A Tour of BeAware
A Situation Awareness Framework for Control Centers[☆]

Norbert Baumgartner[a], Stefan Mitsch[b,*], Andreas Müller[c], Werner Retschitzegger[c], Andrea Salfinger[c], Wieland Schwinger[c]

[a] *team Communication Technology Management GmbH, Linke Wienzeile 4, 1060 Vienna, Austria*
[b] *Carnegie Mellon University, Computer Science Department, 5000 Forbes Ave, Pittsburgh, PA 15213, USA*
[c] *Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria*

Abstract

Large control centers, as needed in road traffic, typically manage highly dynamic environments. They process vast amounts of information from heterogeneous data sources about a large number of real-world objects, which are anchored in time and space. In such systems, human operators are vulnerable to information overload and, thus, may fail to be aware of the *overall meaning* of available information and *its implications*. With *BeAware*, we propose a software framework that supports the development of *situation awareness* applications for control centers. The contribution of this paper is twofold: First, we integrate existing ontologies with spatio-temporal reasoning concepts, focusing on extensibility. We introduce meta-modeling concepts that allow us to assess and project situations and actions using semantic web technology. Second, we compare the runtime performance of the situation comprehension capabilities of a generic, ontology-driven implementation and a domain-specific relational-database-backed implementation, and discuss the strengths and shortcomings of each approach.

Keywords: situation awareness, road traffic control, spatio-temporal reasoning, ontologies, framework

1. Introduction

Large control centers—as needed in domains such as road traffic management or air traffic control—manage geographically large environments that are only partially observable and highly dynamic. Such systems provide a vast and steadily increasing amount of information from heterogeneous sources about a large number of real-world objects, which are anchored in time and space. On the basis of this information, human operators must initiate appropriate actions to ensure smooth functioning of the environment to be controlled.

A common problem in control centers is that human operators are at permanent risk of information overload—or as stated by Naisbitt [1] "we are drowning in information but starving for knowledge". This information overload entails a lack of awareness of the overall meaning and implications of the available information, not least because of insufficient system support. This endangers an operator's ability to respond to critical situations with potentially serious consequences.

To illustrate the challenges we intend to tackle, we briefly introduce road traffic as our demonstration domain. Let us suppose that an accident on a highway causes a traffic jam. A human operator would typically react to this situation by opening the emergency lane, which bypasses the accident and prevents the traffic jam from growing further. Unless the operator consults other sources, he/she might not be aware of facts that further aggravate the situation: e. g., a nearby football game is about to end and the spectators cannot use the usual route to avoid the traffic jam due to road maintenance.

[☆] This work has been funded by the Austrian BMVIT under grants FFG FIT-IT 829598, FFG BRIDGE 838526 and 832160, by AR18/2013 and UA07/2013 WTZ/ÖAD, and by ERC PIOF-GA-2012-328378. * Corresp. author. *Email addresses:* norbert.baumgartner@te-am.net (Norbert Baumgartner), smitsch@cs.cmu.edu (Stefan Mitsch), andreas.mueller@cis.jku.at (Andreas Müller), werner.retschitzegger@cis.jku.at (Werner Retschitzegger), andrea.salfinger@cis.jku.at (Andrea Salfinger), wieland.schwinger@jku.at (Wieland Schwinger)

Practice has shown that such critical situations are encountered frequently. On Austrian highways, which span around 2,200 km of roads, approximately 5,000 traffic incidents are recorded every day. Incident data is both automatically generated (e. g., traffic flow detectors) and manually reported, and thus highly dynamic and vague. This entails that *qualitative knowledge representation* approaches, such as ontologies, are often better suited than quantitative ones [2, 3]. Existing ontologies (e. g., [4, 5]) focus on specific aspects of situation awareness, and hence, their concepts must be integrated. Even then, current traffic management systems scarcely support human operators in such scenarios [6]. In summary, the problems are: (i) high effort of keeping track of evolution in multiple sources; (ii) failure to recognize critical situations in time; (iii) inability to anticipate emerging situations; (iv), risk of initiating incorrect actions.

With the software framework *BeAware*, we propose technical concepts that support the development of situation awareness (SAW) applications for control centers. We aim for on an extended SAW model including *resolution* [7] by proposing actions. Although it is the ultimate goal of such applications to increase situation awareness of human operators according to Endsley's definition [8]—which comprises the *perception* of the elements in the environment, the *comprehension* of the current situation, and the *projection* of future states—it must be emphasized that BeAware takes only a first step in this direction with a software framework for SAW applications and does not yet manage situation control.

Summary of contributions. The contribution of this paper is twofold: First, we propose to integrate existing ontologies with spatio-temporal reasoning concepts, with a focus on extensibility with domain-specific knowledge. The resulting ontology consists of multiple modules and introduces meta-modeling concepts, which allow situation projection and action assessment using semantic web technology. We illustrate the application of the framework with examples from road traffic management. Second, we present a runtime performance comparison regarding the situation comprehension capabilities of a generic, ontology-driven implementation and a domain-specific relational-database implementation, and discuss the strengths and shortcomings of each approach. The real-world traffic data for this comparison was obtained while applying BeAware in the course of our research project *Collaborative Situation Awareness* (CSI[1]).

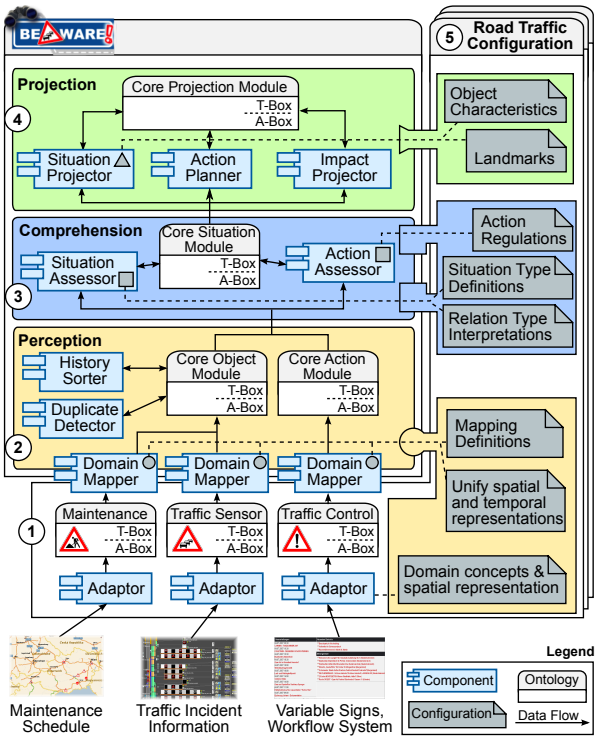


Figure 1: Key components and extension points

Structure of the paper. The next section gives an overview of the BeAware framework. Sections 3–5 describe challenges, related work, ontology modules, and components for each of Endsley's levels of SAW. Section 6 describes the prototypical implementations; their performance is evaluated in Section 7. Section 8 concludes with future work.

2. BeAware at a Glance

This section gives an overview of the BeAware framework and the qualitative knowledge aligned to the three levels of situation awareness according to Endsley [8]. It introduces meta-modeling concepts that are central modeling elements throughout the paper. Note, that although our domain is traffic control, we primarily target decisions of human operators in control centers, not yet real-time control. The underlying ontology of BeAware is split into modules (each in a dedicated file) to support concurrent development.

[1] `csi.situation-awareness.net`

Table 1: Employed DL constructs (definitions from [9]) and their OWL and UML notation

Family	Syntax	Name	OWL construct	UML construct
			T-Box Axioms	
$\mathcal{AL}$	C	Atomic concept	owl:Class	
	$\top, \bot$	Top, bottom concept	owl:Thing, owl:Nothing	
$\mathcal{ALC}$	$\neg$ C	Concept negation	owl:complementOf	
	C$\equiv$ D	Concept equivalence	owl:equivalentClass	Class note (constraint {equivalent})
	C $\sqcap$ D	Concept intersection	owl:intersectionOf (and)	
	C $\sqcup$ D	Concept union	owl:unionOf (or)	
	R	Atomic role	owl:objectProperty	Association
	$\exists$R.C	Existential quantifier	owl:someValuesFrom	Class note with OCL constraint
	$\forall$R.C	Universal quantifier	owl:allValuesFrom	Class note with OCL constraint
$\mathcal{I}$	R^-	Inverse role	owl:inverseOf	Association note (constraint {inverse})
$\mathcal{O}$	$\exists$R.{a}	Exist. nominal quantifier	owl:hasValue	Note on class with OCL constraint
$\mathcal{Q}$	$\leq$ n.R.C	Qualified number restriction	owl:maxCardinality	Multiplicity
	C $\sqsubseteq$ D	Concept inclusion	Generalization	Generalization
$\mathcal{R}$	R $\circ$ S $\sqsubseteq$ T	Role composition	owl:ObjectPropertyChain	Derived association
			A-Box Axioms	
	C(a)	Concept assertion	rdfs:type	Object
	R(a,b)	Role assertion		Object association

2.1. Architecture and Ontology Modules

As depicted in Figure 1, the components of the perception subsystem map domain information—provided by adaptors (1)—to core ontology modules (2). They also eliminate duplicate objects and actions (duplicate detector), and ensure a uniform time scale (history sorter). The framework furthermore assumes that an application provides a uniform spatial representation (e. g., a road network graph); it is the responsibility of the adaptors to map the internal representation of a data source to the uniform representation and of the domain mappers to ground that uniform representation in the core ontology. Further details can be found in [10, 11]. Based on their results, the comprehension system's *situation assessor* (3) detects current situations, which are the basis for (i) the *action assessor* to suggest appropriate actions, and (ii) the *situation projector* (4) to project future situations. These projections drive the *action planner* to search pro-actively for actions mitigating undesired situations. The potential consequences of taking actions are then highlighted by the *impact projector*. Projection is closed by a feedback loop to the situation projector, which allows for projections to be computed recursively. The framework is configured (5) in terms of ontology extensions and rules.

In the architecture, we especially aimed for an extended SAW model including *resolution* [7] (i. e., awareness about actions). Resolution stretches over Endsley's levels of SAW to incorporate the view of Boyd's OODA loop [12]: *observe* (cf. perception), *orient* (cf. comprehension: action assessment), *decide* (projection: action planning and projection), and *act* (human operator initiates action).

The ontology modules provided by BeAware are expressed in description logics (DL) and for presentation as UML class diagrams. These modules are extensions of our previous road traffic management ontology [6] with respect to actions, extensibility with spatial and temporal modules, meta-modeling, situation projection, and formal definition in the description logic $\mathcal{SROIQ}$ (i. e., OWL 2). We address extension points that allow developers to configure BeAware with a vocabulary familiar to human operators in their application domain. Such extensions are integrated by inheriting from or referring to concepts of the ontology of BeAware, and become active when loaded in the same triple store or reasoner. As usual, we distinguish between *concepts* C and *roles* R to describe the structure of the world in a terminological box (T-Box), and assertions thereof ($\mathsf{C(a)}$ and $\mathsf{R(a,b)}$) to describe individual objects and their relationships in an assertional box (A-Box). In subsequent sections, we use the syntax of Table 1, and give extension examples from road traffic control.

2.2. DL-safe Meta-Modeling

Many concepts in spatio-temporal reasoning and action formalizations are expressed between classes (e. g., composition tables of Allen's interval algebra [13], conceptual neighborhood between relations [14], and applicability and effects of actions

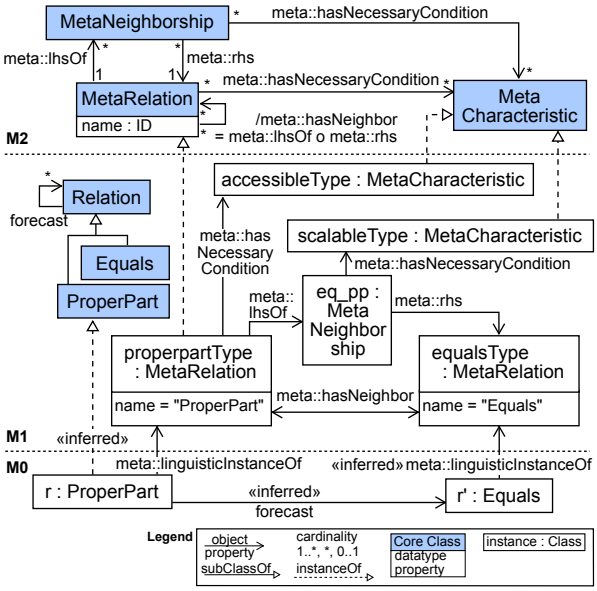


Figure 2: Meta-modeling example.

[15]). In order to allow statements about all instances of a class, we introduce a *meta-modeling* approach, which is used throughout this paper.

A common practice in conceptual modeling that is also applied in OWL is to define the general structure of concepts as classes (T-Box, model level M1) and the state of the world in terms of individuals (instances of classes) in an ontology (A-Box, instance level M0). OWL Full allows making statements about classes, but—due to its undecidability—no efficient reasoner exists [16]. We therefore include a meta-modeling approach expressed in OWL DL using DL-safe rules, which enables statements similar to the contextual semantics proposed in [17] (which are decidable and included in OWL 2 [18]). Figure 2 exemplifies this meta-modeling approach using relation types.

An additional meta-model level M2 is added conceptually on top of M0 and M1 (but in terms of implementation comprises statements in both T- and A-Box). M2 provides concepts for making statements about classes. In Figure 2, MetaRelation (i. e., the class of all relation types) has neighboring relation types and necessary conditions that must be fulfilled, cf. MetaCharacteristic (i. e., the class of all types of characteristics). M1 consists of classes (e. g., ProperPart); in order to relate these classes to each other using object properties, each is accompanied by an instance of an M2 class (e. g., properPartType, which is an instance of MetaRelation). Associations between M0 individuals and M1 classes are inferred from asserted associations to instances of M2 classes, as shown for ProperPart in Example 1.

Example 1. *ProperPart is the class of all individuals that are in a linguisticInstanceOf association with a MetaRelation named "ProperPart", cf.* (1).

$$\begin{aligned} ProperPart \sqsubseteq\ & \exists linguisticInstanceOf. \\ & (MetaRelation \sqcap \exists name.\{\text{“}ProperPart\text{”}\}) \end{aligned} \tag{1}$$

Associations between classes are specified on M1 (e. g., properPartType has equalsType as neighbor). Rules, as in Example 2, derive knowledge on M0.

Example 2. *Instance r' of equalsType is forecast as successor of r, because r is in a meta::linguisticInstanceOf association with properPartType, which equalsType is known to be a neighbor of, cf.* (2).

$$\begin{aligned} & Relation(r) \wedge linguisticInstanceOf(r,t) \\ & \wedge hasNeighbor(t,t') \supset linguistic- \\ & InstanceOf(r',t') \wedge forecast(r,r') \end{aligned} \tag{2}$$

In the following sections, we take a closer look at the conceptual details and the configurability of components and extensibility of ontology modules.

3. The Perception Layer

The main tasks of the perception layer are (i) to *integrate information from heterogeneous sources*, and (ii) to *track evolution* within those sources.

Integration of heterogeneous data sources. Large-scale control centers typically need to integrate information from multiple data sources, which are often characterized by heterogeneous formats and may contain identical, incomplete, and often even contradictory information [19]. Thus, not only must structural heterogeneities be resolved at the schema level, the data itself must be fused into a consistent form at the instance level. In this context the challenges are (i) to provide domain-independent concepts accompanied by *duplicate detection and data fusion components*, and (ii) to define extension points for domain-dependent information about *objects* (e. g., of type TrafficJam), their intrinsic *attributes* (e. g., length), and *actions* (e. g., of type OpenEmergencyLane). It is especially important to represent actions as first-class citizens in the ontology, since actions that coincide with certain objects may characterize situations.

Table 2: Summary of perception-level-related work

Criterion	SOUPA [20]	CONON [21]	CAWE [22]	CA workflows [23]	CPDL [24]	uWDL [25]	Planning ontology [26]	PLANET(-COA) [27]	SNAP [28]	Geospatial [29, 30]	DogmatiX [31]	HumMer [32, 33]
Structural heterogeneity												
Extensible object types		✓						✓				
Spatio-temporal attributes	✓	✓			✓			✓		✓		
Extensible attributes	✓		✓	✓	✓	✓	✓	✓	✓			
Extensible action types	✓	✓	✓	✓	✓	✓	✓	✓				
Actions as objects	✓											
Detect duplicate entries												
Qualitative rule-based												
Similarity-based											✓	✓
Spatio-temporal										✓		
Fuse duplicate entries												
Fusion strategies												✓
Extensible fusion strategies												✓
Object evolution												
Extensible event types	✓		✓	✓	✓	✓	✓	✓	✓			
Event effect modeling			✓	✓	✓		✓	✓	✓			
Reconstructs object histories												

Not applicable

Tracking object evolution. Real-world objects undergo evolution in the form of observable events, which affect their attributes (e. g., a traffic jam may grow when an accident occurs, shrink when the accident has been cleared, and finally dissolve). These *events* (e. g., accident occurs) and their effects on attributes (e. g., grow) are observed in a discrete manner, and may therefore be recorded across data sources in varying sequences and granularities. For example, traffic jam detectors may report accurate values every five minutes, whereas drivers may report incidents infrequently and less accurately, but also at sensor-less locations. Another challenge, therefore, is to develop components to reconstruct object histories and appropriate valid times.

3.1. Related Work

Table 2 summarizes related work on the perception level, w.r.t. the challenges listed above. Concerning domain-independent abstractions of objects and attributes, research on context awareness, such as SOUPA [20] and CONON [21] introduce upper ontologies for context awareness, and thus already clarify core terms from an ontological viewpoint, which are re-used in our ontology (cf. [34] and [35] for surveys on context models). In BeAware, we complement these core concepts with extension points for domain knowledge.

However, context awareness research has not yet particularly focused on representing actions. We therefore investigate actions from a workflow perspective: Current workflow modeling languages and standards either provide only limited context information support (e. g., CAWE [22], or CA workflows [23]), or do not adequately decouple workflow from context modeling [36], often using proprietary languages such as uWDL [25], or CPDL [24]. Moreover, workflow instances themselves have not yet been recognized as beneficial context entities, although meta-models and ontologies for describing workflows exist [37]. Interesting ideas for action ontologies can also be found in the areas of automated planning and recommender systems. Examples of the former are PLANET(-COA) [27], the planning ontology [26], and SOUPA [4], which proposes the concepts "actor" and "time" to describe the point in time at which an action has been performed and by whom. In the field of recommender systems (e. g., SNAP [28]) the concept "causes", which indicates action effects, has been proposed. Considering the wide variety of available workflow (and other action) specification approaches, BeAware again concentrates on providing necessary extension points, and integration of objects with actions.

Concerning duplicate detection and data fusion, a recent survey [30] focuses on semantic similarity of geospatial data, thereby emphasizing the importance of appropriate similarity measures for different spatial representation models. However, many existing similarity measures are restricted to topological and distance aspects of relations between lines and regions, and do not combine these measures to duplicate detection rules in component implementations. With respect to duplicate detection and data fusion, relevant approaches can be found in the data engineering domain: DogmatiX [31] fuses duplicates in XML; HumMer [33] implements generic fusion strategies and operators [32]. Current methods, however, do not consider all characteristics of situation awareness, such as the qualitative nature of spatio-temporal information as revealed by our previous work [38]. In principle, data mining and clustering techniques are applicable to duplicate detection too, but fall short of supporting spatio-temporal concepts as well [39].

3.2. Perception Ontology Module

To provide a uniform representation of relevant concepts found in various domains, as discussed in

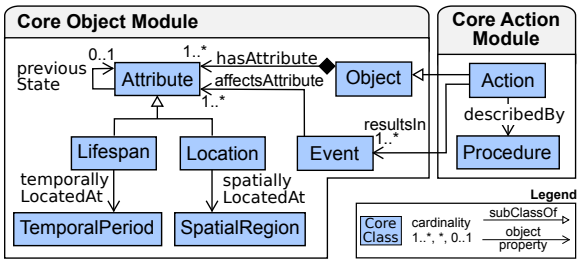


Figure 3: Core object and action ontology module.

the previous section, we have developed a *core object* and *action ontology module* for capturing basic information about *objects* and *actions*. This core module reflects and combines various ontologies and approaches from context- and situation awareness research [40]. It provides the relevant temporal and spatial concepts for our assessment algorithms. Since temporal and spatial aspects are central to enable comprehension and projection, BeAware not only references an existing temporal and spatial ontology in the core perception ontology module that can be used out-of-the box if appropriate, but also provides extension points for temporal and spatial ontologies to customize those aspects. Domain-specific extensions, for instance for road maintenance information, traffic sensor information, or traffic control actions, specify the vocabulary for describing relevant information about objects and their attributes. This domain vocabulary extends the core object and action module, and is specified by domain experts during the configuration phase of an application.

Figure 3 summarizes a subset of the perception module sufficient to support the discussion in this section. In order to exemplify the integration with existing ontologies, below we list sample mappings to the Situation Theory Ontology STO [5] (prefix sto:), CONON [21] (prefix conon:), and SOUPA [20] (prefix soupa:). Alignment can be performed in both directions: (i) our concepts subclass existing ontologies, which would allow for re-use of existing algorithms on our A-Box, and (ii) subclasses of our concepts refer to existing ontologies, which would allow our algorithms to work on other A-Boxes.

Objects—akin to SpatialTemporalThings in SOUPA and Individuals in STO—have attributes, such as their *location* in a *spatial region* and their *lifespan*[2] in terms of a *temporal period*, cf. (3). They are thus a union of what is considered either an object or an event in other ontologies.

$$\begin{aligned} \mathit{Object} &\sqsubseteq \mathit{sto{:}Individual} \\ &\quad \sqcap \mathit{soupa{:}SpatialTemporalThing} \\ \mathit{Object} &\equiv \exists \mathit{hasAttribute}.\mathit{Attribute} \\ \mathit{Attribute} &\sqsubseteq \mathit{sto{:}Attribute} \\ \mathit{Location} &\sqsubseteq \mathit{Attribute} \sqcap \mathit{conon{:}Location} \\ &\quad \sqcap \mathit{soupa{:}SpatialThing} \\ \mathit{Location} &\equiv \exists \mathit{hasValue}.\mathit{SpatialRegion} \\ &\quad \sqcap \leq 1.\mathit{hasValue}.\mathit{SpatialRegion} \\ \mathit{Lifespan} &\sqsubseteq \mathit{Attribute} \\ \mathit{Lifespan} &\equiv \exists \mathit{hasValue}.\mathit{TemporalPeriod} \\ &\quad \sqcap \leq 1.\mathit{hasValue}.\mathit{TemporalPeriod} \end{aligned} \tag{3}$$

Of course, locations and lifespans are disjoint ($\mathsf{Location} \sqcap \mathsf{Lifespan} \equiv \bot$), and both are unique in terms of the underlying spatial or temporal region. Evolution of attribute values over time is captured by listing previous states, thus creating a temporal sequence; not every attribute necessarily has such a previous state ($\mathsf{Attribute} \sqsupseteq \mathsf{previousState}.\mathsf{Attribute}$).

By introducing STOIndividual as a subclass of object, which refers to an sto:Individual[3], existing STO A-Boxes could be aligned with our ontology, cf. (4).

$$\begin{aligned} &\mathit{STOIndividual} \sqsubseteq \mathit{Object} \\ &\mathit{STOIndividual} \equiv \exists \mathit{refersTo}.\mathit{sto{:}Individual} \end{aligned} \tag{4}$$

Actions are described by procedures and result in *events*[4], cf. (5). Events are instantaneous ($\mathit{Event} \sqsubseteq \exists \mathit{at.time{:}Instant}$), spaceless ($\mathit{Event} \sqcap \exists \mathit{hasAttribute}.\mathit{Location} \equiv \bot$), and affect attributes; thus, they differ from SOUPA.

$$\begin{aligned} \mathit{Action} &\sqsubseteq \mathit{Object} \sqcap \mathit{soupa{:}Action} \\ \mathit{Action} &\sqsubseteq \exists \mathit{describedBy}.\mathit{Procedure} \\ &\quad \sqcap \exists \mathit{resultsIn}.\mathit{Event} \end{aligned} \tag{5}$$

By reifying actions as objects, our assessment and projection components not only work with observed objects, but also with actions taken by an operator. We use the term Action only for ones that can be initiated in our system. Thus, there are examples

[2] Lifespans resemble the temporal database concept *valid time* [41].

[3] Of course, one would also need to align attributes, relations, and so forth in that manner.

[4] Events capture meta-information, such as the temporal database concept *transaction times* [41].

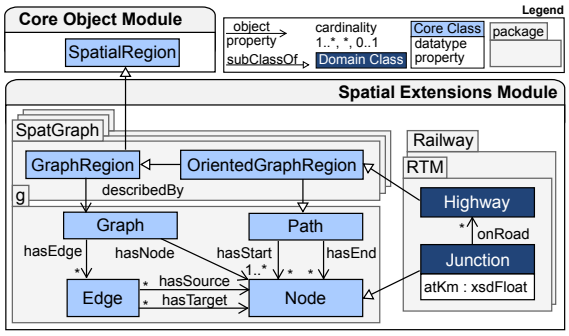


Figure 4: Spatial extension of the object module.

of objects that might be considered actions (e. g., roadworks, which are scheduled through another system and only observed in BeAware).

Temporal, spatial, and action extension points. In order to choose from or extend the set of provided spatial, temporal, and action modules, we generalize the approach outlined by Bateman and Farrar in their spatial ontologies survey [42] to accommodate temporal and action ontologies as well. In particular, objects themselves do not constitute space; rather, they are located at spatial regions to disambiguate statements about space. Without explicit spatial regions (e. g., as in SUMO [43]), "being a proper part of a tunnel" is ambiguous: do we refer to a composition of a tunnel and its parts, or state that an otherwise unrelated object resides inside the spatial region occupied by the tunnel? In our ontology, a "brick being a proper part of a tunnel" represents the former case, whereas a "car's region being a proper part of a tunnel's region" describes the latter. Likewise, objects are decoupled from temporal periods and procedures. The concepts SpatialRegion, TemporalPeriod, and Procedure (depicted in Figure 3) serve as extension points for concrete concepts.

Below, we take a closer look at how to extend the object module with a sample *spatial graph* module, depicted in Figure 4, which generalizes concepts from road traffic control (e. g., nodes represent junctions, edges the roads connecting them). In other domains, different modules may be required, such as geographic coordinates in air traffic control. For defining the graph extension, we derive GraphRegion from SpatialRegion. We re-use an existing ontology [44] (prefix g:), which defines a graph as a non-empty set of nodes connected by edges, cf. (6).

$$\begin{aligned} \mathit{GraphRegion} &\sqsubseteq \mathit{SpatialRegion} \\ \mathit{GraphRegion} &\equiv \exists \mathit{describedBy}.g{:}\mathit{Graph} \\ g{:}\mathit{Graph} &\equiv \exists \mathit{hasNode}.g{:}\mathit{Node} \\ g{:}\mathit{Edge} &\equiv \exists \mathit{hasSource}.g{:}\mathit{Node} \\ &\quad \sqcap \exists \mathit{hasTarget}.g{:}\mathit{Node} \end{aligned} \tag{6}$$

In addition, such a graph region may be oriented, meaning that it occupies a path within the graph ($\mathit{OrientedGraphRegion} \sqsubseteq \mathit{GraphRegion} \sqcap g{:}\mathit{Path}$). This path spans the edges from a start node to an end node, as depicted in Figure 4.

The spatial concepts presented so far are still domain-independent; developers may integrate additional characteristics of their domain—for instance, road names as path labels or edge lengths—by deriving from the concepts in the graph module, thereby anchoring them in another domain-dependent spatial module. For example, the road network type Junction provides additional information for Nodes, such as the corresponding highway (onRoad) or the distance to the beginning of a route (atKm). Subclasses of Edge (e. g., road elements) and Graph (e. g., a highway) may define domain concepts more precisely. For deriving situations with a spatial extension, relation interpretations (e. g., meaning of "very far" in a graph) must be supplied, as discussed in Section 4.

The temporal extension is based on *OWL time* [45] (prefix time:) with proper intervals as temporal periods whose beginnings and ends differ, cf. (7).

$$\begin{aligned} \mathit{ProperInterval} &\sqsubseteq \mathit{TemporalPeriod} \\ &\quad \sqcap \mathit{time}{:}\mathit{ProperInterval} \end{aligned} \tag{7}$$

In the case of action modules, a multitude of modeling possibilities and requirements from different domains open up, ranging from single actions to complete workflows (e. g.YAWL [46]). We decided in favor of Petri Nets, because they are backed by quite a large community as workflow language (e. g., the semantics of UML activity diagrams, or specific workflow definition languages, such as YAWL [46]). BeAware includes the Petri net ontology of Gašević and Devedžić [47] (prefix ptn:) as action extension, as defined in (8).

In the module, a Petri net consists of two kinds of model elements. *Arcs*, which specify workflow execution, connect two distinct kinds of *nodes*: *places*

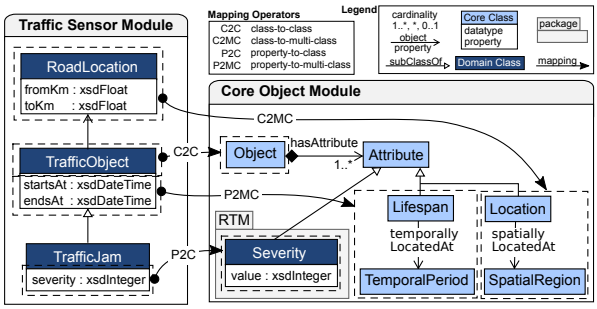


Figure 5: Mapping to core object concepts.

represent states in an RTM workflow, whereas *transitions* represent workflow tasks such as activating a message sign or closing a road.

$$\begin{aligned} \mathit{PetriNet} &\sqsubseteq \mathit{Procedure} \\ \mathit{PetriNet} &\equiv \exists \mathit{refer.ptn{:}Net} \\ \mathit{RtmWorkflow} &\sqsubseteq \mathit{PetriNet} \\ \mathit{RtmState} &\sqsubseteq \mathit{ptn{:}Place} \\ \mathit{RtmTask} &\sqsubseteq \mathit{ptn{:}Transition} \\ \mathit{ActivateSign} \sqcup \mathit{CloseRoad} &\sqsubseteq \mathit{RtmTask} \end{aligned} \tag{8}$$

3.3. Perception Components

At runtime, application-specific *adaptors* bridge structural heterogeneities and stream information about individuals from heterogeneous data sources into domain module A-boxes. Adaptors can provide an initial assessment of the confidence or uncertainty of the provided information (e. g., on the basis of the trustworthiness of the data source). The *domain mappers* of BeAware then allow domain concepts and individuals to be aligned with the core modules, thus keeping component implementations independent of domain-specific information. For this purpose, domain mappers are configured with declarative *mapping definitions*, which are expressed in terms of mapping operators that resolve structural heterogeneities. In addition to simple homogeneous operators (such as class-to-class [48]), we provide operators for extending modules with additional concepts and new individuals. These operators are based on research in heterogeneous mappings (e. g., property-to-class), mapping algebras [49], and design patterns for ontology alignment [50].

Figure 5 shows a sample mapping—which uses class-to-class, class-to-multi-class, property-to-class, and property-to-multi-class operators—between concepts in the road traffic management domain and the object module. Mapping operators are, however, not the focus of this paper; more details on possible heterogeneities and mapping operators to resolve them can be found in [51, 52].

Duplicate detection and data fusion. While structural heterogeneities on the schema level are resolved by the domain mapper as pointed out above, the data itself must be fused into a single consistent form at the instance level. Duplicate detection methods are key to exploring whether information with, e. g., different origins or observation times relate to one and the same real-world object. Duplicate information items about a particular object can increase the confidence if they agree, or lower the confidence if they disagree.

We envision a twofold approach to duplicate detection: (i) domain experts can express their knowledge to detect identical information using rules (e. g., two objects are duplicates, if a relation of type Equal holds between their spatial regions); (ii) incomplete and contradictory information can be detected using similarity measured in conceptual neighborhood graphs (CNGs, [14]). Knowledge of domain experts and analysis of recorded information configure the similarity threshold. BeAware supports both approaches at an early stage [53, 11].

Reconstructing object histories. To ensure consistency, duplicates must be fused into one coherent view that describes the history of an object in terms of the chronological evolution of its attribute values. Different data source update sequences result in 13 possible update cases (e. g., valid times may overlap) according to Allen's [13] enumeration of possible relationships between the valid time of a previous and a current attribute value, as discussed in our previous work [10]. In case multiple attribute values are in conflict, we resolve them employing data integration strategies, which can be configured. For example, let us consider information about an object o reported by one data source, and duplicate information d reported a bit later by another data source. Let us further suppose that the duplicate information only covers a time interval during o ($\mathit{during}(d, o)$). Using the strategy "take the information" [32], the newer information would be used, whereas with the strategy "trust your friends" [32] only the information from the more trustworthy data source would be used later on. A detailed discussion of possible strategies can be found in [32].

4. The Comprehension Layer

According to Endsley [8], comprehension depends on the way humans combine and interpret information. In Situation Theory [54], this is a "cognitive activity [that] categorizes situations in terms of objects having attributes and standing in relations to one another at locations—connected regions of space-time". From a technical point of view, the challenges therefore include implementing *efficient assessment algorithms* based on definitions of *relation and situation types* for detecting current situations, and *action types* for resolving them.

Definition of spatio-temporal relation types. In Situation Theory, relations are a notion central to comparing objects from various aspects, such as space and time. The challenge is to facilitate application development with an extensive set of *predefined spatio-temporal relation types* (e. g., Close and Before) that are applicable in various domains. The set of relation types must be extensible and the *interpretation* of their meaning configurable. For instance, in road traffic we define that Close means "within 1 km", whereas in other domains, it might be interpreted as "in the same room".

Definition of situation types. Another challenge is to specify rules for detecting situations from objects and relations between them. For this purpose, situation assessment must be configurable with *situation types* (e. g., TrafficJamCloseToAccident) and with their accompanying *situation type definitions* (e. g., CloseTo(Accident, TrafficJam) $\supset$ TrafficJamCloseToAccident). The latter define preconditions on the basis of object types and relation types met by objects and relations, respectively, for instantiating a situation type at runtime. Instances of such assessed situation types must themselves be reified as objects. This is not only important for retrieval, but also facilitates re-use and allows developers to apply functional decomposition in order to handle complexity at the type level.

Alignment of action and situation types. The operator must be provided with a viable set of actions in response to a certain object (e. g., TrafficJam instances can be resolved with the action type OpenEmergencyLane) or situation (e. g., all actions applicable in a situation of type TrafficJamCloseToAccident). We must consider not only situation-specific action type definitions (i. e., actions that must or must not be taken in a particular situation), but also the more challenging detection of appropriate actions *across* situations, which deal, for instance, with finding actions for situations sharing objects. For example, a traffic jam instance may contribute to a situation of type TrafficJamCloseToAccident and to another situation of type TrafficJamNearFootballGame. Thus, actions in response to the first situation may also be applicable for resolving the second situation.

Table 3: Summary of comprehension-related work

Criterion	BioPHusion [55]	JDL IF [56]	SA IF[57]	STDF [58]	DOLCE [59]	SAWA [5, 60]	AKTiveSA [61]	Pervasive [62]
Spatio-temporal relation types								
Pre-defined relation types								
Extensible relation types		✓		✓		✓		✓
Configurable interpretation			✓					
Situation types								
Extensible with situation rules	✓	✓	✓	✓	✓	✓	✓	✓
Situations as objects						✓		✓
Action types								
Situation-specific actions							✓	
Cross-situation actions								
Assessment								
Quantitative-value-derived relations		✓	✓	✓				
Rule-based situation assessment		✓	✓	✓	✓		✓	

4.1. Related Work

Table 3 summarizes related work on the comprehension level, w.r.t our challenges listed above. We build upon the notion of a "situation" as defined by Barwise and Perry [54] in their well-known Situation Theory, which emphasizes relations between objects as the central notion of describing situations. The information fusion community has identified such situations as the key solution to information overload [56] more than a decade ago. In this respect, many works are concerned with the peculiarities of particular domains, such as BioPHusion in the health sector [55] or sensor fusion in military environments [63]. Generic frameworks are the conceptual architecture for Endsley's definition of situation awareness (SA IF [57]), the state-transition data fusion model (STDF [58]) and systems based on the JDL data fusion model (JDL IF [56]). Unlike BeAware, concrete data models, components, algorithms, and scalability are not their focus.

In the area of semantic-web-based situation awareness approaches, situations have been recognized as an important instrument for reducing information overload. Examples include an extension to the DOLCE infrastructure supporting situations

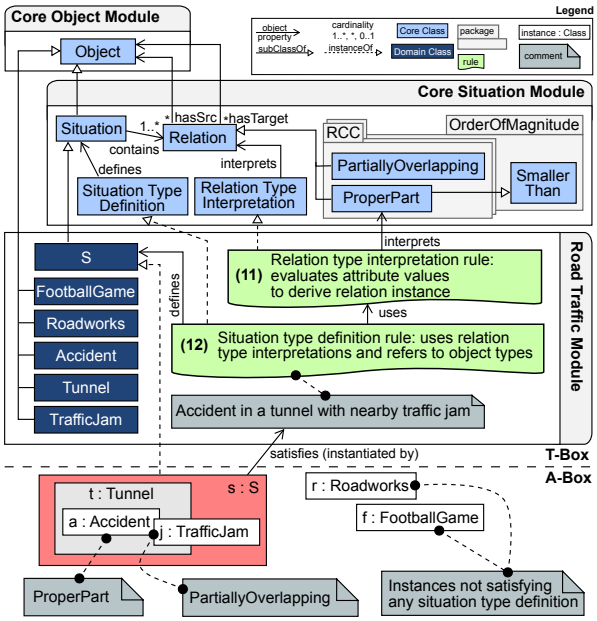


Figure 6: Formalization of a situation type definition on the basis of the core situation concepts.

[59], the situation ontology of SAWA [5, 60], the situation awareness prototype AKTiveSA [61], and situations in pervasive computing [62]. Current approaches, however, do not include relations facilitating spatio-temporal reasoning, but put the burden of modeling on application developers defining rules. Although being very generic, this approach leads to a number of drawbacks: (i) ontologies and rules are hard to keep consistent [64]; (ii) modeling effort on behalf of the application developer is increased; (iii) data fusion features, such as object history reconstruction, are missing; (iv) algorithm optimization is not being focused on. In BeAware, we concentrate on integrating well-established relation types from spatio-temporal reasoning, and on supporting application developers with modeling concepts for defining relevant types of situations, as well as on optimization strategies for relation and situation assessment algorithms.

4.2. Comprehension Ontology Module

The core situation module, which is depicted in Figure 6, follows the Situation Theory [54]. Hence, it is compatible with other SAW ontologies, such as STO [5]. The essential difference is that our approach is extensible with a wide range of relation types, which are assessed from object attributes,

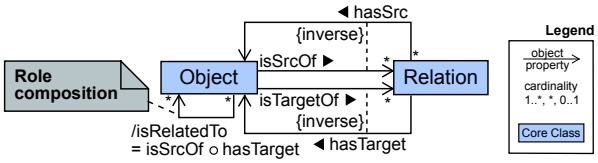


Figure 7: Object property by role composition.

and incorporates relations and situations as first-class citizens into the ontology. The situation module extends the core object module with two main types. Relation is an abstract concept for integrating concrete relation types such as PartiallyOverlapping (which is instantiated in case two objects overlap partially), whereas Situation integrates concrete situation types of a domain. Situations are spatio-temporal ($Situation \sqsubseteq Object$) and connect objects with relations ($Situation \equiv \exists contains.Relation$). To associate further information with relations (e. g., valid times), the core situation module represents them as individuals and only derives object properties when needed (at the cost of redefined meta-information of OWL object properties, such as symmetry for the type Relation), cf. (9).

$$\begin{aligned} Relation &\sqsubseteq \exists hasSrc.Object \\ &\sqcap \exists hasTarget.Object \\ SymmetricRel &\sqsubseteq \exists isSymmetric.true \\ &\sqcap \forall isSymmetric.true \end{aligned} \tag{9}$$

Still, relations between objects are derived as object properties using *role composition*. In (10)—summarized in Figure 7—the composition of a relation's source object (an instance of Object) and its target is interpreted as an object property isRelatedTo between the relation's source and target.

$$hasSrc^{-} \circ hasTarget \sqsubseteq isRelatedTo \tag{10}$$

Spatio-temporal relations. In Situation Theory, major emphasis is put on the relations between objects. The design rationale behind BeAware (cf. [6] for details), is to re-use the vocabulary and semantics of well-established *relation calculi* from the field of spatio-temporal reasoning, such as the Region Connection Calculus (RCC, [65]), the Oriented Point Relations Algebra (OPRA, [66]), Cardinal Directions [67], Distances [68], the Order of Magnitude Calculus [69], Allen's Time Interval Algebra [13], and Freksa's Temporal Semi-intervals [70]. Figure 6 shows three sample spatial relation types

ProperPart (i. e., an object is part of another object in spatial terms), PartiallyOverlapping (i. e., one object's spatial region overlaps another object's region), and SmallerThan integrated within RCC and the Order of Magnitude Calculus.

In order to detail the meaning of a relation type in a particular domain, as well as to derive relations from the attribute values of objects, developers can provide a *relation interpretation* for each relation type. In Example 3, a sample relation interpretation of the relation type ProperPart is given using a Horn-like rule: an individual o being a member of the class C is expressed as C(o), (e. g., Object(o)), whereas two individuals being related by the object property P are expressed as P(o,o') (e. g., hasAttribute(o,l)). These rules use the usual logical connectives for negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), and material implication ($\supset$).

Example 3. *An object o—according to the semantics of RCC—is considered to be a proper part of another object o', if the location of o is contained inside the location of o', which is determined by comparing their start and end points, cf.* (11).

$$\begin{aligned} & Object(o) \wedge Location(l) \wedge hasAttribute(o,l) \\ \wedge\ & Object(o') \wedge Location(l') \wedge hasAttribute(o',l') \\ \wedge\ & l.hasStart.atKm > l'.hasStart.atKm \\ \wedge\ & l.hasEnd.atKm < l'.hasEnd.atKm \\ \supset & ProperPart(r) \wedge hasSrc(r,o) \wedge hasTarget(r,o') \end{aligned} \tag{11}$$

Note, that the syntax employed in (11) for specifying a Horn-like rule is oversimplified, since in current Horn-like rule languages (such as SWRL [71]) it is not possible to use variables in the consequent of a rule that have not been introduced in its antecedent. To create a new individual, essentially, one may use a SPARQL template with blank nodes.

In order to define subsumption lattices of relation types, we define the semantics of a relation type w.r.t. others in the framework: for example, the relation type ProperPart of RCC obviously subsumes the relation type SmallerThan. Thus, we may not only assist developers with consistency checks (e. g., issuing a warning if a developer combines ProperPart and LargerThan), but also take a shortcut in relation assessment (e. g., when an object is a proper part of another object, we know that the first object is smaller than the second one, even without checking the interpretation of SmallerThan).

Situation types. Any two real-world objects are, as described above, in many different relationships with each other[5]. However, only a subset of the potential relation and object combinations are of interest to an operator (e. g., whether or not a particular tunnel is older than some bridge might be irrelevant). A developer can extend the core situation module to define the relevant situation types by forming sub-classes of Situation. For example, in Figure 6, a situation type S is depicted, which represents a class of situations whose members can be described informally as "accident in a tunnel causing a traffic jam" situations.

Such situations (i. e., instances of a situation type) are identified by the involved objects (i. e., instances of Object) and the relations (i. e., instances of Relation) among them. *Situation type definitions* extend the core situation module to define patterns for both, as introduced in [6].

Example 4. *Following* (12)*, a situation type S should be instantiated, when (i) an instance of Accident and an instance of Tunnel satisfy the relation interpretation of ProperPart (i. e., the accident is inside the tunnel), and (ii) the same accident and an instance of TrafficJam satisfy PartiallyOverlapping (i. e., the traffic jam adjoins the accident).*

$$\begin{aligned} & Accident(a) \wedge Tunnel(t) \wedge TrafficJam(j) \\ \wedge\ & isProperPart(a,t) \wedge isPartially- \\ & Overlapping(j,a) \supset S(s) \wedge contains(s,a) \\ & \wedge contains(s,t) \wedge contains(s,j) \end{aligned} \tag{12}$$

The process of evaluating such definitions is part of the assessment described in the next section.

4.3. Comprehension Components

The *situation assessor* of the comprehension subsystem finds relations between objects that satisfy relation type interpretations. These relations are the basis for assessing situations, which subsequently can be resolved with actions suggested by the *action assessor*.

Situation assessor. In general, situation assessors solve a three-step satisfiability problem—i. e., they check which object configurations match situation type definitions: First, objects are selected

[5] Exactly $n \cdot (n-1)/2$ pairs among n objects; when relations are joint exhaustive, the number of relations for each pair is at least the number of relation calculi.

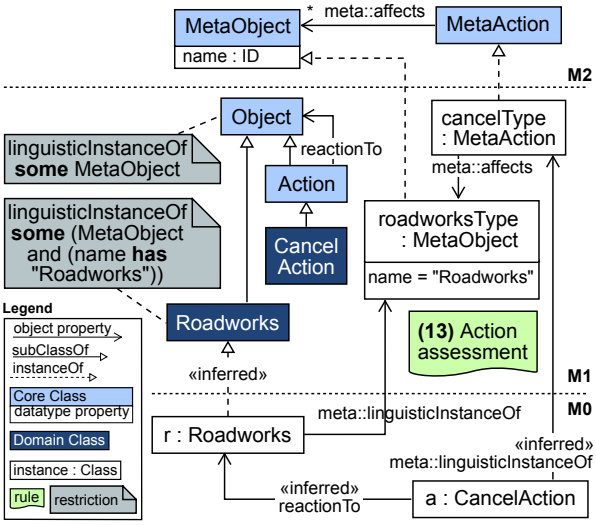


Figure 8: Action type for a particular object.

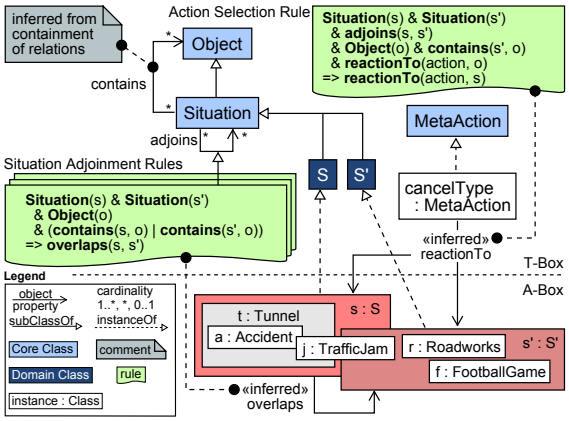


Figure 9: Action type for overlapping situations.

according to the situation type definitions. Second, the selected objects are tested with relation type interpretations to assess relations. This step is time-consuming (quadratic in the number of objects). In Section 7 we discuss, how clustering techniques help to reduce the number of direct object comparisons. Finally, an instance of the appropriate situation type is injected into the A-Box. A confidence measure about a detected situation could be derived from the contained objects and the boundaries of relation types (e. g., lower confidence if the objects just barely satisfy a relation type). The worst-case runtime complexity of situation assessment and possible shortcuts are discussed in [72].

Action assessment. Action assessment makes an operator aware of reactions to current situations. BeAware provides three strategies: Firstly, it associates action types with object types and situation types; thus, actions are suggested when a corresponding object or situation occurs. As defined in (13), action assessment suggests for any object o a particular action a of type ta, which is known to affect all objects of the same type as o.

$$\begin{aligned} &Object(o) \wedge linguisticInstanceOf(o,t) \\ \wedge &affects(ta,t) \supset linguisticInstanceOf(a,ta) \\ &\wedge reactionTo(a,o) \end{aligned} \tag{13}$$

In the example depicted in Figure 8, it is inferred that cancelType is a reaction to roadworks r, because cancelType is known to affect roadworksType, of which r is an asserted meta::linguisticInstanceOf.

Secondly, the effects of actions on attributes can be used to select appropriate actions on the basis of object attributes within a current situation. Such effects can be modeled in the ontology in a fine-grained manner by specifying (i) the types of events in which an action results, and (ii) how such events affect a particular type of attribute. For example, an OpenEmergencyLane action type increases throughput (modeled by subclassing the affectsAttribute relationship) of a particular road segment, and can therefore be selected as an appropriate reaction to a traffic jam on the same road segment (again, on the basis of the meta-modeling support).

Finally, actions can be selected based on situations that adjoin each other, which is indicated by objects that contribute to multiple situations (i. e., situations that overlap, which is modeled as a sub-property of the adjoin OWL object property). These OWL object properties are inferred using situation adjoinment rules, as depicted in Figure 9, and exploited in action selection rules. For a given situation, the action selection rule shown in Figure 9 and formalized in (14) queries all actions that affect any object of any overlapping situation: Applied to situation s, it returns CancelType, which actually belongs to the overlapping situation s' per a shared TrafficJam instance.

$$\begin{aligned} &Situation(s) \wedge Situation(s') \wedge Object(o) \\ &\wedge adjoins(s,s') \wedge contains(s',o) \\ &\wedge reactionTo(a,o) \supset reactionTo(a,s) \end{aligned} \tag{14}$$

5. The Projection Layer

Anticipating future situations and projecting their implications enables timely decision making, which is the highest level of situation awareness [8]. This entails the challenges of (i) anticipating the *evolution of situations*, and (ii) counteracting situations with *action planning*.

Evolution of situations. To anticipate critical situations, we need mechanisms for representing and reasoning about the *evolution* of situations. This is particularly difficult, since we are dealing mainly with qualitative information about objects. Specific problems to be solved, in this respect, are (i) generic projection mechanisms and (ii) distance measures, such as duration, costs, impact, and likelihood, between neighboring situations (e. g., how long does a transition from a situation of type WrongWayDriver*Far*TrafficJam to one of type WrongwayDriver*Close*TrafficJam take?).

Action planning. Action assessment must take into account situations that may not have emerged yet, but are likely to do so in the near future. In this sense, action planning is the basis for pro-active decision making to positively influence the evolution of situations. Thus, we must define algorithms that find appropriate actions for anticipated situations, as well as actions that help to meet particular goals.

5.1. Related Work

The ontology-driven situation awareness approaches of related work discussed above do not focus on projection techniques. We therefore discuss approaches from the data fusion community, qualitative projection in spatio-temporal reasoning, and recommender systems. Table 4 summarizes related work on the projection level, w.r.t. our challenges.

Holsopple and Yang introduce the data fusion system FuSIA [73] for threat assessment of cyber attacks. Their focus is the impact of plausible future situations. For projection, FuSIA combines *observed* hacker behavior, capability, and intent with knowledge about opportunity modeled in a virtual terrain (e. g., a computer network's topology). In many domains, such as in road traffic management, object behavior often cannot be learned from observations due to the criticality of situations. In BeAware, therefore, models describe behavior from an a-priori viewpoint[6].

[6] Note, that, according to Holsopple, hacker behavior should ideally be represented with models as well [83].

Table 4: Summary of projection-level-related work

Criterion	FuSIA [73]	QSIM [74, 75]	QPE [76, 77]	QTC [78]	Agent control [79]	CBQS [80]	RSAC [81]	SNAP [28]	Prob. Sit. Calc. [82]
Situation evolution									
Qualitative projection		✓	✓				✓		
Distance measures				✓	✓				
Object characteristics		✓	✓		✓	✓	✓		
Relation interdependencies		✓	✓	✓		✓	✓		
Action planning									
Actions for anticipated situations								✓	
Planning towards goal situations						(✓)		✓	✓
Action impact projection									
Modeling of action effects	✓						✓	✓	✓
Action effect projection	✓						✓		
Cost and likelihood of actions									✓

Projection in road traffic management, to date, focuses on extrapolating current traffic flow with quantitative methods [84, 85]. In case that reliable numerical values are hard to obtain, qualitative approaches, however, are as already mentioned better suited than quantitative ones [3]. Two areas of major research directions focus on such qualitative projection. First, methods for projecting physical processes with qualitative differential equations abstract from numerical simulations [86]. Second, methods based on conceptual neighborhood between relations [74, 75]—e. g., modeled in conceptual neighborhood graphs (CNGs, [70])—focus on projection of processes that continuously vary [87].

The central notion of relations on the comprehension level makes the second area a natural choice for situation awareness. CNGs are used, for instance, in qualitative simulations (QSIM [74, 75]), in qualitative projection and envisioning (QPE [76, 77]), for tracking moving objects (QTC [78]), or in agent control [79]. However, the fact that neither object characteristics (e. g., whether the involved objects are moveable) nor interdependencies between calculi (e. g., transitions in mereotopology may depend on spatial distance) are considered within such CNGs leads to inaccurate and rudimentary situation projections [88].

Existing approaches such as in qualitative simulation [80, 89], try to increase projection quality by manually constructing domain- and even situation-specific relation calculi. Such an approach demands considerable modeling effort from the application developer: In [80], a constraint-based qualitative

simulation (CBQS) method is presented, which requires human operators to manually define starting points and constraints on CNG transitions for simulations more precisely. Dynamic constraints in reasoning about space, actions, and change (RSAC [81]) provide similar manual means for describing valid transitions to determine subsequent simulated situations. In BeAware, we have the goal to derive customized CNGs automatically to reduce modeling effort while keeping CNGs domain-independent.

Action planning and action impact projection is a major issue in recommender systems. In particular, the recommender system SNAP [28] performs reasoning to recommend a set of actions, which reflect the steps to attain a desired goal. Our aim is to complement such action assessment for current situations with action planning for possible future situations. Also, situation calculus-based approaches, such as [28], assume deterministic action outcomes to reason about action impacts. As a generic framework, BeAware aims to not force developers into a single outcome interpretation, but seeks to allow also actions entailing non-deterministic effects. That is, we have to provide extension points in our ontology modules for defining domain knowledge about action impacts, and develop impact projection algorithms tackling the projection of possible outcomes differently based on concepts such as uncertainty [90] and probabilistic approaches (e. g., Probabilistic Situation Calculus [82]).

5.2. Projection Ontology Module

Conceptual neighborhood graphs [70], as introduced above, impose constraints on the existence of transitions between relations of a calculus, and hence, on the evolution expressible w.r.t. the calculus. Two relations are conceptual neighbors "if a direct transition from one relation to the other can occur" [70] . Such transitions define in which directions a current situation—defined by its relations—may evolve. Conceptual neighborhood is the basis for projection in BeAware. Figure 10 shows an excerpt of the core projection module, including a sample CNG of RCC. The core projection module is a major extension of our previous work [91] with respect to the incorporation of object characteristics [88] using meta-modeling concepts.

The CNG of RCC consists of five relations (illustrated by the objects o and o' according to the meaning of the relation: e. g., Disrelated is represented by two circles with a gap) and models the possible neighborhood in-between. For example, if

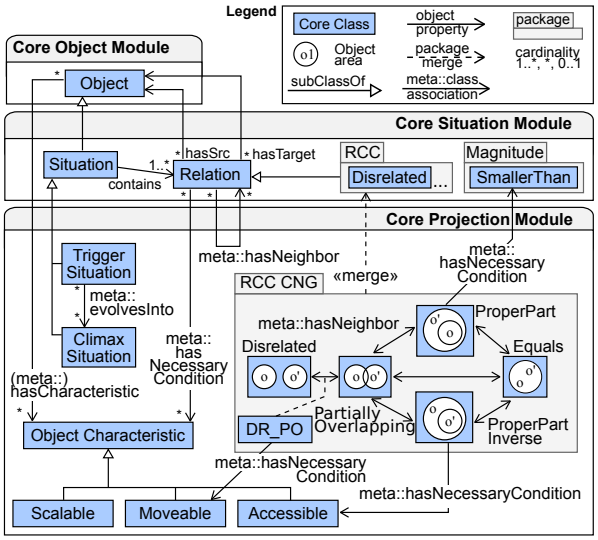


Figure 10: The core projection ontology module.

a relation of type Disrelated exists between o and o', an instance of ProperPart cannot hold true unless an instance of PartiallyOverlapping holds true first.

As already stated above, existing approaches (e. g., [66]) try to increase quality by manually constructing domain- and even situation-specific calculi. In the core projection module, we use our meta-modeling concepts to annotate conditions on CNGs concerning (i) objects w.r.t. their characteristics (meta::hasNecessaryCondition with range object type), and (ii) other relation types w.r.t. interdependencies (meta::hasNecessaryCondition with range relation type). Projection components use these conditions to automatically customize calculi.

Concerning object characteristics, application developers can either define the particular kind of change (meta::hasCharacteristic) supported by a concrete object type during modeling (and hence, applicable to all objects of that type as a default behavior), or they can define rules that derive them from the attributes of individual objects (similar to relation interpretations).

In order to make implicit relation interdependencies explicit, a common model must be built to reintegrate relation calculi that were originally designed in isolation to abstract from the same aspect of the world (e. g., time or space) by applying different foci (e. g., topology vs. distance). Means for building such an integrated model are discussed in our previous work in terms of a so-called *combined dominance space* [92], which is based on Galton's

approach [93] to deriving a geometrical representation of relations. In summary, a combined dominance space models subsumption lattices, as well as necessary and sufficient conditions for relations and for transitions. In this paper, we exploit combined dominance spaces to derive the desired relation interdependencies.

Concerning subsumption lattices, geometrical containment of relations in a combined dominance space defines a subsumption hierarchy of relation types (e. g., as indicated by inheritance between ProperPart and SmallerThan in Figure 6).

Necessary conditions of relations are defined by geometrical neighborhood of relations, and must be specified in more complex form, cf. PartiallyOverlapping in Example 5.

Example 5. *PartiallyOverlapping (PO) cannot hold between two objects o and o′ in any situation s, when their centers are considered to be either far (F) or very far (VF) apart, cf.* (15).

$$(\forall o, o' \in O, s \in S) Holds(\phi_{dc}(o, o'), \gamma_1, s) \supset \neg Holds(\phi_{rcc}(o, o'), PO, s) \quad (15)$$
$$where\ \gamma_1 \in \{F, VF\}$$

Even more interestingly, a combined dominance space models necessary conditions for transitions between relations too, as described in Example 6.

Example 6. *Two objects that are currently disrelated (DR) may transition to partially overlapping, if they are: very close to each others boundary (VC) and have far apart centers (F), regardless of their relative size (summarized by a region that borders PO and spans all size relations [92]).*

$$(\forall o, o' \in O, s \in S) \quad Poss(tran(PO, o, o'), s) \quad Holds(\phi, \gamma_1, s) \quad (16)$$
$$\wedge\, \gamma_1 \in \{DR\} \times \{VC\} \times \{F\} \times \{<, =, >\}$$

5.3. Projection Components

The projection subsystem provides components for reasoning about future developments based on the results of the comprehension subsystem.

Qualitative situation projection. Qualitative situation projection can be based on (i) domain knowledge about specific situations, indicating that a critical one will emerge, and (ii) knowledge about possible transitions between relations. For this purpose, BeAware provides two qualitative situation projection mechanisms, which utilize the core projection module introduced above.

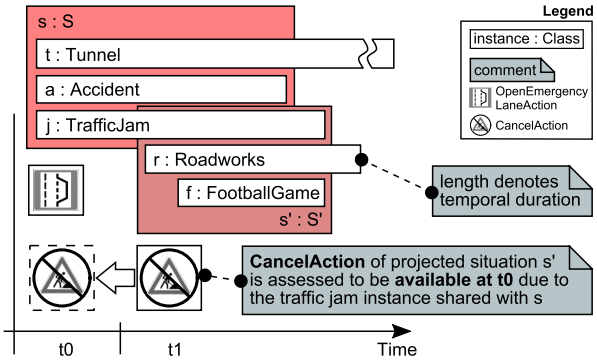


Figure 11: Situation enables pro-active actions.

The *landmark projection mechanism* uses extended situation type definitions to express the beginning and potential ends of situation evolutions as *landmarks*, as introduced in [91]. Such kinds of landmarks are *trigger* situations, which indicate that one or more *climax* situations may emerge. Landmarks enable domain experts to model critical situations that are frequently preceded by early indicators [91]. The landmark projection mechanism exploits such modeled knowledge to provide early warnings of climax situations, without projecting evolution paths (i. e., situations between the trigger and the climax situation).

To complement landmark projection with forecast evolution and to derive additional situations in the neighborhood of trigger situations, the *Situation Prediction Net (SPN) mechanism* (see [94] for details) simulates potential evolution when initialized with a current situation. For this purpose, we build upon hierarchical Colored Petri Nets [95] that are commonly known as appropriate models for describing the static and dynamic aspects of a system. SPNs are created from the projection module of BeAware and encode relation types and CNGs, object types and their attributes, as well as necessary and sufficient conditions for relations and transitions. The firing of transitions in an SPN generates markings; each marking denotes a possible future situation, which can be read from the SPN and injected as individual into the ontology.

Action planning. Instead of dedicated action planning algorithms, we apply the strategies for action assessment to action planning on the basis of projected situations (making reactions pro-active).

The example in Figure 11 shows a current situa-

tion s of situation type S representing an accident a in a tunnel t causing a traffic jam j. Since the traffic jam and the planned road works r coincide with a football game f, another situation s' of situation type S' at time t1 is likely to occur in the real world. Note that both, s as a current situation and s' as a projected situation are instantiated by the situation assessment and the situation projection algorithm, respectively, at time t0, even though the real world occurrence of s' lies in the future. As a result, at t0 the operator can be provided not only with an instance of the OpenEmergencyLane action type (an appropriate reaction to s), but also with an instance of CancelRoadWorks appropriate for the roadworks object contained in s'. Canceling road works before s' actually occurs in the real world could proactively prevent the traffic jam from worsening.

6. Implementation

This section introduces the technical architecture of BeAware and its implementation and configuration aspects. We describe two alternative implementations: one generic semantic-web-based implementation and a domain-specific, relational-database-backed implementation. The prototypical implementation on the basis of semantic web technology allows for exploiting the extension mechanisms described in this paper at the expense of runtime performance, see Section 7. The relational-database-backed implementation, in contrast, focuses on performance by a domain-specific implementation of our concepts.

6.1. Semantic-Web-Based Implementation

Architecture. BeAware follows a pipes-and-filters architectural style [10, 96], which is especially suitable in a data streaming environment. Such an architectural style promotes incremental data processing and enables distribution and parallelization of components, while it defines a uniform component interface with arbitrary component order (e. g., if necessary, one can use multiple situation assessors, each focusing on one type of situations).

Implementation platform. The core and domain ontology modules were defined using the ontology editor Protégé[7]. The RDF triple store AllegroGraph[8] was employed as an ontology repository, due to its capability of integrating distributed triple

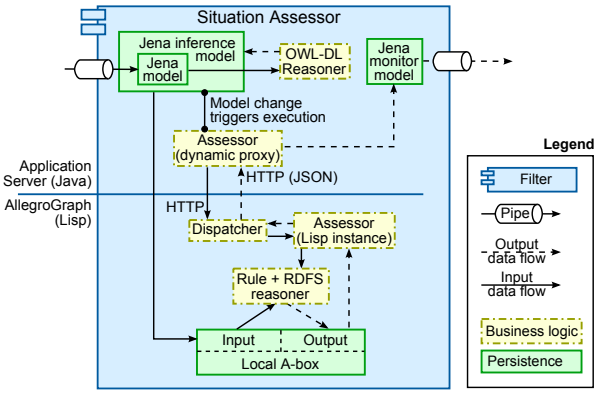


Figure 12: Java and AllegroGraph Lisp integration.

stores in a virtual, federated store, as well as its integrated, optimized RDFS, rule, and temporal reasoning capabilities. AllegroGraph can be accessed from Java via the Jena Semantic Web Framework[9]. In-server rule reasoning is supported via a combined Prolog and Lisp environment. Consequently, component implementations aim to exploit the advantages of both worlds, Prolog/Lisp and Java. On the one hand, communication overhead is reduced by in-server execution of native Lisp and Prolog code; on the other hand, developers using BeAware benefit from Java's capabilities in terms of configuration, automated testing, and integration with off-the-shelf frameworks. Therefore, as depicted in Figure 12, each component is split into the actual Lisp and Prolog implementation on the server side and a corresponding Java interface on the client side.

To reduce the effort that developers must put into using BeAware, we abstracted from the functionality provided by AllegroGraph and rely on Java's dynamic proxies to provide access to server-side Lisp object instances. For this purpose, we use Java's capabilities of creating dynamic proxies from interface specifications. Such a dynamic Proxy delegates method calls and marshalled parameter values to the server, where a dispatcher unmarshalles the supplied information to handle method invocations within Lisp. One of the advantages of this approach is that complex server-side components can be configured using Java component technology.

The situation projection component is built using the CPN Tools[10], which supports Java access [97]. Details on projection can be found in [94].

In order to study the performance gains achiev-

[7] protege.stanford.edu [8] agraph.franz.com

[9] jena.sourceforge.net

[10] wiki.daimi.au.dk/cpntools/

able by abandoning the generic approach, we introduced domain-specific clustering techniques in situation types. These clustering techniques reduce the number of objects prior to deriving pairwise relations, for instance, by creating clusters of objects being on the same road. Furthermore, we introduce a domain-specific implementation backed by a relational database and a rule engine that is closely coupled to an unmodifiable data model to compare the semantic-web-based implementation to traditional relational database techniques.

6.2. Relational-DB-Backed Implementation

Components in the relational-DB-backed prototype are implemented using a single technology, not split across multiple platforms, and optimized to the data model and data distribution of individuals per class of a domain.

Architecture. In contrast to the pipes-and-filters architectural style applied in the generic solution, the domain-specific implementation exchanges data purely via the knowledge base. This reduces communication between filters, which turned out to be a costly operation. Every component is run by a scheduler at a dedicated processing interval (e. g., adapters can adjust their import frequency to the underlying data source). When highly critical incoming messages require instant assessment (e. g., a wrong-way driver), an additional event notification mechanism triggers components explicitly.

Implementation platform. The domain-specific components are implemented using Java, Spring[11] for configuration purposes, Quartz[12] as task scheduler, Hibernate[13] to access the underlying knowledge base stored in a PostgreSQL[14] database with PostGIS[15] extension for spatial information, and JBoss Drools[16] as rule engine.

Implementation details. The data layer encapsulates the *domain model*. Unlike the generic, ontology-based implementation, the domain-specific implementation requires a complete model of the domain (i. e., all *object*, *relation* and *situation types* including their attributes) before the launch of the system. It is thus only allowed in a restricted manner to add novel situation types at runtime, as these are closely coupled to the employed database schema. Together, DB and GIS permanently store a complete history of the observed objects and inferred situations. The database schema, together with the Hibernate mappings, have to include the core classes as well as the domain-specific classes. As a result, adapters are not only statically linked to a data source, but also to the combined core and domain-dependent data model.

The *situation assessor* uses JBoss Drools as a rule engine, which allows us to define situation types as rules in a similar fashion to the semantic-web-based prototype. These situation types use the domain-dependent Hibernate mappings and are tightly coupled to the domain in terms of knowledge of the actual number of individuals per class: the order of clauses is optimized to reduce the state space early in the assessment process (i. e., rare object types are put upfront).

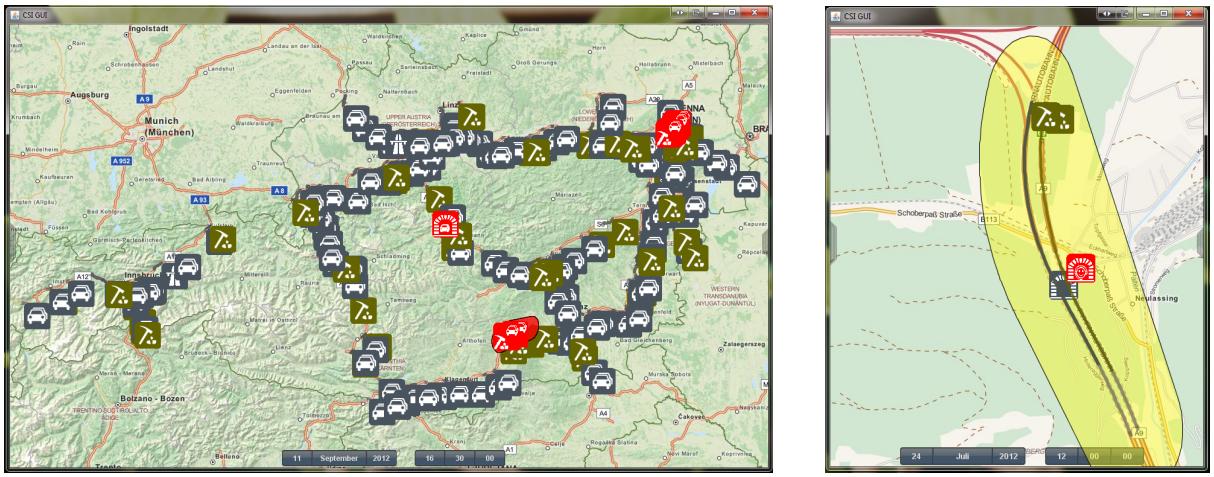
7. Performance Evaluation

To demonstrate the applicability of BeAware, we collaborated with our project partner Heusch-Boesefeldt GmbH[17], who provide a commercial platform for road traffic management systems (GeoDyn2). GeoDyn2 is used, for instance, by the Austrian highways agency (ASFINAG). This platform already provides adaptors for various information sources that are re-used to continuously update the road traffic domain ontology modules and the database of BeAware. We use these adaptors to evaluate BeAware on real-world data: based on the objects perceived we evaluate the runtime performance of situation comprehension components that are part of the comprehension layer. We are not focusing on the correctness of the implementation of the components (there exists a plethora of methods in the software engineering community to establish confidence about program correctness, such as unit testing and program verification) and not focusing on the correctness of the situation types themselves, since those are not part of the framework but provided by the configuration. Figure 13 shows a screenshot of the user interface with real-world traffic objects and derived traffic situations.

7.1. Real-world and Sampled Test Data

The evaluation is based on real-world traffic data recorded for Austrian highways over a period of four weeks (specifically May 25th, 2012 to June 25th,

[11] www.springsource.org
[12] quartz-scheduler.org
[13] www.hibernate.org
[14] www.postgresql.org
[15] postgis.refractions.net
[16] www.jboss.org/drools
[17] www.heuboe.de

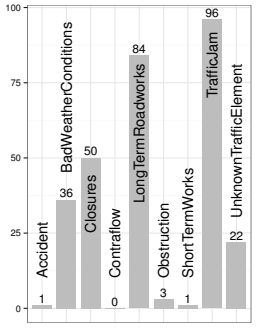


(a) Inferred critical situations in red, sensed information (i. e., all other traffic-related objects) depicted by icons.

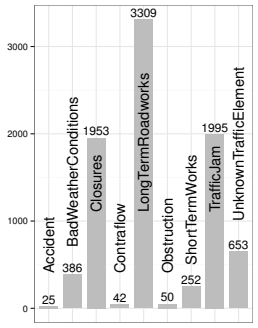
(b) Expanded situation shows constituting objects.

Figure 13: Screenshots of objects and situations in real-world traffic data of Austrian highways.

2012). The data set contains traffic information about (i) scheduled roadworks, (ii) automatically detected traffic jams, (iii) traffic-related incidents, (iv) weather conditions, and (v) manual traffic reports from a nation-wide broadcasting station. The number of individuals per class at a representative time point is summarized for the most important classes of our situation types in Figure 14a. The dominant objects are traffic jams, long term roadworks (duration of days to months), closed roads (closure), and bad weather conditions. Less often, we observe obstructions, accidents, short term works (e. g., street cleaning, duration of hours), contraflow, or other traffic incidents of unknown type.



(a) Distribution of observed traffic objects recorded on 6/17/2012 4:30pm (assessment summary in Table 5).



(b) Distribution of traffic objects in sampled data (performance results shown in Figure 15).

Figure 14: Evaluation data sets.

Only a rather small number of objects are present simultaneously in the real-world data set (about 270 on average). Thus, for evaluating the scalability of our implementations to larger environments w.r.t. the number of simultaneously handled objects, we sample a data set with 30 times the simultaneous objects from real-world data. In the sampled data set we want to largely preserve the traffic characteristics and composition of the real-world data set, which vary over the time of day [98]. Thus, we aggregate a month of recorded real-world data by shifting each day's active traffic objects (4:30pm) to June 6th, 2012 (4:30pm). As a result, the number of simultaneous traffic objects on June 6th, 2012 (4:30pm) was increased from 290 traffic objects in the real-world data set to 8667 in the sampled data set. This sampling introduces artificial duplicates (e. g., objects that span several days are duplicated for each day of their lifespan), which slightly changes data distribution (irrelevant for scalability). Figure 14b shows the distribution of objects per class in the sampled data set; it resembles the real-world distribution with the exception of boosting the number of long term roadworks.

7.2. Situation Types

For the evaluation in this paper, we configured situation assessment with seven critical situation types based on those modeled with domain experts from ASFINAG [72]. Table 5 summarizes these situation types and their definitions together with an

Table 5: Situation types (based on [72]) and their occurrences in the real-world data set.

	Description	Situation Type Definition	objs	Relation pattern 1st	2nd	spat	temp	full
S_0	Bad driving conditions near or within traffic jam	WeatherMessage(wm) ∧ StationaryTraffic(st) ∧ (VeryClose(wm, st) ∨ Intersects(wm, st))[1]	3456	25	25	25	–	25
S_1	Operator action entails traffic jam	OperatorAction(a) ∧ StationaryTraffic(st) ∧ VeryClose(a, st) ∧ StartedBefore(a, st)[2]	13056	46	43	46	12685	43
S_2	Poor driving conditions cause an accident	GeneralAccident(ga) ∧ WeatherMessage(wm) ∧ Intersects(wm, ga)[1] ∧ StartedBefore(wm, ga)[2]	36	0	0	0	0	0
S_3	Two traffic jams potentially merge	StationaryTraffic(st1) ∧ StationaryTraffic(st2) ∧ VeryClose(st1, st2) ∧ RCC8Disconnected(st1, st2)	338	106	30	30	–	30
S_4	A wrong-way driver heads towards roadworks	VehicleOnWrongCarriageway(w) ∧ Roadworks(r) ∧ (Commensurate(w, r) ∨ Close(w, r) ∨ VeryClose(w, r))	0	0	0	0	–	0
S_5	Accident causes a traffic jam	GeneralAccident(ga) ∧ StationaryTraffic(st) ∧ StartedBefore(ga, st)[2] ∧ VeryClose(ga, st)	6	4	2	3	4	2
S_6	Accident in area of roadworks	GeneralAccident(ga) ∧ Roadworks(r) ∧ Intersects(ga, r)[1]	85	0	–	0	–	0

[1] Intersects(x, y) := ¬RCC8Disconnected(x, y)
[2] StartedBefore(x, y) := (AllenOverlaps(x, y) ∨ AllenMeets(x, y) ∨ AllenStarts(x, y) ∨ AllenEquals(x, y) ∨ AllenContains(x, y))

informal description, and shows the number of assessed situations in the real-world data set (column *full*). In this section, we take a closer look on the constituents of a situation type and their effect in reducing matched objects: (i) The *objs* column lists the number of matched object pairs (first two conjuncts of the situation type definition). (ii) The column *1st* relation pattern intersects the matched object pairs with those objects satisfying the third conjunct (i. e., the first relation clause). (iii) The column *2nd* relation pattern further restricts with the fourth conjunct (if any), whereas (iv) *spat* and *temp* consider only spatial or temporal clauses.

By following a top-down [99] design approach for situation types, irrespective from the underlying data sources, an *ideal* situation awareness application as envisioned by domain experts was defined. An important lesson concerning the challenges of modeling situation types was learned: when modeling situation types top-down one is not especially aware of the available data sources; thus, the specificity of created situation types is hard to judge. For example, situation type S_3 matches rather often, which means that it is not particularly successful in reducing information overload. S_1 applies rather seldom w.r.t. the number one would expect from the object distribution, which indicates that the situation type is too focused. Thus, initial situation types need to be refined in an iterative manner as knowledge about the underlying data is gained.

Nevertheless, a significant information reduction can be achieved using the situation types in Table 5. Since the objects referenced in S_0 and S_1 appear very frequently in the data set, situation types help to identify the "interesting" occurrences. S_2 and S_6 never apply, although the corresponding objects exist in the system, because they do not interfere in the required spatial and temporal fashion. Concerning S_2, 32 of the object pairs in question are very far from each other and four are of commensurate distance and thus not even remotely interesting. Similarly for S_6: 78 of the object pairs in question are very far, one is far, four are of commensurate distance, and two are close. However, the two object pairs of close distance are not too far off an interesting situation, and could be considered for revising situation types.

In the next sections we focus on the runtime reasoning performance of our implementations.

7.3. Runtime Reasoning Performance

The tests were run on the sampled test data introduced above on an Intel Core i7-2620 M platform running at 2.7 GHz and utilizing 6 GB RAM. We measured the runtime for each situation type in isolation and for all situation types together, both on object sets of increasing volume. The performance of the relational-DB-backed rule engine turned out to be highly sensitive to the Java garbage collection process. Therefore, we took the median of five repetitions to reject corrupted results (except for the measurement of *all situations*, where we only considered three repetitions due to memory limitations). The results are summarized in Figure 15 and discussed in the paragraphs below.

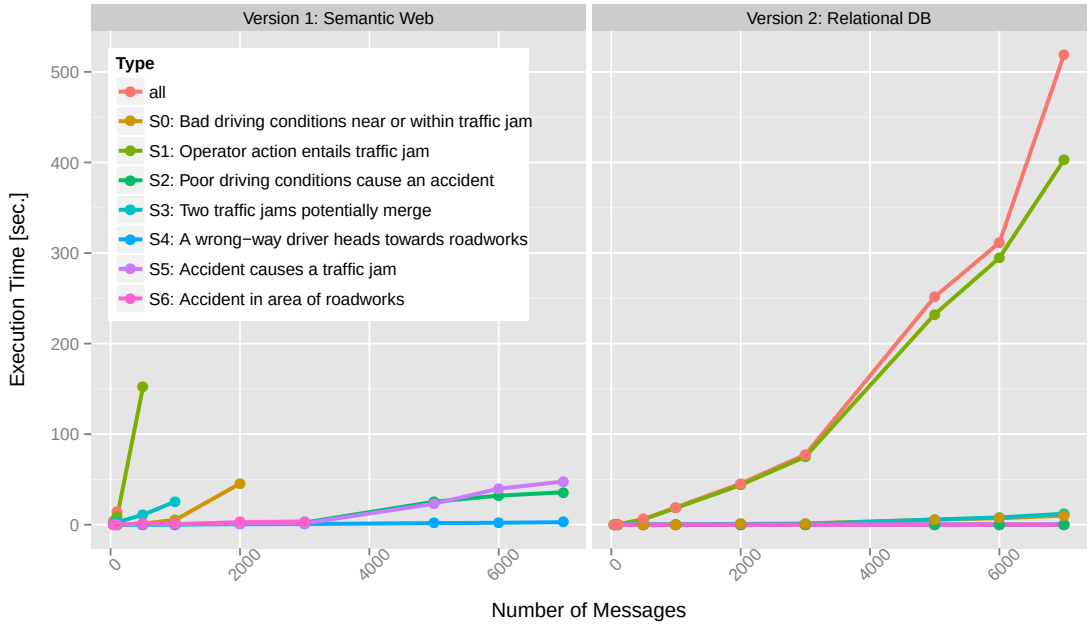


Figure 15: Scalability comparison

Semantic-Web-Based Performance We complement our previous evaluation [10, 72] of the effectiveness of exploiting spatio-temporal knowledge encoded in the ontology on relation disjointness, equivalency, subsumption, symmetry, inverseness, and composition. Although the previous results showed that such knowledge reduces the number of pairwise object comparisons and suffices to scale BeAware to systems of the size of Austrian highways, the performance gains are still unsatisfactory when considering larger systems. Therefore, we measure performance with clustering objects prior to situation assessment. As introduced above, this clustering uses knowledge encoded in relation type definitions to still derive the same situations as without clustering. Specifically, we cluster objects by the highway section they are on, since none of the introduced situation types S_0–S_6 uses spatial conjuncts that span multiple highway sections. Figure 15 shows the performance results with clustering, which outperform our previous results [10]. The missing results for higher numbers of messages (i. e., S0, S1, S3, S6, all) are due to server timeout.

Relational-DB-Backed Performance We evaluate the scalability of the domain-specific situation assessor w.r.t. the number of simultaneously assessed objects. In summary, the runtime increases quadratically in the number of objects, which is not surprising given that the inner loop of situation assessment is pairwise object comparison. On close inspection of the performance of situation types assessed in isolation, it becomes evident that the performance of simultaneously assessing all situation types is completely dominated by a single situation type, notably S_1. The reason for this can be inferred from the definition of the situation type as stated in Table 5, and the distribution of objects as depicted in Figure 14b: since traffic jams and roadworks occur frequently in the domain data set, the rule results in a large number of pairwise object comparisons. Contrary, other situation types involving one or more rarely occurring objects (e. g., wrong-way drivers) can be optimized by placing the rarest object type in the first clause, as already stated above. The main difference to the semantic web implementation is that the domain-specific implementation prunes large portions of the search space early in the rule evaluation, and thus manages to handle much larger object sets. This requires a basic knowledge of the underlying data distribution prior to expressing the rules.

8. Conclusion

In this section, we discuss several lessons learned from applying BeAware in the course of a collab-

orative traffic situation awareness application, and summarize our contributions.

8.1. Lessons Learned

Performance gains through domain-specific implementation. Situation assessment, which is quadratic in the number of objects due to pairwise object comparisons, is a rather expensive operation. A thorough data analysis and carefully crafted rules that prune the search space early in the assessment are key for handling large data sets.

Proper duplicate detection vital for integration of automated and manual data sources. The real-world traffic information indicates that duplicate detection is important to consolidate automated data sources (e. g., traffic jam detectors) with manually entered data (in our case, e. g., provided by a radio station). The most important aspects are (i) to detect duplicates in space-anchored objects evolving over time, (ii) to improve the performance of our preliminary duplicate detection components such that detection on data streams is fully supported, and (iii) to use context information—such as object characteristics that is already present in the projection of future situations—also for duplicate detection purposes. Handling all three points is crucial for operating on data streams: a typical technique on data streams is to reduce the amount of data by using a sliding window. Also, a sliding window allows re-use of previous duplicate detection results to improve performance [100]. Another option is to use object characteristics (e. g., typical duration of traffic jams) to dynamically adjust the size of the sliding window per object type [101], which is especially useful in presence of long-living objects as in road traffic management. For example, it is not uncommon that information about extensive road works (which may last several weeks) must be compared with incoming messages about the same road works from drivers during the whole duration of the road works.

Projection of and reasoning about action impacts. *Projection of action impacts* should make an operator aware of the effects of actions that are possible or should be taken pro-actively in the current situation. Although the execution of actions is non-deterministic in many domains (as the intended effect on the environment cannot always be achieved), projecting possible outcomes can provide a valuable approximation of the consequences. Thus, modeling concepts to define the (likely) effects of actions on the fine-grained level of attributes (e. g., the costs of a particular action) is highly desired. Furthermore, algorithms are needed to enumerate the states of affairs that result thereof.

Another cornerstone for action awareness is better modeling support for capturing the goals of human operators. Interesting work can be found in research on autonomous agents in terms of the beliefs-desires-intention (BDI) model [102]. In our model, the complete knowledge base (A-boxes of perception, comprehension, and projection ontologies) models the beliefs about the current and possible future states of the world, which should be enhanced with additional meta information (e. g., criticality of situations, confidence about a situation, data fusion and projection quality). For example, a human operator could then express goals as desires (e. g., find an action that could reduce traffic congestion); the action planner finds plans to satisfy these desires, and when the human operator chooses one of the plans it becomes an intention.

Multimodal traffic. As a current focus in the research project CSI, multiple control centers of different traffic authorities are integrated on the basis of BeAware, and situations being relevant to those authorities are shared and collaboratively resolved.

Ontology development process. BeAware does not propose a development process itself, since there exist several ontology development processes (e. g., [103, 104, 105]). Recently, platforms to collaboratively develop ontologies and to systematically include user feedback emerged (e. g., [106]). An important distinction to make is which concepts should be shared by many applications (i. e., be considered domain-independent), and which ones should be handled on a per-application-basis; likewise, we want to avoid confusion between concepts and individuals, or between object-level and meta-level concepts [107]. Some orientation with respect to these distinctions is given by ontology design patterns [108], or in ontology design guidelines (e. g., [109]). Another interesting direction is concerned with validation of ontologies with respect to some meta-properties (e. g., OntoClean [110]).

Modeling tools. Modeling tools are desirable to support domain experts in the initial modeling and also the maintenance of a situation awareness application, such as evolution of its situation types. A first step in this direction is a method to create modeling tools from ontologies [111]. For example, the characteristics of relation types in the ontology of BeAware could be used to highlight inconsistencies in situation types, for example between

the relation types Equals of RCC and VeryFar in one situation type, since those relation types are in fact disjoint.

In a recent survey [112], we studied how current SAW systems support maintaining situation awareness. From a systemic point of view, SAW systems development processes and tools should manage (i) the evolution of the observed environment, (ii) the evolution of the SAW system, and (iii) the evolving needs of the operators interacting with the system.

Provably safe traffic actions. Another interesting direction in traffic management concerns automation of traffic control actions, and next-generation traffic control systems with extensive vehicle-to-vehicle and vehicle-to-infrastructure communication, possibly even with autonomous control actions. Such technology, however, would not be particularly useful if it led to incorrect control decisions, possibly even endangering safety on the road. One particularly interesting challenge, thus, is the question of how to ensure correct functioning and reliability of actions. For this, modeling and formal verification techniques are of paramount importance. As a first step in this direction, we formally verified the interaction between traffic centers and autonomous vehicles [113, 114], and developed a modeling tool for hybrid systems [115].

8.2. Summary

To date, operators of large-scale control centers, as encountered in road traffic management, have minimal support for situation awareness. The main problems are: failure to comprehend critical situations in information from multiple data sources, the risk of initiating incorrect actions, and, finally, the inability to project possible futures.

With BeAware, we propose a software framework for developers of situation awareness applications. We described the BeAware prototype that currently implements situation assessment, action assessment, and situation projection. We evaluated its performance on real-world data and gained several lessons learned from applying BeAware in the domain of road traffic management.

References

[1] J. Naisbitt, Megatrends: Ten New Directions Transforming Our Lives, Grand Central Publishing, 1982.

[2] F. Dylla, R. Moratz, Exploiting qualitative spatial neighborhoods in the situation calculus, in: C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, T. Barkowsky (Eds.), Spatial Cognition IV. Reasoning, Action, Interaction, volume 3343 of *LNCS*, Springer Berlin Heidelberg, 2005, pp. 304–322.

[3] A. G. Cohn, J. Renz, Qualitative spatial representation and reasoning, in: F. van Hermelen, V. Lifschitz, B. Porter (Eds.), Handbook of Knowledge Representation, Elsevier, 2008, pp. 551–596.

[4] H. Chen, F. Perich, T. Finin, A. Joshi, SOUPA: Standard ontology for ubiquitous and pervasive applications, in: Mobile and Ubiquitous Systems: Networking and Services, IEEE, 2004, pp. 258–267.

[5] M. M. Kokar, C. J. Matheus, K. Baclawski, Ontology-based situation awareness, International Journal of Information Fusion 10 (2009) 83–98.

[6] N. Baumgartner, W. Retschitzegger, W. Schwinger, Lost in time, space, and meaning—an ontology-based approach to road traffic situation awareness, in: Proc. of the 3rd Workshop on Context Awareness for Proactive Systems (CAPS), CCSR, University of Surrey, Guildford, UK, 2007.

[7] B. Mcguinness, L. Foy, A subjective measure of SA: The crew awareness rating scale (CARS), in: M. R. Endsley, D. B. Kaber (Eds.), Proc. of Human Performance, Situation Awareness and Automation: User-Centered Design for the New Millenium, Savannah, GA, USA, 2000.

[8] M. R. Endsley, Theoretical underpinnings of situation awareness: A critical review, in: M. R. Endsley, D. J. Garland (Eds.), Situation Awareness Analysis and Measurement, Lawrence Erlbaum Associates, New Jersey, USA, 2000, pp. 3–33.

[9] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (2010).

[10] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger, Reasoning on data streams for situation awareness, in: Proc. of the 3rd Int. Conf. on Knowledge Engineering and Ontology Development, INSTICC, 2010, p. 6.

[11] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger, Towards duplicate detection for situation awareness based on spatio-temporal relations, in: Proc. of the 9th Int. Conf. ODBASE, Springer, 2010, pp. 1097–1107.

[12] J. Boyd, The essence of winning and losing, `http://dnipogo.org/john-r-boyd/`, 2010. Last access: 2013/02/07.

[13] J. F. Allen, Maintaining knowledge about temporal intervals, Comm. of the ACM 26 (1983) 832–843.

[14] C. Freksa, Conceptual neighborhood and its role in temporal and spatial reasoning, in: Proc. of the Imacs Int. Workshop on Decision Support Systems and Qualitative Reasoning, North-Holland, 1991, pp. 181–187.

[15] R. Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, The MIT Press, 2001.

[16] I. Horrocks, P. F. Patel-Schneider, Three theses of representation in the semantic web, in: G. Hencsey, B. White, Y.-F. R. Chen, L. Kovács, S. Lawrence (Eds.), Proc. of the 12th Int. WWW Conference, ACM, New York, NY, USA, 2003, pp. 39–47.

[17] B. Motik, On the properties of metamodeling in OWL, Journal of Logic and Computation 17 (2007) 617–637.

[18] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics 6 (2008) 309–322.

[19] E. Rahm, H. H. Do, Data cleaning: Problems and current approaches, IEEE Data Engineering Bulletin 23 (2000) 3–13.

[20] H. Chen, T. Finin, A. Joshi, The SOUPA ontology for pervasive computing, in: V. Tamma, S. Cranefield, T. Finin, S. Willmott (Eds.), Ontologies for Agents: Theory and Experiences, Whitestein Series in Software Agent Technologies, Springer Berlin Heidelberg, 2005, pp. 233–258.

[21] X. H. Wang, D. Q. Zhang, T. Gu, H. K. Pung, Ontology based context modeling and reasoning using OWL, in: A. Tripathi, L. Iftode, K. Nahrstedt, P. Nixon (Eds.), Proc. of the 2nd Annual Conference on Pervasive Computing and Communications, Workshops, IEEE, 2004, pp. 18–22.

[22] L. Ardissono, R. Furnari, A. Goy, G. Petrone, M. Segnan, A framework for the management of context-aware workflow systems, in: J. Filipe, J. Cordeiro, B. Encarnação, V. Pedrosa (Eds.), Proc. of the 7th Int. Conf. on Web Information Systems and Technologies, INSTICC Press, 2007, pp. 1870–1875.

[23] M. Heravizadeh, D. Edmond, Making workflows context-aware: A way to support knowledge-intensive tasks, in: A. Hinze, M. Kirchberg (Eds.), Proc. of the 5th Asia-Pacific Conf. on Conceptual Modelling, Australian Computer Society, 2008, pp. 79–88.

[24] J. Li, Y. Bu, S. Chen, X. Tao, J. Lu, FollowMe: On research of pluggable infrastructure for context awareness, in: Proc. of the 20th Int. Conf. on Advanced Information Networking and Applications, IEEE, 2006, pp. 199–204.

[25] K. Shin, Y. Cho, J. Choi, C.-W. Yoo, A workflow language for context-aware services, in: S. soo Kim, J. H. Park, N. Pissinou, T. hoon Kim, W. C. Fang, D. Slezak, H.-R. Arabnia, D. Howard (Eds.), Proc. of the Int. Conf. on Multimedia and Ubiquitous Engineering, IEEE, 2007, pp. 1227–1232.

[26] D. Rajpathak, E. Motta, An ontological formalization of the planning task, in: A. C. Varzi, L. Vieu (Eds.), Proc. of the Int. Conf. on Formal Ontology in Information Systems, IOS Press, 2004, pp. 305–316.

[27] Y. Gil, J. Blythe, PLANET: A shareable and reusable ontology for representing plans, in: Proc. of the AAAI 2000 Workshop on Representational Issues for Real-World Planning Systems, AAAI Press, 2000, pp. 28–33.

[28] L. Morgenstern, D. Riecken, SNAP: An action-based ontology for e-commerce reasoning, in: Proc. Formal Ontologies Meet Industry, 2005, pp. 1–12.

[29] A. Schwering, Evaluation of a semantic similarity measure for natural language spatial relations, in: S. Winter, M. Duckham, L. Kulik, B. Kuipers (Eds.), Spatial Information Theory, volume 4736 of *LNCS*, Springer Berlin Heidelberg, 2007, pp. 116–132.

[30] A. Schwering, Approaches to semantic similarity measurement for geo-spatial data: A survey, Transactions in GIS 12 (2008) 5–29.

[31] M. Weis, F. Naumann, U. Jehle, J. Lufter, H. Schuster, Industry-scale duplicate detection, Proceedings of the VLDB Endowment 1 (2008) 1253–1264.

[32] J. Bleiholder, F. Naumann, Data fusion, ACM Comput. Surv. 41 (2008) 1–41.

[33] A. Bilke, J. Bleiholder, C. Böhm, K. Draba, F. Naumann, M. Weis, Automatic data fusion with HumMer., in: K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, B. C. Ooi (Eds.), Proc. of the 31st Int. Conf. on Very Large Data Bases, ACM, 2005, pp. 1251–1254.

[34] T. Strang, C. Linnhoff-Popien, A context modeling survey, in: Proc. of the 1st Int. Workshop on Advanced Context Modelling, Reasoning and Management, Springer, 2004, p. 8.

[35] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, L. Tanca, A data-oriented survey of context models, SIGMOD Rec. 36 (2007) 19–26.

[36] M. Wieland, O. Kopp, D. Nicklas, F. Leymann, Towards context-aware workflows, in: Proc. of the CAiSE'07 Workshops and Doctoral Cons., Springer, 2007, pp. 1–15.

[37] A. Haller, E. Oren, P. Kotinurmi, An ontology for internal and external business processes, in: Proc. of the 15th Int. Conf. on World Wide Web, ACM Press, 2006, pp. 1055–1056.

[38] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger, "Same, Same but Different"—A Survey on Duplicate Detection Methods for Situation Awareness, in: Proc. of the 8th Int. Conf. ODBASE, Springer, 2009, pp. 1050–1068.

[39] S. Mitsch, A. Müller, W. Retschitzegger, A. Salfinger, W. Schwinger, A survey on clustering techniques for situation awareness, in: To appear in Proc. of the 1st Int. Workshop on Management of Spatial Temporal Data, Springer, 2013.

[40] N. Baumgartner, W. Retschitzegger, A survey of upper ontologies for situation awareness, in: Proc. of the 4th Int. Conf. on Knowledge Sharing and Collaborative Engineering, ACTA Press, 2006, pp. 1–9.

[41] R. T. Snodgrass, Developing Time-Oriented Database Applications in SQL, Morgan Kaufmann Publishers, Inc., USA, 1999.

[42] J. Bateman, S. Farrar, Spatial Ontology Baseline, Internal report, Collaborative Research Center for Spatial Cognition, 2004.

[43] I. Niles, A. Pease, Towards a standard upper ontology, in: C. Welty, B. Smith (Eds.), Proc. of the 2nd Int. Conf. on Formal Ontology in Information Systems, ACM, 2001, pp. 2–9.

[44] B. Krieg-Brückner, U. Frese, K. Lüttich, C. Mandel, T. Mossakowski, R. Ross, Specification of an ontology for route graphs, in: Spatial Cognition IV: Reasoning, Action, and Interaction, Springer, 2005, pp. 390–412.

[45] J. R. Hobbs, F. Pan, An ontology of time for the semantic web, ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing 3 (2004) 66–85.

[46] W. M. P. van der Aalst, A. H. M. ter Hofstede, Yawl: Yet another workflow language, Information Systems 30 (2005) 245–275.

[47] D. Gasevic, V. Devedzic, Petri net ontology, Journal of Knowledge-Based Systems 19 (2006) 220–234.

[48] P. Haase, B. Motik, A mapping system for the integration of OWL-DL ontologies, in: Proc. of the 1st Int. Workshop on Interoperability of Heterogeneous Information Systems, ACM, 2005, pp. 9–16.

[49] J. Euzenat, Algebras of ontology alignment relations, in: Proc. of the 7th Int. Conf. on The Semantic Web, Springer, Heidelberg, 2008, pp. 387–402.

[50] F. Scharffe, J. Euzenat, D. Fensel, Towards design patterns for ontology alignment, in: Proc. of the 23rd Annual Symp. on Applied Computing, ACM, 2008,

pp. 2321–2325.
[51] M. Wimmer, G. Kappel, A. Kusel, W. Retschitzegger, J. Schönböck, W. Schwinger, Surviving the heterogeneity jungle with composite mapping operators, in: Proc. of the 3rd Int. Conf. on Model Transformation, Springer, 2010, pp. 260–275.
[52] M. Wimmer, G. Kappel, A. Kusel, W. Retschitzegger, J. Schönböck, W. Schwinger, Towards an expressivity benchmark for mappings based on a systematic classification of heterogeneities, in: Proc. of the 1st Int. Workshop on Model-Driven Interoperability, ACM, 2010, pp. 32–41.
[53] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger, Improving situation awareness in traffic management, in: Proc. of the 8th Int. Workshop on Quality in Databases, ACM, 2010, p. 5.
[54] J. Barwise, J. Perry, Situations and Attitudes, MIT Press, 1983.
[55] H. Rolka, J. C. O'Connor, D. Walker, Public health information fusion for situation awareness, in: Proc. Workshop BioSecure, Springer, 2008, pp. 1–9.
[56] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, Revisiting the JDL data fusion model II, in: Proc. of the 7th Int. Conf. on Information Fusion, International Society of Information Fusion, 2004, pp. 1218–1230.
[57] J. Salerno, M. Hinman, D. Boulware, P. Bello, Information fusion for situational awareness, in: Proc. Information Fusion, ISIF, 2003, pp. 507–513.
[58] D. A. Lambert, A blueprint for higher-level fusion systems, Information Fusion 10 (2009).
[59] A. Gangemi, P. Mika, Understanding the semantic web through descriptions and situations, in: Proc. of the Int. Conf. on Ontologies, Databases and Applications of Semantics, Springer, 2003, pp. 689–706.
[60] C. Matheus, M. Kokar, K. Baclawski, J. Letkowski, C. Call, M. Hinman, J. Salerno, D. Boulware, SAWA: An assistant for higher-level fusion and situation awareness, in: Proc. of SPIE Conf. on Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications, SPIE, 2005, pp. 75–85.
[61] P. R. Smart, A. Russell, N. R. Shadbolt, M. C. Schraefel, L. A. Carr, AKTiveSA: A technical demonstrator system for enhanced situation awareness in military operations other than war, The Computer Journal 50 (2007) 703–716.
[62] S. S. Yau, J. Liu, Hierarchical situation modeling and reasoning for pervasive computing, in: Proc. of the 3rd Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, IEEE Computer Society, 2006, pp. 5–10.
[63] K. Sycara, R. Glintona, B. Yu, J. Giampapa, S. Owens, M. Lewis, C. Grindle, An integrated approach to high-level information fusion, Information Fusion 10 (2009) 25–50.
[64] C. J. Matheus, M. M. Kokar, K. Baclawski, J. Letkowski, An application of semantic web technologies to situation awareness, in: Proc. of the Int. Semantic Web Conference, Springer Berlin Heidelberg, 2005, pp. 944–958.
[65] A. G. Cohn, B. Bennett, J. M. Gooday, N. Gotts, RCC: A calculus for region based qualitative spatial reasoning, GeoInformatica 1 (1997) 275–316.
[66] F. Dylla, J. O. Wallgrün, On generalizing orientation information in OPRA$_m$, in: Proc. of the 29th Annual German Conf. on AI, Springer, 2007, pp. 274–288.
[67] M. Ragni, S. Wölfl, Temporalizing cardinal directions: From constraint satisfaction to planning, in: Proc. of 10th Int. Conf. on Principles of Knowledge Representation and Reasoning, AAAI Press, 2006, pp. 472–480.
[68] D. Hernández, E. Clementini, P. D. Felice, Qualitative distances, in: Proc. of the Int. Conf. COSIT '95, Springer, 1995, pp. 45–57.
[69] O. Raiman, Order of magnitude reasoning, Artificial Intelligence 51 (1991) 11–38.
[70] C. Freksa, Temporal reasoning based on semi-intervals, Artificial Intelligence 54 (1992) 199–227.
[71] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, W3C member submission, SWRL: A semantic web rule language, `http://www.w3.org/Submission/SWRL`, 2004.
[72] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger, BeAware!—Situation awareness, the ontology-driven way, Journal of Data and Knowledge Engineering 69 (2010) 1181–1193.
[73] J. Holsopple, S. J. Yang, FuSIA: Future situation and impact awareness, in: Proc. of the 11th Int. Conf. on Information Fusion, IEEE, 2008, pp. 1–8.
[74] Z. Cui, A. G. Cohn, D. A. Randell, Qualitative simulation based on a logical formalism of space and time, in: Proc. AAAI-92, AAAI Press, 1992, pp. 679–684.
[75] J. M. Gooday, A. G. Cohn, Transition-based qualitative simulation, in: Proc. of the 10th Int. Workshop on Qualitative Reasoning, AAAI Press, 1996, pp. 74–82.
[76] A. G. Cohn, N. M. Gotts, D. A. Randell, Z. Cui, B. Bennett, J. M. Gooday, Exploiting temporal continuity in qualitative spatial calculi, in: Spatial and Temporal Reasoning in Geographical Information Systems: Report on the Specialist Meeting on Time in Geographic Space, Oxford University Press, 1998, p. 25.
[77] E. Davis, Qualitative reasoning and spatio-temporal continuity, in: S. M. Hazarika (Ed.), Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions, IGI Global, 2012, p. 50.
[78] N. van de Weghe, P. D. Maeyer, Conceptual neighborhood diagrams for representing moving objects, in: Proc. of the ER Workshop on Perspectives in Conceptual Modeling, Springer, 2005, pp. 228–238.
[79] F. Dylla, J. O. Wallgrün, Qualitative spatial reasoning with conceptual neighborhoods for agent control, Journal of Intell. Robotics Systems 48 (2007) 55–78.
[80] K. R. Apt, S. Brand, Constraint-based qualitative simulation, in: Proc. of the 12th Int. Symp. on Temporal Representation and Reasoning, IEEE, 2005, pp. 26–34.
[81] M. Bhatt, Reasoning about Space, Actions and Change, in: S. M. Hazarika (Ed.), Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions, IGI Global, 2012, p. 52.
[82] P. Mateus, A. Pacheco, J. Pinto, A. Sernadas, C. Sernadas, Probabilistic situation calculus, Annals of Math. and Artificial Intelligence 32 (2001) 393–431.
[83] D. Fava, J. Holsopple, S. J. Yang, B. Argauer, Terrain and behavior modeling for projecting multistage cyber attacks, in: Proc. of the 10th Int. Conf. on Information Fusion, IEEE, 2007, pp. 1–7.
[84] M. Sabry, H. Abd-El-Latif, S. Yousef, N. Badra, A time-series forecasting of average daily traffic volume, Australian Journal of Basic and Applied Sciences 1 (2007) 386–394.
[85] S. Sun, C. Zhang, G. Yu, A bayesian network approach to traffic flow forecasting, IEEE Trans. on Intelligent

Transportation Systems 7 (2006) 124–132.
[86] B. Kuipers, Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge, MIT Press, 1994.
[87] G. Ligozat, Towards a general characterization of conceptual neighborhoods in temporal and spatial reasoning, in: Proc. of the AAAI-94 Workshop on Spatial and Temporal Resoning, AAAI, 1994, pp. 55–59.
[88] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger, On optimization of predictions in ontology-driven situation awareness, in: Proc. of the 3rd Int. Conf. on Knowledge, Science, Engineering and Management, Springer Berlin Heidelberg, 2009, pp. 297–309.
[89] M. Bhatt, W. Rahayu, G. Sterling, Qualitative simulation: Towards a situation calculus based unifying semantics for space, time and actions, in: Spatial Information Theory (Doctoral Colloqium), Springer, 2005, p. 6.
[90] A. D. Sarma, O. Benjelloun, A. Halevy, S. Nabar, J. Widom, Representing uncertain data: Models, properties, and algorithms, VLDB Journal (2009).
[91] N. Baumgartner, W. Retschitzegger, W. Schwinger, G. Kotsis, C. Schwietering, Of situations and their neighbors—Evolution and Similarity in Ontology-Based Approaches to Situation Awareness, in: Proc. of the 6th Int. and Interdisc. Conf. on Modeling and Using Context, Springer, 2007, pp. 29–42.
[92] S. Mitsch, W. Retschitzegger, W. Schwinger, Towards modeling dynamic behavior with integrated qualitative spatial relations, in: Proc. of the 5th Int. Workshop on Semantic and Conceptual Issues in GIS, Springer, 2011, pp. 271–280.
[93] A. Galton, Continuous motion in discrete space, in: Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 2000, pp. 26–37.
[94] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, W. Schwinger, Situation prediction nets—playing the token game for ontology-driven situation awareness, in: Proc. of the 29th Int. Conf. on Conceptual Modeling, Springer, 2010, pp. 202–218.
[95] K. Jensen, L. M. Kristensen, L. Wells, Coloured petri netsand cpn tools for modeling and validation of concurrent systems, Journal on Software Tools for Technology Transfer 9 (2007) 213–254.
[96] N. Baumgartner, W. Retschitzegger, W. Schwinger, A software architecture for ontology-driven situation awareness, in: Proc. of the 23rd Annual Symp. on Applied Computing, ACM, 2008, pp. 2326–2330.
[97] M. Westergaard, L. M. Kristensen, The Access/CPN framework: A tool for interacting with the CPN tools simulator, in: Applications and Theory of Petri Nets, Springer, 2009, pp. 313–322.
[98] R. Chrobok, O. Kaumann, J. Wahle, M. Schreckenberg, Three categories of traffic data: Historical, current, and predictive, in: Proc. of the 9th Int. Fed. Automatic Control Symposium: Control in Transporation Systems, International Federation of Automatic Control, 2000, pp. 250–255.
[99] J. Holsopple, S. J. Yang, Designing a data fusion system using a top-down approach, in: Proc. of the Military Communications Conf., IEEE, 2009, pp. 1–7.
[100] U. Draisbach, F. Naumann, A generalization of blocking and windowing algorithms for duplicate detection, in: Data and Knowledge Engineering (ICDKE), 2011 International Conference on, IEEE, 2011, pp. 18–24.
[101] U. Draisbach, F. Naumann, S. Szott, O. Wonneberg, Adaptive windows for duplicate detection, in: A. Kementsietsidis, M. A. V. Salles (Eds.), ICDE, IEEE Computer Society, 2012, pp. 1073–1083.
[102] M. Georgeff, B. Pell, M. Pollack, M. Tambe, M. Wooldridge, The belief-desire-intention model of agency, in: J. Mller, A. Rao, M. Singh (Eds.), Intelligent Agents V: Agents Theories, Architectures, and Languages, volume 1555 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 1999, pp. 1–10.
[103] M. Uschold, M. King, Towards a methodology for building ontologies, in: In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95, AIAI-TR-183, University of Edinburgh, 1995.
[104] T. R. Gruber, Toward principles for the design of ontologies used for knowledge sharing?, Int. J. Hum.-Comput. Stud. 43 (1995) 907–928.
[105] S. Staab, R. Studer, Handbook on Ontologies, International Handbooks on Information Systems, Springer, 2nd edition, 2009.
[106] P. R. Alexander, C. Nyulas, T. Tudorache, P. L. Whetzel, N. F. Noy, M. A. Musen, Semantic infrastructure to enable collaboration in ontology development, in: W. W. Smari, G. Fox (Eds.), CTS, IEEE, 2011, pp. 423–430.
[107] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, L. Schneider, Sweetening ontologies with DOLCE, in: Proc. of the 13th Int. Conf. on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web (EKAW), Springer, 2002, pp. 166–181.
[108] V. Presutti, E. Blomqvist, E. Daga, A. Gangemi, Pattern-based ontology design, in: M. del Carmen Suárez-Figueroa, A. Gómez-Pérez, E. Motta, A. Gangemi (Eds.), Ontology Engineering in a Networked World, Springer, 2012, pp. 35–64.
[109] A. Scherp, C. Saathoff, T. Franz, S. Staab, Designing core ontologies, Applied Ontology 6 (2011) 177–221.
[110] N. Guarino, C. A. Welty, An overview of OntoClean, in: S. Staab, R. Studer (Eds.), Handbook on Ontologies, International Handbooks on Information Systems, Springer, 2004, pp. 151–172.
[111] A. Kusel, S. Mitsch, W. Retschitzegger, W. Schwinger, R. Mayr, J. Schönböck, Ontology-driven generation of multi-view modeling tools, in: Software Engineering, IASTED, 2012, pp. 45–51.
[112] A. Salfinger, W. Retschitzegger, W. Schwinger, Maintaining situation awareness over time—A survey on the evolution support of situation awareness systems, in: Conf. on Technologies and Applications of Artificial Intelligence (TAAI), IEEE, 2013.
[113] S. Mitsch, S. M. Loos, A. Platzer, Towards formal verification of freeway traffic control, in: Proc. of the 3rd Int. Conf. on Cyber-Physical Systems, IEEE/ACM, 2012, pp. 171–180.
[114] S. Mitsch, K. Ghorbal, A. Platzer, On provably safe obstacle avoidance for autonomous robotic ground vehicles, in: Robotics: Science and Systems, Springer, 2013.
[115] S. Mitsch, G. O. Passmore, A. Platzer, A vision of collaborative verification-driven engineering of hybrid systems, in: M. Kerber, C. Lange, C. Rowat (Eds.), Do-Form, AISB, 2013, pp. 8–17.