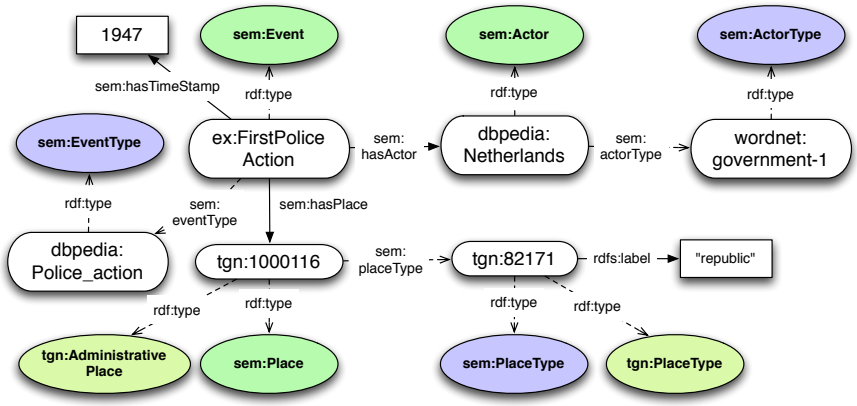

Figure 3. A simple representation of the historical example event in SEM.

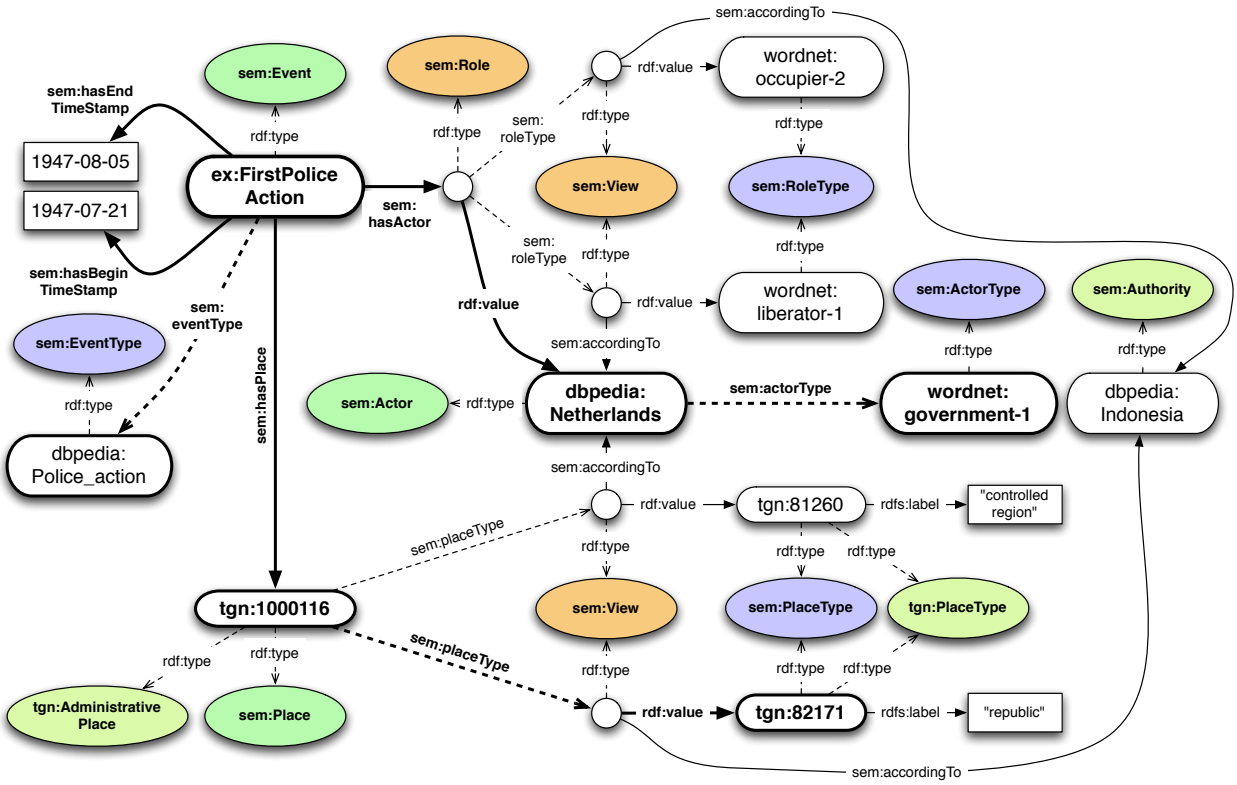

Figure 4. A more elaborate representation, including Role and View property constraints of the historical example event in SEM.

count all views and roles of the sentence. The instance of the event ex:FirstPoliceAction in Figure 4 has a blank node as range for the property sem:hasActor, to express the sem:Role that the sem:Actor (the Netherlands) plays in the context of this given event. As there are two sem:Roles for the Netherlands in the context of the event, according to two different points of view (those of the Netherlands itself and Indonesia), there are two sem:roleType properties leading to two separate blank nodes representing the two different sem:Views on the type of the role (liberator and occupier). SEM allows for the representation of different roles for the same actor within one event. The resolution of the interpretation of such statements, however, is out of the scope of the model. Likewise, the example shows two views on the sem:placeType property for the same instance at which the example event takes place. In this way, it is possible to represent very complex statements that are contra-

dictory depending on whom you ask, which correspond to queries historians are interested in.

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, going beyond purely "factual" event models. However, SEM does not model relationships between events like causality, or specific properties about the semantics of the way an actor participates in an event (e.g. active/passive participation). Other knowledge patterns like the D&S from DUL [2], 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. [16] 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. The reason for creating SEM as a distinct event model will be shown by a detailed description of three representative event models SEM has been partially mapped to.

The forgiving modeling style used in SEM has consequences for the way SEM can be used. Without strong semantic constraints many kinds of automatic validation techniques can not be applied. Conflicting views can exist at the same time and be compared, but they are not automatically resolved. Much of the responsibility of enforcing consistency is taken from the SEM and given to the application layer.

3. SEM Prolog API

In this section we describe the SEM API and illustrate its use with a scenario. [17] The code of the API can be downloaded from the GIT repository [18] .

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 providing 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 possible representations separately (see Figure 2).

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 with 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.

To find all possible instantiations of the variables Event, Actor and ActorType, such that the Actor has the actor type ActorType and Event sem:hasActor Actor, you can use the following query:

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

Filling in a value for any of these variables will constrain the results: specifying

```
event_actor(Event, Actor, ex:yacht).
```

will retrieve all the Events in which the participating Actor was a yacht. Missing values are indicated with a hyphen. Variables that are irrelevant can be left out by prefixing them with an underscore.

The API is integrated with the SWI-Prolog space [10] and semweb packages [13], which together, with built in Prolog predicates, provide indexing and reasoning for geopositions and literal values (strings, numbers, dates, etc.). 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 [11]; the integration with the space package provides proximity and inclusion reasoning such as "space_nearest", "space_within_range". The space package includes import facilities for various ways to encode location in RDF, such as the W3C WGS84 vocabulary and GeoRSS.

The geometries encountered in RDF are converted to a common shape format (*e.g.* `point`, `linestring`,

[16] Currently, SEM is mapped to concepts from: LODE, OpenCYC, CIDOC-CRM, Dublin Core, DOLCE Lite, SUMO, the Event Model (Queen Mary University), FOAF, OWL time, the W3C WGS84 vocabulary, and CultureSampo

[17] 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.

[18] `http://eculture.cs.vu.nl/git/poseidon/sem.git`

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 [19] output to display results, and can crawl the Linked Data [20] 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 [21] as literals, or TIDES TIMEX [22] expressions as XML literals. All these functionalities in the API are useful for data coming from various domains, but espcially for those in which geospatial reasoning plays an important role. The functionalities of the API will be demonstrated in our second use case: maritime safety and security events, specifically piracy events in the Gulf of Aden.

4. Example Scenario: Modeling Piracy Events in SEM using the API

To illustrate modeling in SEM and in particular the use of the SEM API, we present the following example. The International Chamber of Commerce lists all pirate attacks that are reported by victim ships. The reports are available on the ICC-CCS website. [23] With the kind permission of ICC-CCS, reports of the years 2006 to 2009 were crawled, parsed and finally converted to RDF using the SEM API. The resulting SEM descriptions of the events contain an actor type (type of the victimized ship), an event type (type of the situation, *e.g.* hijacking), and the place and time (as timestamps) of the event. Figure 5 shows one SEM instance of a piracy event, which has anonymous actors with a type, and anonymous places with coordinates. The event assertions (automatic instantiation with the SEM API) are listed below.

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

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

% the unnamed place has coordinates
% and the type "out at sea"
assert_event_location(ex:event_2008_164,
        -, ex:out_at_sea, point(9.5899,51.635)),
```

[19] http://code.google.com/apis/kml/
[20] http://linkeddata.org/
[21] http://en.wikipedia.org/wiki/ISO_8601/
[22] http://timex2.mitre.org/
[23] http://www.icc-ccs.org/

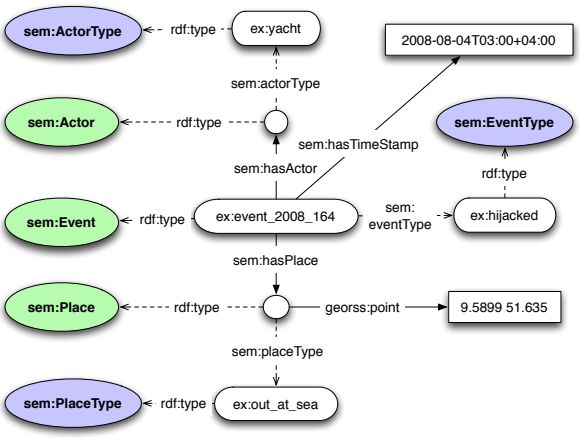

Figure 5. A representation of a piracy event in SEM.

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

Using Google Earth we created a KML shape of the coordinates of the two safety transit corridors in the Gulf of Aden, that were created as a measure to prevent the increase of piracy cases reported in that area. We converted them 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 in time. These counts are shown in Table 1. 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 the type hijacked. 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 between the different safety corridor areas. However, querying the whole piracy reports dataset, we notice that there is a large difference with respect to the

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

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.

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.

These examples show that the SEM API eases the burden of instance creation, data conversion and enables a user to perform easily multidimensional queries, involving RDF, spatial and temporal reasoning, in an integrated manner.

5. Related Work

Various models have been proposed for representing events on the Semantic Web, such as the Event Ontology (EO) [5] [24] , Linking Open Descriptions of Events (LODE) [8] [25] , the F-Model (F) [7] [26] and the event ontology used in CultureSampo [6]. Some more general models for semantic data organization also include event models like CIDOC-CRM [27] and the ABC ontology[4].

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 (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. F, CIDOC-CRM, and ABC are class-based models, while others are rather 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 difference in scope translates in the fact that some models aim for a minimal representation of events (EO, LODE) while others aim at representing a broader range of event constituants and their relations (CIDOC-CRM and F). Some models are constrained by the use of OWL constraints (LODE: rdf:domain, rdf:range, rdfs:subPropertyOf, owl:sameAs , 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, borrowing classes and properties from DUL [2]) 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 a 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, property based model with few classes, some restrictions and a higher level of formalization.

CIDOC-CRM as a domain-specific and class-based event model, 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 show their advantages and drawbacks, and the way SEM answers these drawbacks.

Event Ontology The Event Ontology (EO) [28] 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. [29]

[24] http://motools.sf.net/event/event.html
[25] http://linkedevents.org/ontology/
[26] http://isweb.uni-koblenz.de/eventmodel/
[27] http://cidoc.ics.forth.gr/official_release_cidoc.html
[28] http://motools.sf.net/event/event.html
[29] http://www.foaf-project.org/

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, the sem:Event class is optional. It was important for us to be able to model a (potential) actor or a place independently of the events they might participate in.

LODE LODE [8] 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 to 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. [30] In SEM, we also adopt the principle of using external vocabularies for modeling properties that are beyond the model's scope, like the causality. But contrary to LODE, we do not make formal mappings and functional property restrictions, and we 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. This way, the borrowed hierarchies are kept external from our Core classes.

CIDOC-CRM CIDOC-CRM [3] 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, to fulfill our requirements.

Comparison of SEM with Related Work SEM gathers the elements that give a light-weight description of events, but without importing strong semantic definitions that easily lead to inconsistency, e.g. owl:FunctionalProperty, owl:disjointWith. In addition to this, SEM specifies the necessary additions for dealing with heterogeneous and messy data from the web, *i.e.* foreign types, constraints, and authority. In principle, SEM is meant to record different, possibly conflicting, descriptions of an event as co-existing facts. Therefore we avoid constructs that lead to inconsistency due to such conflicts. These can be added later by applications if resolution of the differences is needed.

6. 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 we use SKOS to link to other vocabularies or event models to avoid inheriting their constraints. By use of the OWL 2 features such as 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, is crucial for modeling data from different sources. We take into account 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. It also enables an integrated spatial and RDFS reasoning.

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.

[30] F specializes D&S patterns from DUL.

A few aspects 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 like 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] D. Allemang, J. Hendler, Semantic Web for the Working Ontologist, Morgan Kaufmann, 2008, ISBN-13: 978-0-12-373556-0.

[2] S. B. Claudio Masolo, A. Gangemi, N. Guarino, A. Oltramari, L. Schneider, Wonderweb deliverable D18. ontology library library, Tech. rep., ISTC-CNR WonderWeb project (2003).

[3] N. Crofts, M. Doerr, T. Gill, S. Stead, M. S. (editors), Definition of the CIDOC conceptual reference model, online (November 2009).

[4] C. Lagoze, J. Hunter, The ABC ontology and model, in: Proceedings of the International Conference on Dublin Core and Metadata Applications (DMCI 2001), National Institute of Informatics, Tokyo, Japan, 2001.

[5] Y. Raimond, S. Abdallah, The event ontology, online (2007).
URL http://purl.org/NET/c4dm/event.owl

[6] T. Ruotsalo, E. Hyvönen, An event-based approach for semantic metadata interoperability, in: 6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007), 2007.
URL http://data.semanticweb.org/conference/iswc-aswc/2007/tracks/research/papers/407

[7] A. Scherp, T. Franz, C. Saathoff, S. 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,, 2009.

[8] R. Shaw, R. Troncy, L. Hardman, LODE: Linking open descriptions of events, in: 4th Annual Asian Semantic Web Conference (ASWC'09), Shanghai, China, 2009.

[9] W. R. van Hage, V. Malaisé, G. de Vries, G. Schreiber, M. van Someren, Abstracting and reasoning over ship trajectories and web data with the simple event model (SEM), Multimedia Tools and Applications to appear: Special issue on Events in Multimedia.

[10] W. R. van Hage, J. Wielemaker, G. Schreiber, The space package: Tight integration of space and semantics, in: Proceedings of the 8th International Semantic Web Conference Workshop: TerraCognita, 2009.

[11] V. Vassiliadis, J. Wielemaker, C. 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.

[12] U. Westermann, R. Jain, Toward a common event model for multimedia applications, IEEE MultiMedia 14 (1) (2007) 19–29.

[13] J. Wielemaker, Z. Huang, L. van der Meij, SWI-Prolog and the web, in: A. Bossi (ed.), Theory and Practice of Logic Programming, vol. 8, Cambridge University Press, 2008.

[14] M. F. Worboys, K. Hornsby, From objects to events: GEM, the geospatial event model, in: M. J. Egenhofer, C. Freksa, H. J. Miller (eds.), Third International Conference on Geographic Information Science, GIScience 2004, vol. 3234 of Lecture Notes in Computer Science, Springer, 2004.
URL http://dblp.uni-trier.de/db/conf/giscience/giscience2004.html#WorboysH04