ONTOLOGY-BASED TASK SIMULATION

Martin Raubal and Werner Kuhn
Institute for Geoinformatics
University of Münster

Abstract
Information services assist people in their decision-making during the performance of certain tasks. In order to determine if a data source, which commits to a given ontology, can be employed for a service, the service provider needs to evaluate its usability and utility for the decision-making process. We propose to do this by simulating with the ontologies the tasks to be supported by the service. Such a simulation needs to access data about entities based on the actions they afford and the events they participate in. This requires that ontologies include information about these affordances and events. The paper demonstrates a formalized framework, which satisfies this requirement by including functions in the ontologies and making the specifications executable. A real-world scenario for a navigation service—instructions for crossing a river by car—demonstrates the applicability and benefits of the approach in a dynamic scenario.

1. Introduction

People use information services for assistance in situations where they have to make decisions. Consider, for example, a car navigation service, which calculates an optimal route and provides a sequence of instructions for driving along this route. Each instruction guides the user from one decision point to the next. Such a service uses data sources that commit, at least implicitly, to specific application ontologies, such as a navigation ontology, which may themselves commit to a domain or upper-level ontology (Kuhn & Raubal, 2003). Instructions should communicate to the user what she can *do* in a decision situation, i.e., the service needs to know about the roles that entities (represented by the data) play in actions (Fonseca, Egenhofer, Davis, & Câmara, 2002). Current ontologies are static and entity-based and do not specify these actions. Their emphasis is on attributes and relationships rather than operations (Kuhn, 2001). The importance of action-oriented approaches in GIScience was recently highlighted by an international research workshop[1].

The omission of actions and their context-dependent semantics from ontologies can lead to counterproductive assistance when using a navigation service. A German motorist experienced this when he drove his car into a river after following instructions from its navigation service (Hamburger-Abendblatt, 1998; McKie, 1998; Neumann, 1999). The system implied the presence of a bridge when it should have indicated a ferry. Although both bridge and ferry are links that afford river crossing to cars, the semantics of this action is crucially different depending on which entity it is applied to. Bridges are static and therefore continuously afford crossing. Ferry lines have a dynamic component and can only be used by a car when the car gets on the ferry. The accident of the motorist would not have occurred had the underlying ontology included such *dynamic aspects.*

This paper presents a method to evaluate whether data committing to a particular application ontology can be usefully employed by an information service. The core idea is to simulate with the ontology the specific tasks to be supported by the service. This idea of *ontology simulation* is new and represents the main contribution of the paper. Such a simulation requires first that the ontology comprises not only entities, relationships, and attributes, but also actions, because these will be referred to in the simulations. Secondly, the actions need to be specified in an executable form, to enable the simulation without the need for further programming. The formalized[2] approach shown here satisfies both requirements. It uses a functional specification (Frank & Kuhn, 1999), where ontologies are specified algebraically, and simulations are run in their initial model (Goguen, Thatcher, & Wagner, 1978).

Section 2 describes in more detail the case study of crossing rivers that we use to demonstrate the need for and applicability of the method. In section 3 we review previous work on ontologies for information systems from a dynamic point of view, and describe the general ideas behind ontology-based task simulation. Section 4 discusses the two main requirements ontologies need to satisfy in order to be used for task simulation, i.e., action specification and executability. It further introduces our formalization method, fulfilling these requirements. The formalized framework is applied to the case study in section 5 by simulating the task of crossing a river by car based on two different versions of an application ontology. The final section presents conclusions and directions for future research.

[1] http://www.spatial.maine.edu/~actor2002/

[2] In the sense of expressed in a formal language.

2. Case study – car navigation service

In order to demonstrate the proposed method for evaluating data sources through simulations in their ontologies, we use the real-world example from the introduction. It illustrates the kind of situation in which an evaluation is needed.

2.1 Car navigation services

A car navigation service is a special case of a location-based service. Its main task is to guide a driver from a starting point to a destination by calculating an optimal route and giving instructions for driving along this route. Traditional car navigation systems (White, 1991) integrate a positioning system, car sensors, and a local copy of the map data. Their derived turn-by-turn instructions have two major drawbacks: they are often based on out-of-date data and do not consider real-time traffic information. Car navigation services delivered via wireless communication networks solve these problems by including access to real-time traffic information and detailed up-to-date mapping data (Coleman, 2001).

Digital map data for navigation is complex and includes multiple aspects of road networks. The geometry of the street network is needed for comparing and matching the car's path with the street locations in the database. Topological information is used as input to optimal path algorithms. Data producers have handled these aspects successfully and data standards for describing road networks and transferring the data have been developed, such as the international standard GDF (Geographic Data File) (Konijn, 1998). These standards typically focus on the relevant geographic entities, their attributes, and relationships, but do not relate them directly to actions and related decisions that the data can support. In this paper we argue that the specification of real-world actions in which the features participate is an essential requirement for assessing data usability (Riedemann & Timm, 2003) and therefore needs to be represented explicitly. The case study from the navigation sector is used to demonstrate why and how.

2.2 Background of the case study

On December 25, 1998, a German motorist drove his car into the Havel river near Caputh in eastern Germany, after following instructions from its navigation system. A ferry operates across the river at that point, but the system was unaware of the crucial difference between crossing a river on a ferry and on a bridge (Hamburger-Abendblatt, 1998; McKie, 1998; Neumann, 1999).

While one can muse about non-technical factors leading to the accident, we hypothesize that the core problem was one of semantics: When calculating a route, ferry lines and bridges are treated equally, as parts of the road network. When giving instructions, however, the different semantics of moving across the river need to be taken into account. A bridge can typically be crossed at any time, while a ferry operates at certain times only, and the driver should be instructed to wait for the next crossing.

It has proven practically impossible to obtain specifics about the data and software that were used in that car. However, the fact that navigation systems use road data for at least two purposes, calculating routes and giving instructions, is undisputed (Timpf, Volta, Pollock, & Egenhofer, 1992). The software modules responsible for driving instructions assume that a link in a road network affords driving. If road data supporting route calculation (and treating ferry lines as regular parts of the network) are used to produce driving instructions, motorists will be guided into the water. This reasoning underlies our hypothesis about the semantic causes involved in the accident. We cannot prove this, but use it as an assumption for the sake of our argument: *action semantics is key to assessing data usability*.

One kind of such road data is that following the GDF standard (Konijn, 1998). This standard adheres to a sophisticated semantic model of road data, differentiating three levels (see section 5 for details). It is the responsibility of the software producers to match these levels to the tasks supported by their systems. A mismatch between the two ontologies (both implicit, of the data and of the software) creates a potential semantic problem.

Note that this is not an issue of incorrect data or software. It is one of a mismatch between the semantics of the two. The case appears to demonstrate the danger of separating data from the software that „knows“ about their semantics. Most of today’s navigation systems use a traditional data exchange architecture: data are digitized from maps and delivered (for instance, on commercially available CDs) to software systems that make certain assumptions about their meaning. Splitting the data from the operations lets semantics fall between the cracks.

In an object-oriented service architecture, where data are only accessed by the operations they support, such semantic problems are in principle less likely. They only occur, if the semantics of the (correctly operating) service are not available to the user. Standards for interoperability like those of the OpenGIS Consortium[3] represent a first step in this direction, though they cannot yet solve the issue of service semantics. Indeed, the current deviation from service interoperability by so-called “data interoperability” based on mark-up languages that separate data from operations further delays a solution. The architecture of today’s navigation systems, however, dates from a previous generation and can cause problems wherever the semantics varies from one context to another.

3. ONTOLOGIES AND SIMULATION

This section describes the current use of static ontologies for information systems, with its limitations from a dynamic point of view, and presents the idea of ontology-based task simulation.

3.1 Ontologies for information systems

In the traditional sense, ontology is a subfield of philosophy and can be defined as the science of existence. It tries to determine “the various types and categories of objects[4] and relations in all realms of being” (Smith, 2001, p. 79). From an information systems and artificial intelligence perspective, ontologies are content theories identifying specific classes of objects and relations that exist in some domain (Chandrasekaran, Josephson, & Benjamins, 1999; Frank, forthcoming). Ontologies in the latter sense are language-dependent. Guarino (1998) distinguishes between *Ontology* in philosophy and *ontologies* in knowledge engineering. This paper focuses on ontologies for information systems.

Gruber (1993) defines an ontology as “an explicit specification of a conceptualization.” Ontology designers have to make explicit choices of what to admit as referents in a particular system or language. According to their level of generality, different kinds of ontologies for information systems exist (Guarino, 1998):

1. *Top-level ontologies* describe general concepts, such as space and time, independent from a particular domain.

[3] http://www.opengis.org

[4] Like this and other sources, we use the terms „object“ and „entity“ interchangeably, where there is no danger of confusion between entities in the real world and their representation as objects in a database.

2. *Domain and task ontologies* include the vocabulary for a generic domain (e.g., geospatial) or task (e.g., moving).
3. *Application ontologies* represent concepts, which depend both on a particular domain and task. They are often specializations of domain and task ontologies.

In order to be used in information systems, ontologies need to be formalized and machine-readable. Formal ontological theories include definitions of terms and sets of axioms (Smith, 2002). The current practice of formalizing ontologies uses various subsets and extensions of first-order logic (Welty & Smith, 2001). Examples are the IEEE's Standard Upper Ontology (IEEE, 2003), which is developed with KIF (Genesereth & Fikes, 1992); GOL (Degen, Heller, & Herre, 2002), a framework for ontological engineering, and ONTOCLEAN (Guarino & Welty, 2002), a methodology for ontological analysis based on description logic. The formalized theories emphasize entities with attributes and relationships over processes, actions, and operations. Reasons include (Kuhn, 2001):

- an emphasis on attributes and relationships rather than operations in object-oriented design methods;
- the weakness of logic-based formal languages in dealing with operations and their semantics;
- the lack of understanding how natural language represents actions—witness the noun-bias in WordNet and its theory (Fellbaum, 1998);
- a general lack of theories of function (Barsalou, Sloman, & Chaigneau, forthcoming).

Even if actions are specified in ontologies, the first-order logic formalisms allow only for pre- and post-conditions to be stated, preventing simulations. Supporting human beings in geographic space requires ontologies that are developed paying full attention to both, entities and actions. We have previously suggested (Kuhn, 2001) that the tasks (or actions) to be supported by a Geographic Information System (GIS) should even *determine* the entities that are captured in an ontology.

In this paper, we are less interested in putting labels on various kinds of ontologies than in the purposes that the ontologies serve. The support for simulating tasks that we request may be seen to belong to task or application ontologies (or to yet another kind). However, we avoid further contributions to the inflation of kinds of ontologies and simply suggest that an ontological description of an application needs to be able to answer questions about the tasks that can be performed in it.

3.2 Ontology-driven simulation

Simulation techniques are being widely used in scientific study and many other fields. The Oxford English Dictionary[5] defines simulation as "the technique of imitating the behaviour of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus, esp. for the purpose of study or personnel training."

For economic and ethical purposes it is often useful and necessary to test people's interactions with systems before employing them in the real world. Tests show how systems work under various conditions and with different participants. Pilots, for example, perform extensive tests on flight simulators before flying a real aircraft. Despite this practical aspect, simulation of human behavior in space is a powerful research method to advance our understanding of the interaction between people and their environment. It allows for both the examination and testing of models and their underlying theory as well as the observation of the system's behavior (Frank, Bittner, & Raubal, 2001; Gimblett, Durnota, & Itami, 1997;

[5] http://www.oed.com/

Raubal, 2001b). In the geographical sciences this has led to the emerging field of *geosimulation* research[6], mainly for the study and planning of urban systems.

Simulations imitate processes in certain domains. In order to achieve useful results, a simulation has to be based on a consistent ontology, which represents plausible specifications of the application concepts. Ontologies are also needed for the integration of information from different applications or domains. Various simulations require the combination of higher-level and domain- or task-specific concepts. This can be done by relating top-level, domain, task, and application ontologies. In line with Fonseca's (2002) term of "Ontology-Driven Information Systems" we can speak in this case of *Ontology-Driven Simulation*.

Information services are provided by systems, which are based, implicitly or explicitly, on some application ontology. The services assist people in their decision-making in the real world. It is necessary to test their designs to avoid negative consequences for their potential users. Testing the service and its data source starts with testing the underlying ontology, i.e., the ontology that the data source commits to. If the data have the wrong semantics, services built on them will most likely fail to support their users appropriately. We propose to do such ontology testing by simulating with the ontology the tasks to be supported by the service.

One could argue that the use for simulation purposes is a different concern from that of the specification of application concepts, and that it should not influence the form or contents of ontologies. However, we posit that the possible uses of data are a key aspect of the semantics that an information system ontology should define. The data uses are directly linked to the possible uses of entities in human actions (e.g., moving a car across a river). They should therefore be part of ontological specifications, rather than a special kind of application built on top of ontologies. Our method will show that ontologies and task simulations can indeed be produced in the same language and in one pass.

4. Requirements for simulating tasks with ontologies

Two key requirements ensue from the premise that ontologies should support task simulation. One is the possibility to include actions in the ontologies. The other is to make these action specifications testable, in the sense of letting their designers and users observe the effects they have on entities, attributes, and relationships. This section discusses the two requirements in detail and introduces our method of formalization.

4.1 Ontologies with actions

The first major requirement for simulating a task using an ontology is that the ontology describes sets of actions, in which its entities can participate. A *task* can be defined as a process within a specific time frame. A *process* is "a particular course of action intended to achieve a result" (WordNet[7]). If an ontology is used for simulating courses of action, then it must include the semantics of these actions. Sections 5.2 and 5.3 will present examples from our domain ontology, which demonstrate how action-based ontologies could look.

Current ontology representation frameworks lack satisfactory ways of specifying actions (Mota, Bento, & Botelho, 2002). Cases for action-driven ontologies have been made by (Camara, Monteiro, Paiva, & de Souza, 2000) and (Kuhn, 2001) for geographical space. Their arguments for a dynamical perspective were that geographic entities are continuously being transformed and new ones created, and that the use of information about them is tied to human activities. For information services it is of particular importance what the entities can

[6] http://www.geosimulation.org/

[7] http://www.cogsci.princeton.edu/~wn/

be *used for*, i.e., what kind of action can the user perform with an entity? In general, information services offer pragmatic information to assist the user in the decision to perform a particular action. The usefulness of such information for a given task and context can be measured by the result after an action has been taken (Frank, 2003).

The things that people distinguish in the world depend on the actions they afford. We have previously proposed to enrich ontologies with affordances (Kuhn, 2001; Raubal, 2001b) to compensate for their failure to integrate actions into domain theories. Affordances describe possibilities for actions with reference to a user (Gibson, 1979). Here, we focus on *physical affordances* (Raubal, 2001b), which require bundles of physical properties that match the agent's properties and capabilities. Similar to (Kuhn, 2001) we adopt a broad sense of the term agent: for example, we consider a road to afford driving to a human being in a car. A more restrictive use might see the road as affording support for a car and the car as affording driving actions.

4.2 Executable specifications

The second major requirement for ontology-driven task simulation is that the actions in the ontology are *testable*. Current ontologies are mostly described in static formalisms and do not allow dynamic testing, i.e., the process of viewing an initial situation (state), performing an operation, and then viewing the resulting situation (state). An example is WordNet (Fellbaum, 1998), an online lexical reference system, which is a top-down categorization including nouns, verbs, adjectives, and adverbs. It gives definitions of terms and places them in a hierarchy, but it is not formalized and therefore not testable. Many other efforts to produce ontologies use formal tools—mainly based on first-order logic—and cannot be dynamically tested (in the sense described above). Examples are KIF (Genesereth & Fikes, 1992), Ontolingua (Gruber, 1992), and KADS (Schreiber, Wielinga, de Hoog, Akkermans, & van de Velde, 1994). How such dynamic testing can be done is shown in section 5.4.

In the area of geographic information, the Open GIS Consortium (Kottman, 1999) is developing formal software specifications, which can be seen as ontologies. One of their products is the Geography Markup Language (GML)[8], an XML encoding for the transport and storage of geographic information. It is used to describe data and relationships between components. Features are described as lists of properties. For example, a road might be defined to have a name, a surface-construction, a destination, and a centerline. These properties can be modeled in UML (Booch, Rumbaugh, & Jacobson, 1999) as attributes of the feature class (Figure 1). Besides missing information on what a road can be used for and by which agent, i.e., the lack of specified actions, UML is only semi-formal (Winter & Nittel, forthcoming) and static and can therefore not be used for testing.

[8] http://www.opengis.org/techno/specs/02-009/GML2-11.html

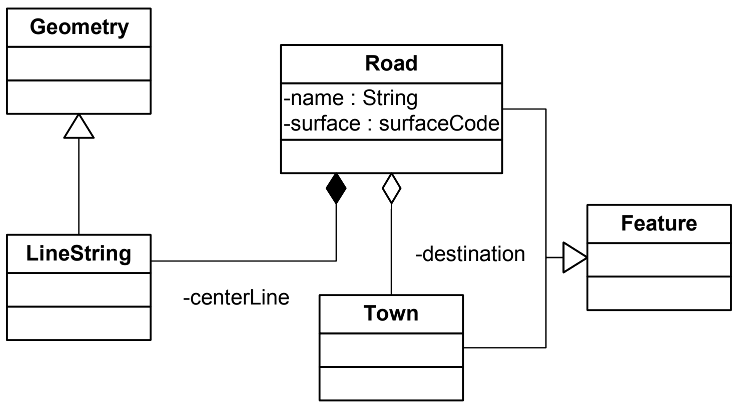

Figure 1: Representation of Road in UML (redrawn from http://www.opengis.org/techno/specs/02-009/GML2-11.html).

4.3 Formalization

Our method of ontology formalization uses algebraic specifications, which present a natural way of representing actions. Algebraic specifications have proven useful for specifying data abstractions for spatial and temporal domains (Car & Frank, 1995; Frank, 2000; Kuhn, 1996; Raubal, 2001a; Winter & Nittel, forthcoming). Data abstractions are based on *abstract data types*, which are representation-independent formal definitions of all operations of a data type (Guttag, Horowitz, & Musser, 1978). Algebraic specifications describe objects (or entities) in terms of their operations (or actions). They do not express what the objects are, but how they behave.

Algebraic specifications can also be used for testing at design time. Specifications for a software system can be tested before committing to build the system. This is exactly what is needed to fulfill our second requirement: tasks to be supported by a service need to be tested at design time, using the ontology of the data source. Algebraic specifications written in an executable programming language can be tested as a prototype of the service (Frank & Kuhn, 1995).

The tool chosen here is Hugs, a dialect of the functional language Haskell (Hudak, 2000), which includes the additional capabilities necessary for ontological engineering in our sense (Kuhn, 2001), namely the representation of

- *types:* ontologies draw distinctions between types of objects;
- *type classes:* types are characterized through collections of types, i.e., type classes, for which common functions are defined;
- *algebraic axioms:* the meaning of an action is expressed through algebraic axioms; a set of axioms is a set of rules that describes the effects of an operation in terms of other operations on the same or on simpler types;

Haskell is a purely functional language and provides typing as well as higher-order capabilities. In a nutshell, the necessary constructs for our case study formalization are the following:

- *algebraic data types*: `data Car = Car Name Position`
  (introducing a user-defined type),
- *type synonyms*: `type Position = [Int],type Name = String`
  (calling a pre-defined type differently),
- *type classes*: `class Named n where name :: n -> Name`
  (collecting types sharing the specified behavior),

- *instances*: `instance Named Car where name (Car n p) = n`
  (inheriting the class behavior to a type).

Haskell can be seen as an executable ontology development and testing environment (Kuhn & Raubal, 2003). One of its major strengths is *strong typing*. It means that every object has a particular type and the compiler checks that operations can only be applied to certain types. This results in strongly typed ontologies, which allow for avoiding wrong assumptions, i.e., specifying incompatible types for operations. The following section shows how its Hugs dialect is used for this purpose in practice.

5. ONTOLOGY SIMULATION FOR A CAR NAVIGATION SERVICE

The formalization method is now applied to the case study introduced in section 2. We first present the GDF data model and the relevant data types. Next, the necessary concepts from an experimental domain ontology are introduced. The GDF data types are then related to the behavior specified for the concepts in the domain ontology. The resulting specifications can be executed and therefore used for simulating tasks to be supported by the navigation service. The simulation explains the ontology mismatch that presumably occurred in the case study.

5.1 GDF[9] application model

The overall data model of GDF consists of feature themes, feature classes, features, complex features, attributes, and relations (ISO, 2001). All objects are conceptually divided into three different levels. *Level 0* is concerned with the topology and defines the basic graph-theoretical building blocks: nodes, edges, and faces. Simple features are represented on *level 1*, whereas complex features, i.e., aggregates of simple features, are represented on *level 2*. The basic building blocks of simple features are stored on level 0, while complex features are defined with simple or other complex features. Attributes are characteristics of features, which are independent of other features. Relations are meaningful links between two or more features. Each feature belongs to exactly one feature class and one feature theme.

The case study is concerned with the objects *Node* and *Edge* and the simple features *Road element* and *Ferry connection* representing the road network at levels 0 and 1 respectively. Figure 2 shows parts of the data model for the level 0. Figure 3 illustrates parts of the data model for levels 1 and 2 with respect to *Roads* and *Ferries*. Level 1 consists of the simple features *Road element* and *Ferry connection* and some of their attributes. These get aggregated to the complex features *Road* and *Ferry* at level 2. Definitions for the objects and features are given in Table 1.

[9] Let us reiterate that our choice of the GDF data model to demonstrate the need for ontology-driven simulation does not imply that GDF actually played a role in the reported accident.

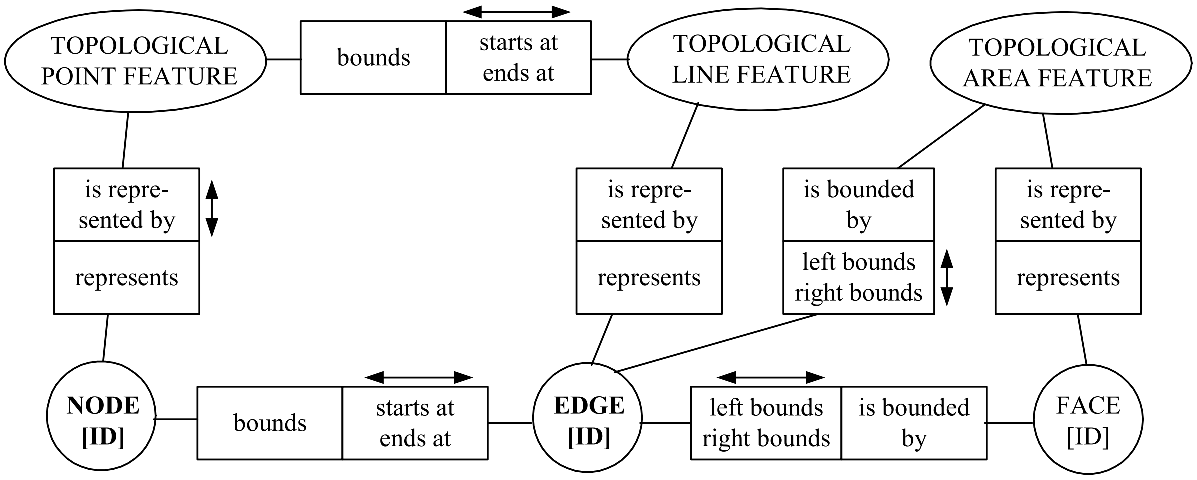

Figure 2: Extract of the GDF data model level 0 based on (ISO, 2001).

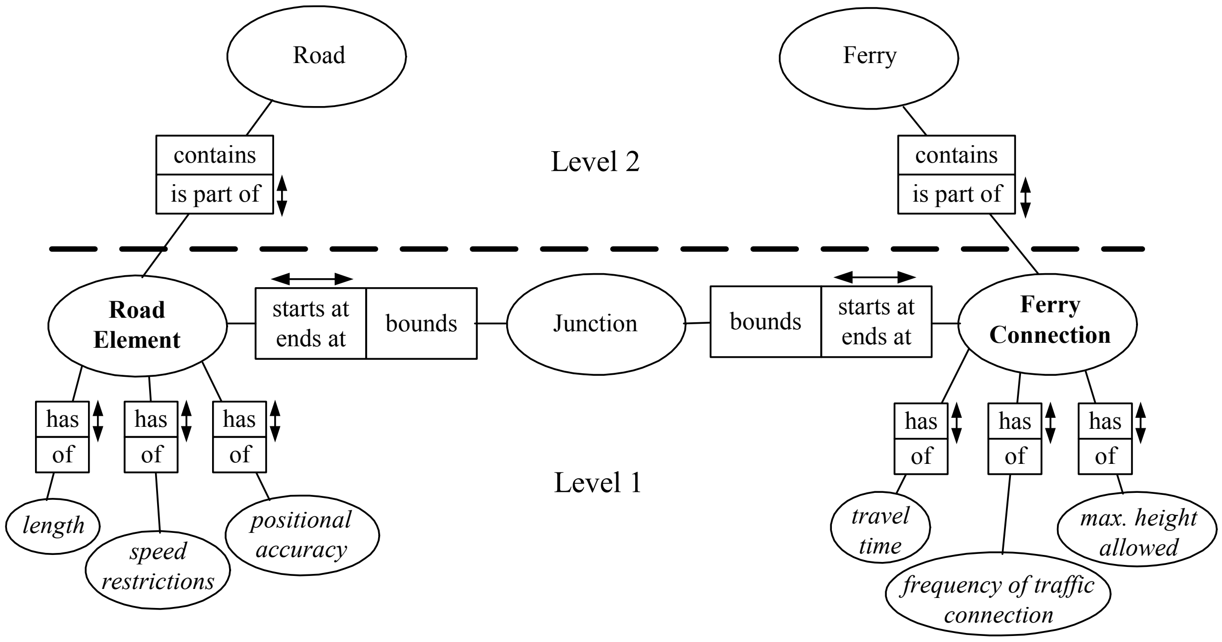

Figure 3: Part of the GDF data model level 1 and 2 for *Roads* and *Ferries* based on (ISO, 2001).

Object / Feature	**Definition**
Node	A zero-dimensional element that is a *topological junction* of two or more *Edges*, or an end point of an *Edge*.
Edge	A directed sequence of non-intersecting line segments with *Nodes* at each end.
Road element	A linear section of the earth, which is designed for or the result of vehicular movement. It serves as the smallest unit of the road network at Level 1 that is independent and having a *Junction* at each end.
Ferry connection	A vehicle transport facility between two fixed locations on the road network, which uses a prescribed mode of transport, for example, ship or train.
Road	A Level 2 *Feature* composed of one, many, or no *Road elements* and joining two *Intersections*. It serves as the smallest independent unit of a road network at Level 2.
Ferry	A set of *Ferry connections* that describe a passage of a particular ferry line.

Table 1: GDF definitions for *Node*, *Edge*, *Road element*, *Ferry connection*, *Road*, and *Ferry* (ISO, 2001).

The objects in this application model, which are relevant for the case study shall now be represented as Hugs data types, together with their attributes: nodes with a name, edges with two nodes, and road elements and ferry connections with an edge as attribute.

```
data Node = Node Name
data Edge = Edge Node Node
data RoadElement = RoadElement Edge
data FerryConnection = FerryConnection Edge
```

In addition, we define the necessary agents. These are cars (each car with a node as location attribute, the plural cars as lists of individual cars) and the car ferry, with the carried cars and a locating node as attributes:

```
data Car = Car Node
type Cars = [Car]
data CarFerry = CarFerry Cars Node
```

This completes the definition of the data types in our GDF application model. No axioms for these types have been defined yet. They will result from tying the data types to the domain ontology in section 5.3.

5.2 Domain ontology

A domain ontology serves as a semantic reference frame (Kuhn, 2003; Kuhn & Raubal, 2003) for the terms of data models in a domain. It represents the conceptualization underlying the use of the data model’s terms and provides a generic structure that can be employed for semantic reference in multiple applications. Here, we specify only the geospatial domain concepts to be used in the case study. They are part of a generic geospatial domain ontology currently under development.

Two important basic concepts are *naming* and *location*. Objects have names (or other identifiers) and can be compared for equality using these names. Two objects are considered equal (identical) if they have the same identifier. The type of a name depends on the type of the object, which gets expressed as a so-called type dependency (`object -> name`).

```
class Named object name | object -> name where
  name :: object -> name
instance (Eq name, Named object name) => Eq object where
  object1 == object2 = (name object1) == (name object2)
```

Objects also have locations (possibly of several different types):

```
class LocatedAt object location where
  location :: object -> location
```

The following four categories correspond to basic cognitive patterns called *image schemata*. It has been proposed that meaning involves image-schematic structures (Gärdenfors, 2000; Johnson, 1987). Image schemata fall between abstract propositional structures and concrete images. They are developed through bodily experiences and influence our reasoning through the recurrence of form and function. An image schema can be seen as a generic, maybe universal, and abstract structure that helps people establish a connection between different experiences that have this same recurring structure. Previous formalized representations of image schemata consist of algebraic definitions (Frank & Raubal, 1999; Kuhn & Frank, 1991; Rodríguez & Egenhofer, 1997) and predicates (Raubal, Egenhofer, Pfoser, & Tryfona, 1997). Without attempting a cognitive, psychological, or linguistic validation, we have found image schemata to provide excellent candidates for geospatial domain concepts. The case for this choice will be made elsewhere. Here it should suffice to see the core role of image-schematic operators in defining the semantics of actions in navigation.

Links connect two objects of the same type, with the object type being determined by the link type. The link schema introduces operations to ask for each end of the link and, given one end, for the other.

```
class Link link object | link -> object where
  from, to :: link -> object
  other :: Eq object => link -> object -> object
  other link object     | object == from link = to link
                        | object == to link = from link
                        | otherwise = error "not linked"
```

The *Path* schema allows us to express the behavior of moving: an object moves from one end of the path to the other end.

```
class Path path object where
  move :: path -> object -> object
```

The *Surface* schema is specified through behavior that one commonly associates with support. One can put objects on surfaces and take them off again and one can check whether a specific object is on a surface:

```
class Surface surface object where
  putOn    :: object -> surface -> surface
  takeOff  :: object -> surface -> surface
  isOn     :: object -> surface -> Bool
```

A collection of individual objects is represented through the *Collection* schema. Here, we specify the special case of collections of homogeneous type elements (with the collection type determining the element type and being determined by it). Collections can be empty, one can add or remove an element, one can ask whether a specific element is in a collection, and one can apply a modifying operation to all elements.

```
class Collection collection single | collection -> single,
                                             single -> collection where
      empty     :: collection
      addOne    :: single -> collection -> collection
      remove    :: single -> collection -> collection
      element   :: single -> collection -> Bool
      doToAll   :: (single -> single) -> collection -> collection
```

The concepts of links, paths, surfaces, and collections are sufficient for the purpose of anchoring our case study application ontology in a (otherwise rudimentary) domain ontology.

5.3 Application Ontology

We now specify the concepts, which the GDF data model uses in terms of the domain ontology given in the previous sub-section, producing an application ontology for GDF navigation. First, we assign nodes to the class of named objects. This expresses that each node can be asked for its name, using the name function:

```
    instance Named Node Name where name (Node n) = n
```

Then, edges are declared to be links between nodes. Note that, so far, edges had only been given two node attributes, without any semantics for these attributes. Now, edges are being subjected to the two functions that explain these as the “from” and “to” nodes:

```
    instance Link Edge Node where
      from (Edge node1 node2) = node1
      to (Edge node1 node2) = node2
```

Next, cars and car ferries are stated to be objects located at nodes. Again, each of them only had a node attribute in the data model—here it is explained what that attribute means:

```
    instance LocatedAt Car Node where
      location (Car node) = node
    instance LocatedAt CarFerry Node where
      location (CarFerry cars node) = node
```

Next, we say that car ferries act as surfaces, carrying a collection of cars:

```
    instance Surface CarFerry Car where
     putOn car (CarFerry cars node) = if (location car == node)
      then CarFerry (addOne car cars) node
      else error "Car and CarFerry are not at same location"
     takeOff car (CarFerry cars node) = CarFerry (remove car cars) node
     isOn car (CarFerry cars node) = element car cars
```

Car ferries can carry 0, 1, or many cars. This behavior is captured by the `Collection` class in the domain ontology. The data type `Cars` is, thus, declared to be an instance of this class. The behavior is the same as that of a list and the specification uses the Hugs List functions:

```
    instance Collection Cars Car where
      empty = []
      addOne car cars = car:cars
      remove car cars = delete car cars
      element car cars = elem car cars
      doToAll f cars = map f cars
```

CarFerries are also conveyances for the cars they carry. This means that they can transport the cars across ferry connections (note the difference between the behaviors of move and transport!). The concept of *Conveyance* blends the domain concepts *Path* and *Surface* into the combined transport behavior.

```
    class (Path path conveyance, Surface conveyance object) =>
    Conveyance conveyance path object where
    transport :: path -> conveyance -> object -> object
  instance Conveyance CarFerry FerryConnection Car where
    transport (FerryConnection edge) carFerry (Car node) =
        Car (other edge node)
```

Proceeding with the explanation of the GDF data types, we need to explain road elements and ferry connections (both from level 1). To begin with, they have the same behavior as an edge, linking two nodes in the navigation network:

```
instance Link RoadElement Node where
  from (RoadElement edge) = from edge
  to (RoadElement edge) = to edge
instance Link FerryConnection Node where
  from (FerryConnection edge) = from edge
  to (FerryConnection edge) = to edge
```

In addition to the link behavior, road elements and ferry connections act as paths, but of two different kinds: Road elements are paths affording a car to move,

```
instance Path RoadElement Car where
  move (RoadElement edge) (Car node) = Car (other edge node)
```

while ferry connections are paths for car ferries:

```
instance Path FerryConnection CarFerry where
  move (FerryConnection edge) (CarFerry cars node) =
      CarFerry (doToAll (transport (FerryConnection edge)
      (CarFerry cars node)) cars) (other edge node)
```

Note that the move operation for ferries needs to make sure that all cars are transported with the ferry (which it does through the `doToAll` function from the `Collection` class).

This completes the formalized explanation of the navigation concepts in our case study. All terms in the GDF data model have now been committed to the domain ontology of section 5.2 and thereby given meaning. This lays the ground for simulating activities such as driving on roads or crossing a river on a ferry.

5.4 Simulation and Hypothesis Testing

The application ontology in the previous sub-section provides more than a static definition of navigation concepts. Its axioms are *executable* specifications of the actions occurring in navigation tasks of cars and ferries. Thus, they can be used to simulate these tasks, as we will now demonstrate.

Let us first introduce some names used in the simulations. Two nodes, called "start" and "end", are linked by "theEdge" and an individual car, "theCar", is located at the start node:

```
start = Node "start"
end = Node "end"
theEdge = Edge start end
theCar = Car start
```

The edge can serve, alternatively, as a road element or a ferry connection:

```
theRoadElement = RoadElement theEdge
theFerryConnection = FerryConnection theEdge
```

Finally, various states of the car ferry (empty, loaded, and moved across the ferry connection) are defined:

```
emptyCarFerry = CarFerry empty start
loadedCarFerry = putOn theCar emptyCarFerry
```

```
movedCarFerry = move theFerryConnection loadedCarFerry
```

It is now straightforward to simulate the task of moving a car over a road element. We formulate this as a test, t1, whether the car is located at the end of the road element after moving over it. The value of t1 has to evaluate to "True":

```
t1 = location (move theRoadElement theCar) == end
```

Similarly, t2 tests whether moving the loaded ferry over the ferry connection gets it to the end node:

```
t2 = location (move theFerryConnection loadedCarFerry) == end
```

And t3 tests the more interesting question whether the car is at the end of the ferry connection after being transported on the ferry:

```
t3 = location (transport theFerryConnection loadedCarFerry
        theCar) == end
```

It turns out that, with the behavior specified in the application ontology, all these tests evaluate to "True." However, the simulation of a car moving over a ferry connection (rather than a road element) fails. The following statement is rejected by the Hugs interpreter, because ferry connections have been defined as paths for ferries, not for cars:

```
* move theFerryConnection theCar
```

Similarly, it is not possible to move the car along the edge:

```
* move theEdge theCar
```

The results of these simulations confirm the behavior expected from cars, roads, and car ferries. So, what might have gone wrong in the accident? A likely possibility is that the navigation system used correct navigation data, but confused level 0 and level 1. To test this hypothesis, we need to define just one additional axiom, expressing the interpretation of level 0 edges as paths for cars (no matter whether they represent road elements or ferry connections):

```
instance Path Edge Car where
  move edge (Car node) = Car (other edge node)
```

With this axiom added to the application ontology, a test whether the car can move along the edge evaluates to "True":

```
t = location (move theEdge theCar) == end
```

This interpretation applied by the navigation system is suitable for calculating routes, but not refined enough for giving instructions to a driver. While we do not know what has caused the accident on which our case study is based, the simulation demonstrates how such a hypothesis about an ontological mismatch between the data and the navigation system can be tested.

6. CONCLUSIONS AND FUTURE WORK

We presented the idea of ontology-based task simulation as a method for evaluating the usability of a data source for an information service supporting human activities. We identified two key requirements for ontologies to support the simulation of tasks: *actions must be represented and testable in the ontologies*. Current ontologies rarely include actions and are mainly formalized in a static way. We propose dynamic ontologies, consisting of a domain ontology that includes processes, and application ontologies explaining concepts in terms of the interaction between actions and entities. The application ontologies contain executable specifications of actions. The functional language Haskell was chosen as a tool for ontology design because it offers the required capabilities.

The method was demonstrated through a real-world scenario for a navigation service—instructions for crossing a river by car. We first specified the relevant data types from a navigation model. Next, a domain ontology consisting of image-schematic concepts was introduced as a semantic reference frame. The data types were then related to the domain ontology in order to assign meaning to them. The resulting executable specifications of the actions occurring in navigation tasks were used for the task simulation[10]. This simulation demonstrated how an ontological mismatch between the data source and the navigation system can be discovered. Such simulations can therefore help a service provider to determine whether a data source can be employed by a service. In cases where it is intended that a service uses more than one data source with different semantics, each of the data sources needs to be wrapped separately into the domain ontology. This again results in sets of executable specifications, which can be used for simulation.

Although the workload for a service provider to perform such simulations can be high, it seems to be the only comprehensive and reliable method to check whether a data source can be used by a service. In any case, one needs to specify the appropriate actions for the data set in order to evaluate whether they fit for the tasks to be supported by the service. Even then, having a designer acquire the data first and then look through the data set to see what could go wrong with it when used for a particular purpose is too expensive, error-prone and will most likely omit some important and special cases.

The work presented here suggests many questions and directions for future research:

1. The simulation of tasks in various (geospatial) information services requires the establishment of an extensive geospatial domain ontology. Our idea of using image-schematic concepts as elements of a domain ontology seems plausible and useful. It is currently being further explored.
2. The specifications of actions in the domain ontology can be seen as defining mediator functions for data. Thus, this special form of ontologies would deliver testable mediator specifications at the same time. Tests with this approach to data wrapping and mediating have been performed already [Gerding, forthcoming #873] and are currently being expanded. It appears that the tight connection between the software-engineering notion of mediating between information resources and the dynamic aspects of ontologies has not yet been exploited in theory or practice.
3. The question of whether the method of ontology-based simulation and the tool can be efficiently used for evaluating data usability in more complex geospatial applications and scenarios can only be answered through more extensive case studies. We are currently undertaking several such studies in the areas of environmental planning and emergency management (http://musil.uni-muenster.de). These studies include the generalization of the approach to service chains and semantic translation.
4. As a long-term goal, ontology-based simulation could be used as part of the semantic web to automatically evaluate the usability of different data sets for a given web service. For example, car drivers could log on to web-based car navigation services via a wireless connection and depending on the context (e.g., driver's task and car's position) relevant data sources for the service could be found in real time and ranked by their suitability.
5. The idea of simulating tasks is also part of the bigger picture of testable ontologies. Shifting the focus from user needs to designers, the issue whether and how ontologies can be tested at design time is of growing importance.

[10] The complete Hugs code for this paper is available (possibly in updated form) from the *Experiments* section of http://musil.uni-muenster.de. Hugs interpreters can be downloaded freely from http://www.haskell.org.

7. References

Barsalou, L., Sloman, S., & Chaigneau, S. (forthcoming). The HIPE Theory of Function. In L. Carlson & E. van der Zee (Eds.), *Representing functional features for language and space: Insights from perception, categorization and development*. New York: Oxford University Press.

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *The Unified Modeling Language Reference Manual*. Reading: Addison-Wesley.

Camara, G., Monteiro, A., Paiva, J., & de Souza, R. (2000). Action-Driven Ontologies of the Geographical Space: Beyond the Field-Object Debate. In A. Caschetta (Ed.), *GIScience 2000* (pp. 52-54). Savannah, Georgia, USA: University of California Regents.

Car, A., & Frank, A. (1995). Formalization of Conceptual Models for GIS using Gofer. *Computers, Environment, and Urban Systems, 19*(2), 89-98.

Chandrasekaran, B., Josephson, J., & Benjamins, R. (1999). What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems*(1), 20-26.

Coleman, M. (2001). Turn-by-turn Car Navigation. *GeoInformatics, 4,* 20-21.

Degen, W., Heller, B., & Herre, H. (2002). GOL: A Framework for Building and Representing Ontologies. In B. Heller & H. Herre & B. Smith (Eds.), *Ontological Spring - A Reader in Formal and Applied Ontology* (pp. 182-203). Leipzig: IFOMIS.

Fellbaum, C. (1998). A Semantic Network of English Verbs. In C. Fellbaum (Ed.), *WordNet - An Electronic Lexical Database* (pp. 69-104): The MIT Press.

Fonseca, F., Egenhofer, M., Davis, C., & Câmara, G. (2002). Semantic Granularity in Ontology-Driven Geographic Information Systems. *Annals of Mathematics and Artificial Intelligence, 36*(1-2), 121-151.

Frank, A. (2000). Spatial Communication with Maps: Defining the Correctness of Maps Using a Multi-Agent Simulation. In C. Freksa & W. Brauer & C. Habel & K. Wender (Eds.), *Spatial Cognition II - Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications* (Vol. 1849, pp. 80-99). Berlin, Heidelberg, Germany: Springer.

Frank, A. (2003). Pragmatic Information Content: How to Measure the Information in a Route Description. In M. Duckham & M. Goodchild & M. Worboys (Eds.), *Foundations of Geographic Information Science* (pp. 47-68). London, U.K.: Taylor & Francis.

Frank, A. (forthcoming). Ontology for Spatio-Temporal Databases. In T. Sellis (Ed.), *Spatiotemporal Databases: The Chorochronos Approach*: Springer.

Frank, A., Bittner, S., & Raubal, M. (2001). Spatial and Cognitive Simulation with Multi-agent Systems. In D. Montello (Ed.), *Spatial Information Theory - Foundations of Geographic Information Science, Proceedings of COSIT 2001, Morro Bay, CA, USA, September 2001* (Vol. 2205, pp. 124-139). Berlin, Heidelberg, New York: Springer.

Frank, A., & Kuhn, W. (1995). Specifying Open GIS with Functional Languages. In M. Egenhofer & J. Herring (Eds.), *Advances in Spatial Databases (SSD'95)* (Vol. 951, pp. 184-195). Portland, ME, USA: Springer.

Frank, A., & Kuhn, W. (1999). A Specification Language For Interoperable GIS. In M. Goodchild & M. Egenhofer & R. Fegeas & C. Kottman (Eds.), *Interoperating Geographic Information Systems* (pp. 123-132). Boston / Dordrecht / London: Kluwer Academic Publishers.

Frank, A., & Raubal, M. (1999). Formal Specifications of Image Schemata - A Step to Interoperability in Geographic Information Systems. *Spatial Cognition and Computation, 1*(1), 67-101.

Gärdenfors, P. (2000). *Conceptual Spaces - The Geometry of Thought*. Cambridge, MA: Bradford Books, MIT Press.

Genesereth, M., & Fikes, R. (1992). *Knowledge Interchange Format*. Stanford: Computer Science Department, Stanford University.

Gibson, J. (1979). *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin Company.

Gimblett, H., Durnota, D., & Itami, R. (1997). Some Practical Issues in Designing and Calibrating Artificial Human-Recreator Agents in GIS-based Simulated Worlds. *Complex International Journal - Workshop on Comparing Reactive (ALife-ish) and Intentional Agents, 3*.

Goguen, J., Thatcher, J., & Wagner, E. (1978). An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types. In R. Yeh (Ed.), *Current Trends in Programming Methodology* (pp. 80-149). Englewood Cliffs, NJ: Prentice-Hall.

Gruber, T. (1992). *Ontolingua: A Mechanism to Support Portable Ontologies*. Stanford: Knowledge Systems Laboratory, Stanford University.

Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition, 5*(2), 199-220.

Guarino, N. (1998). Formal Ontology and Information Systems. In N. Guarino (Ed.), *Formal Ontology in Information Systems* (pp. 3-15). Trento, Italy: IOS Press.

Guarino, N., & Welty, C. (2002). Evaluating Ontological Decisions with OntoClean. *Communications of the ACM, 45*(2), 61-65.

Guttag, J., Horowitz, E., & Musser, D. (1978). The Design of Data Type Specifications. In R. Yeh (Ed.), *Current Trends in Programming Methodology* (Vol. 4 - Data Structuring, pp. 60-79). Englewood Cliffs, NJ: Prentice-Hall.

Hamburger-Abendblatt. (1998, December 28, 1998). Auto in Havel gestürzt - Vom Computer ins Wasser gelotst*,* pp. 15.

Hudak, P. (2000). *The Haskell School of Expression: Learning Functional Programming through Multimedia*. New York: Cambridge University Press.

IEEE. (2003). Standard Upper Ontology (SUO) Working Group. http://suo.ieee.org/

ISO. (2001). *GDF - Geographic Data Files - Version 4.0* (Draft International Standard ISO/CD 2001-02-14): ISO/TC 204 N 34.

Johnson, M. (1987). *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason*. Chicago: The University of Chicago Press.

Konijn, M. (1998). Geographic Data Files (GDF). http://www.ertico.com/links/gdf/gdf.htm

Kottman, C. (1999). The Open GIS Consortium and progress toward interoperability in GIS. In M. Goodchild & M. Egenhofer & R. Fegeas & C. Kottman (Eds.), *Interoperating Geographic Information Systems* (pp. 39-54). Norwell, MA: Kluwer.

Kuhn, W. (1996). *Semantics of Geographic Information*. Vienna: Department of Geoinformation.

Kuhn, W. (2001). Ontologies in support of activities in geographical space. *International Journal of Geographical Information Science, 15*(7), 613-631.

Kuhn, W. (2003). Semantic Reference Systems. *International Journal of Geographical Information Science, 17*(5), 405-409.

Kuhn, W., & Frank, A. (1991). A Formalization of Metaphors and Image-Schemas in User Interfaces. In D. Mark & A. Frank (Eds.), *Cognitive and Linguistic Aspects of Geographic Space* (NATO ASI Series ed., pp. 419-434). Dordrecht, Boston, London: Kluwer Academic Publishers.

Kuhn, W., & Raubal, M. (2003). Implementing Semantic Reference Systems. In M. Gould & R. Laurini & S. Coulondre (Eds.), *AGILE 2003 - 6th AGILE Conference on Geographic Information Science* (pp. 63-72). Lyon, France: Presses Polytechniques et Universitaires Romandes.

McKie, R. (1998). BMW's computer makes driver turn into drinker. *The Observer, December 27, 1998*.

Mota, L., Bento, J., & Botelho, L. (2002). Ontology definition languages for Multi-Agent Systems: the Geographical Information Ontology case study, *Workshop on Ontologies in Agent Systems at 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Bologna, Italy.

Neumann, P. (1999). *Car computer directs couple into river, The Risks Digest, Forum on Risks to the Public in Computers and Related Systems, ACM Committee on Computers and Public Policy (Volume 20: Issue 14, Sunday 3 January 1999) - http://catless.ncl.ac.uk/Risks/20.14.html#subj1.1.*

Raubal, M. (2001a). Human wayfinding in unfamiliar buildings: a simulation with a cognizing agent. *Cognitive Processing*(2-3), 363-388.

Raubal, M. (2001b). Ontology and epistemology for agent-based wayfinding simulation. *International Journal of Geographical Information Science, 15*(7), 653-665.

Raubal, M., Egenhofer, M., Pfoser, D., & Tryfona, N. (1997). Structuring Space with Image Schemata: Wayfinding in Airports as a Case Study. In S. Hirtle & A. Frank (Eds.), *Spatial Information Theory—A Theoretical Basis for GIS, International Conference COSIT '97, Laurel Highlands, PA* (Vol. 1329, pp. 85-102). Berlin: Springer.

Riedemann, C., & Timm, C. (2003). Services for Data Integration. *Data Science Journal, 2*(26), 90-99.

Rodríguez, A., & Egenhofer, M. (1997). Image-Schemata-Based Spatial Inferences: The Container-Surface Algebra. In S. Hirtle & A. Frank (Eds.), *Spatial Information Theory—A Theoretical Basis for GIS, International Conference COSIT '97, Laurel Highlands, PA* (Vol. 1329, pp. 35-52). Berlin: Springer.

Schreiber, A., Wielinga, B., de Hoog, R., Akkermans, J., & van de Velde, W. (1994). CommonKADS: A comprehensive methodology for KBS development. *IEEE Expert, 9*, 28-37.

Smith, B. (2001). Objects and Their Environments: From Aristotle to Ecological Ontology. In A. Frank & J. Raper & J.-P. Cheylan (Eds.), *Life and Motion of Socio-economic Units* (Vol. 8, pp. 79-97). London: Taylor & Francis.

Smith, B. (2002). Ontology. In B. Heller & H. Herre & B. Smith (Eds.), *Ontological Spring - A Reader in Formal and Applied Ontology* (pp. 1-14). Leipzig: IFOMIS.

Timpf, S., Volta, G., Pollock, D., & Egenhofer, M. (1992). A Conceptual Model of Wayfinding Using Multiple Levels of Abstraction. In A. Frank & I. Campari & U. Formentini (Eds.), *GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning* (Vol. 639, pp. 348-367). Pisa, Italy: Springer.

Welty, C., & Smith, B. (2001). *Formal Ontology in Information Systems - Collected Papers from the Second International Conference*. New York: Association for Computing Machinery, Inc. (ACM).

White, M. (1991). Car Navigation Systems. In D. Maguire & M. Goodchild & D. Rhind (Eds.), *Geographical Information Systems: Principles and Applications* (Vol. 2, pp. 115-125). New York: Longman Scientific and Technical.

Winter, S., & Nittel, S. (forthcoming). Formal Information Modeling for Standardisation in the Spatial Domain. *International Journal of Geographical Information Science*.