A Model for Ontology-Based Scene Description for Context-Aware Driver Assistance Systems

Simone Fuchs, Stefan Rass, Bernhard Lamprecht, Kyandoghere Kyamakya
Transportation Informatics Group
Department of Smart System-Technologies
University of Klagenfurt
{firstname.lastname}@uni-klu.ac.at

ABSTRACT

Driving assistance systems (DAS) offer support in potentially dangerous situations, especially for unexperienced drivers. Co-operative systems improve their performance by sharing information with each other. One key-enabler for describing and exchanging context between intelligent vehicles, which use it for reasoning about their environment, is a common context-model. In this paper, we briefly discuss the influence of the driving context on decision-making and present an OWL-based context-model for abstract scene representation of driving scenarios. We further outline the integration of scene-descriptions with a logic-based reasoning system, based on a set of transformation rules.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalism and Methods; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding

1. INTRODUCTION

One of the key-enablers for bringing context-awareness to collaborative driver assistance systems (DAS) is a common domain representation for information exchange. By context we mean "context" in the understanding of "situation-awareness", as defined by Endsley [5] and cited by [1] as "*... the perception of elements in the environment within a span of space and time, the comprehension of their meaning and the projection of their status in the near future*". So, context for a driving assistance system refers to the driving situation, consisting of the environment and all objects and traffic participants within it, which are currently relevant to the own vehicle. Additionally, the driver, the own vehicle's state and the national driving regulations are part of the context.

Current approaches of DAS (e.g. ACC - adaptive cruise control) are mostly stand-alone solutions, focusing on a highly specialized sub-task, with limited context-awareness. For the future - with the ongoing development in the fields of machine-vision and sensor technology - we expect integration of stand-alone sub-solutions to take place, thus resulting in smarter DASs. The overall driving context will become important for correctly recognizing and interpreting complex driving situations. DAS will become increasingly knowledge-based and methods will be needed for modeling and handling the vast amount of context information. A context-model serves as common domain description and is necessary to represent context information and to exchange it between collaborating vehicles and infrastructure. Collaboration allows DAS to achieve better performance in supporting the driver in difficult situations, simply because it provides *additional information* about a vehicle's surrounding. An abstract model of this information is the basis for a reasoning process that deduces driving recommendations.

We start the paper with a discussion of related work. We then briefly describe the driving context and present an ontology-based context-model for representation of driving scenarios. Afterwards we show the integration of the model into a constraint-based reasoning process and conclude with a discussion of the proposed approach.

Ambi-sys'08 February 11-14, 2008, Quebec, Canada

2. RELATED WORK

2.1 Context-Modeling

Several methods have been introduced for context modeling. [9] proposed the Context Modeling Language (CML). CML is a graphical modeling approach extending the Object-Role Modeling Language (ORM[1]). Because of the closeness between ORM and relational algebra, CML can be used to directly transform a context-model to an underlying relational database. CML provides situation abstraction to define high-level context using a form of predicate logic. Situations can be arbitrarily combined to reuse and model more complex situations. The model has been recently extended to support unknown, ambiguous, imprecise and erroneous context-data.

[21] present a context-modeling profile (CMP). The CMP approach uses meta-model extension capabilities provided by the Unified Modeling Language (UML) - stereotypes, tagged values and Object Constraint Language (OCL). CMP graphically supports context meta-information, especially for source and validity of context-information, as well as for privacy managment. CMP is a lightweight extension of the UML meta-model, meaning it extends the existing meta-model without changing it. Lightweight profiles have the

[1] http://www.orm.net

advantage of being supported by a great number of UML modeling tools, whereas heavyweight profiles, which change the UML meta-model [20], are not.
Context-Modeling has also been done using the Semantic Web resp. the Ontology Web Language (OWL[2]). [4] presentes COBRA-ONT, a collection of ontologies formulated in OWL for pervasive context-aware systems. Further examples for OWL-based approaches are [22], [7] and [26]. OWL has the main advantage that simple reasoning about the structure of designed ontologies is possible without necessary transformation to other representations (e.g. description logics). The reasoning capabilities of OWL are limited to concepts like inheritance, transitivity between OWL-classes, symmetry etc.
[23] provides a survey of the most common context modeling approaches, based on a set of previously identified requirements. The authors conclude that ontology-based approaches are most promising for context modeling in ubiquitous environments, since they outperform the other approaches in all points.

2.2 Context-Modeling for DAS

In [24], Toennis et al present their context-aware deduction system, called SCORE (Spatial Context Ontology Reasoning Environment). SCORE consists of separate autonomous components that distribute context knowledge and reason about spatial properties of the contextual data. SCORE uses low-level spatial information together with ontological context-representation for high-level context-processing. However, it remains unclear which context objects the ontology contains and how their spatial information and relationships are actually represented (qualitative vs. quantitative, egocentric vs. allocentric etc.). It is also unclear which traffic objects the ontology contains (vehicles only, other participants, etc.) A rule-based approach derives information from the gathered spatial data. Spatial queries like "Is car X overtaking car Y?" can be processed by the system, but the authors do not explain the content of their rule base or reasoning mechanisms behind it. Therefore, in our opinion, the modeling capabilities and reasoning power of the approach remain undemonstrated. The authors further claim that their rule base handles spatial regulations from traffic rules, but it is not clear how these are incorporated into the knowledge base. Also, the approach seems to consider neither temporal concepts nor uncertain information.
[18] proposes a framework for modeling and predicting driver behavior using Dynamic Belief Networks. The authors state that a driver's behavior is evolving dynamically within a certain situation. Context is necessary to explain driving behavior and to improve generalizability and reliability of behavior models. The framework uses Bayesian networks and conditional probability tables to represent contextualized driver actions. For integration of time dependencies the authors suggest using Dynamic Belief Networks. The presented approach focuses on predicting driver behavior, which is of course important, but only a small part of the overall driving environment. Concepts with substantial influence onto the driver's behavior are left out, e.g. spatial relationships between vehicles, environmental conditions or traffic regulations. The question remains if the approach will scale well for modeling more complicated scenarios.
In [15, 13] the authors present a knowledge-based approach for modeling spatio-temporal driving situations using qualitative motion descriptions. Direction, speed and distance are mapped from quantitative data into qualitative classes for high-level abstraction. A rule-base reasons on the qualitative scene descriptions to deduce driving behavior. The authors clearly demonstrate the appropriateness of qualitative scene representation for successful reasoning about spatio-temporal patterns of moving objects. Beside abstraction to qualitative classes, no further concepts for dealing with imprecise and uncertain data are introduced. However, this would be important, since the approach is meant for the operational level of autonomous vehicles. Further, it seems questionable if qualitative classes are the best solutions for all parameters of driving patterns. For example, to reason about an overtaking situation it seems more feasible to use absolute speed and distance values to determine the necessary speed difference and time-frames for the overtaking maneuver. The presented rule-based focuses solely on spatio-temporal reasoning, further influence factors on the driving task that where already mentioned are not taken into account.
[2] presents an ontology for representing road traffic situations in a traffic management center, involving representation and reasoning for qualitative spatio-temporal patterns. The applicability of the developed ontology to the recognition of road traffic management scenarios is demonstrated. Situations that need the attention of the human supervisor are recognized and pointed out, e.g. an accident in front of a traffic jam inside a tunnel. The presented ontology is kept very simple and tailored to the needs of the traffic management task, but it demonstrates the applicability of ontology-based techniques to context-modeling outside the semantic web. The spatio-temporal concepts for traffic management are similar to those needed within DAS and the presented solution has potential for reuse, but besides the two problem fields have little in common.
It seems that most of the current modeling approaches focus on the spatio-temporal representation. None of them has tried to extend and include additional relevant information, like traffic objects, environmental conditions and traffic rules. The remainder of the paper briefly motivates the need for representing the overall context within a DAS and presents our approach for a more comprehensive context-model for scene description of driving situations, which extends around the already well-researched spatio-temporal representation.

3. A CONTEXT-MODEL FOR DRIVING SCENE REPRESENTATION

The overall context for a driving situation consists of four major sub-contexts:

1. the operating environment of the vehicle and all relevant objects within it,
2. the driver,
3. the vehicle with the built-in DAS (own vehicle) and
4. national traffic regulations.

These four contexts have been identified in most projects concerned with intelligent DAS (examples are [14], [3], [25]). The most complex context with the greatest variety of participating objects is the **environment**. By "spatial" context

[2]http://www.w3.org/2004/OWL/

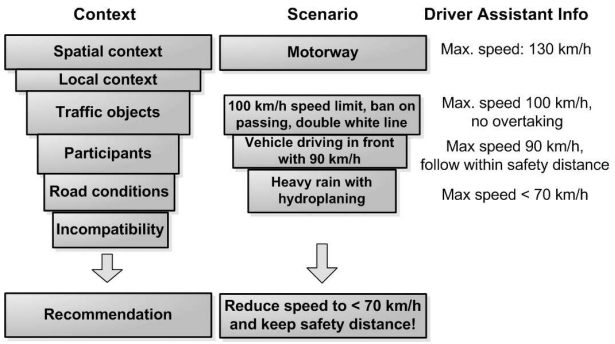

Figure 1: Example: How does context-awareness influence decisions of a DAS?

we refer to the type of the road the own vehicle is currently driving on. The "local" context is a regional physical environment where special driving rules apply and it is located within a spatial context. Examples for a local context are intersections, level crossings, tunnel, crosswalks etc. Traffic objects like signs, pedestrians, markings, etc. are located within and valid for a spatial or local context, respectively. Representing the relationship between traffic objects and their spatial/local context improves the scene-analysis and reasoning process, because non-likely objects and scenarios can be excluded in advance. Road conditions complement the driving environment.

The **driver-context** comprises the current state, experience, the risk-willingness - using qualitative classes - and also the driver's intent for the next planned manoeuvre. For the **own vehicle**, in addition to standard-parameters like speed, built-in safety systems must be considered, because recommended driving behavior differs with their presence/absence.

Driving regulations also have substantial influence on the recommended behavior and have to be incorporated in the reasoning process.

We take detection of traffic objects as well as driver states with sensor systems for granted, as there are already successful research projects in this field. In traffic object detection exemplary tasks are pedestrian recognition [19, 6] and traffic sign recognition [16]. Projects concerned with driver state detection are [8, 17, 25].

Figure 1 shows the influence of all four context parts on the recommendation given to the driver by a DAS. The spatial context (type of road) dictates the applicable standard driving rules under optimal conditions. Any other object within the context presents an additional constraint to the given standard behavior. In the example, under best conditions the DAS would only monitor the maximum speed limit. If a front-driving car with a speed of 90 km/h appears, the DAS can give the recommendation to overtake, depending on the driver's preferences (defensive or offensive behavior). If further scene analysis detects a ban-of-passing and a 100 km/h speed limit, the DAS deduces to follow the car within safety-distance. With additional bad weather conditions the system may even propose a lower speed. The last step in the reasoning hierarchy - "Incompatibility" - is used for determining unforeseen circumstances, which are conflicting with the current context (e.g. a pedestrian on a highway) and demand immediate attention.

This simple example shows the high influence context-awareness has on an intelligent DAS. Still, as we have discussed in the previous section, most present-day projects only focus on certain small parts of the context. The example further shows that it is feasible to analyze driving situations step-wise in a hierarchical fashion, reducing the number of possible decisions. Reasoning starts with determining the standard behavior. Additional objects detected throughout the hierarchy impose further restrictions.

In the next section, we show an ontology-based model for textual driving scene representation, containing information from the overall context.

3.1 The Context-Model

We developed an OWL-based context-model for representing information about traffic objects and relationships relevant for the driving task. We started with a comprehensive UML-model as a design base and transformed it to an ontology, utilizing the OWL-modeling-tool "Protégé". OWL was finally chosen over UML to facilitate the common information representation and exchange process between collaborating vehicles. Since OWL was developed with the intent of describing context to software agents, it is a suitable representation language for information sharing (cf. [22, 23]). Modeled objects can be processed, exchanged with other systems and transformed into a variety of other languages for further processing with reasonable effort. Figure 2 shows an overview of our driving ontology.

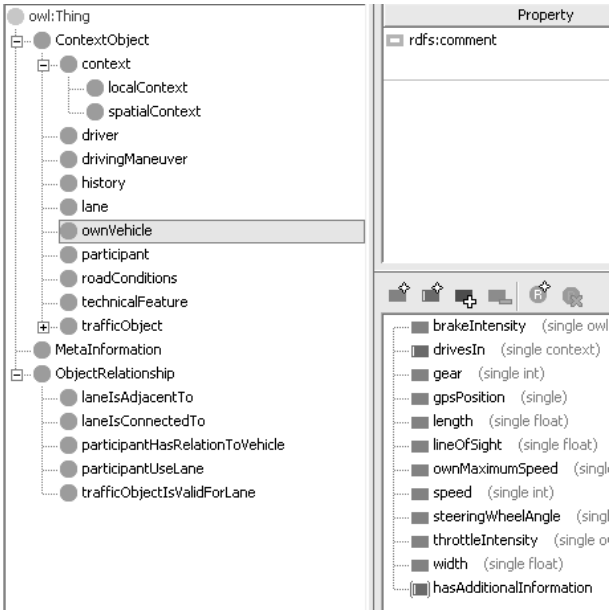

Figure 2: Driving scenario ontology for scene representation

As we can see immediately, the number of possible objects participating in a driving context is comparatively low; the main challenge for the reasoning system arises from the extensive number of possible object-combinations within a traffic scene. The ontology was developed with the main intention of providing a framework for description of traffic scenarios based on the egocentric view of the own vehicle, independent of a certain task (be it overtaking, turning or

holding distance to a vehicle). Abstract scene representations can be exchanged between collaborating vehicles and infrastructure and used for reasoning on a tactical level. For the operational level (autonomous driving), the model is easily extendable using the OWL super/subclass structure. Every object can be further detailed with additional information, without changing the high-level representation.
The classes in the ontology represent traffic objects from the four context categories, together with their attributes (datatype properties), one-to-one relationships (object properties) and many-to-many relationships (classes). Appropriate domains and ranges, as well as cardinality restrictions, are assigned to each property.
The class "trafficObject" is further extended into a more detailed hierarchy with the subclasses *marking, trafficLight* and *trafficSign*. Each of those is a generic traffic object, valid for a specific number of lanes and restricting driving maneuver. However, each object has specific properties, not shared by the others: a traffic light has a certain state (e.g. red) and is valid for specific directions of a lane (e.g. straight and right-turn), whereas a traffic sign has a specific name, additional add-on signs etc. The "traffic sign" class is again extended into *danger, information* and *order*, deploying different behavior during the reasoning process.
Modeling of separate lanes of the street network was done in the style of the TeleAtlas©lane representation, in order to facilitate future integration with their digital map data. Two Lanes are connected with each other if the are reachable by driving maneuver. Additionally, an adjacency relationship is maintained.
To represent noisy and imprecise data, information about an object's quality is included. In our model, we have a special class "Meta-Information" that is related to each context-object and object-relationship. This class contains the name and reliability of each source, a quality-measurement and a time-span for the object's validity. The list of meta-attributes and their allowed values can be arbitrarily extended as needed.
For modeling the spatio-temporal data, the model was inspired by existing approaches, discussed in section 2.2, especially [15] and [13]. The meta-property "time-span" is used to integrate the notion of time for moving objects. It gives the time-interval for which the current object-relationship-state is going to be valid. [15] shows that a single time-interval is sufficient to represent high-level motion description. The spatial relationship attributes "direction" and "directionMovement" defining the relative relation between the own vehicle and other participants are represented using quality classes with finite domains, e.g. direction domain = {*front, frontLeft, frontRight, left, right, rear, rearLeft, rearRight*}. However, for distance, speed measures and line of sight we vote for numerical values. They are comparatively easy to obtain from various sensing systems (unlike e.g. driver state) and support for many difficult driving maneuvers, like overtaking, relies heavily on the calculation of necessary time-frames and speed/distance differences, which need numerical values. Although humans apparently cope with this situations using estimated measures all the time, one should keep in mind that most accidents related to e.g. overtaking are due to underestimation of the parameters speed and distance.
Complex relationships between traffic objects are modeled with separate classes (as mentioned above), assigned to the superclass "ObjectRelationships" to distinguish them from context objects. Some relationships consist solely of object properties, referencing to the participating context-object classes, e.g. "trafficObjectIsValidForLane" has object properties for the traffic object's id and the lane's id. If a traffic object is valid for more than one lane, separate class instances have to be created for every lane. This facilitates the reasoning process, where the lane is usually given and the question is, which objects are valid for this lane at the moment. Other relationship classes additionally have datatype properties, to represent association attributes for a relationship between context objects. The class "participantUseLane", for example, has additional properties for the remaining space on the left and right of the participant with respect to the lane's width.
Further context-objects, attributes and relationships with influence on the decision process are included in the model. Examples are the driver, road conditions, technical features of the own vehicle and spatial/local context, described in the previous section.
It is not possible to discuss every aspect of the context-model within this paper, therefore we refer to the complete ontology, published on our homepage[3]. The ontology is restrained to the representation of traffic scene descriptions. Because OWL has no framework yet for the intuitive representation of complex and sophisticated rules [10, 11], a logic-based approach is used for representation of traffic regulations. The next section describes the process of transferring ontology-based scene description to the logic knowledge base.

3.2 Mapping from OWL to ECLiPSe

As already mentioned, OWL does not support representation of complex reasoning rules. We therefore decided to represent traffic rules with a logic-based approach: as user-defined constraints within the open-source prolog-based logic programming environment "ECLiPSe". We expect good results from using constraint logic programming (CLP) for the reasoning component. Few decision variables together with a countless number of possible combinations is a common situation in CLP and a number of efficient algorithms exist to cope with the resulting difficulties.
For representing scenarios in ECLiPSE, we introduced a transformation step: each OWL-based scenario description is translated to ECLiPSe syntax and fed into the knowledge base as a collection of dynamic facts. Translation is rather straightforward, since ECLiPSE allows for abstract object representation by utilizing structures, a concept similar to a C/C++-Struct. An ECLiPSe structure is a special notation used for annotating facts with field names, thus improving readability and ease of modification, without loosing efficiency. A structure and it's attributes have to be defined before it's first use, using a struct-predicate, e.g. `:-local struct(driver(id, name, age, state)).` Afterwards, dynamic facts based on the structure can be assert with `:-assert(driver{id:d1, name:sfuchs, age:28}).` Attributes are optional, if they are not listed in the assert-statement, they are assigned a variable. The statement is equivalent to `:-assert(driver(d1, sfuchs, 28, _)).` The main advantage of using structs is that fields within the structure can be addressed using their name, without knowing the exact sequence within the predicate - the translation to the underlying dynamic predicate is automatically done by the parser.

[3] `http://vi.uni-klu.ac.at/ontology/DrivingContext.owl`

We defined nine rules for mapping the abstract objects of the context-model to ECLiPSe structs.

R1: A *class* is mapped into a new structure, using the class name as functor.

```
<owl:Class rdf:about="#driver">
...</owl:Class>
```

is mapped to
`struct(driver(...)).`

R2: A *subclass* is mapped into a new structure, with the superclass' structure as nested field.

```
<owl:Class rdf:about="#spatialContext">
... <rdfs:subClassOf rdf:resource="#context"/>...
</owl:Class>
```

is mapped to
`struct(spatialContext(c:context,...))`

R3: A *functional datatype property* is mapped into a structure field of the corresponding class.

```
<owl:FunctionalProperty rdf:ID="driverId">
...<rdfs:domain rdf:resource="#driver"/>
</owl:FunctionalProperty>
```

is mapped to
`struct(driver(driverId,...))`

R4: A *non-functional datatype property* is mapped into a structure field with a list as valid value. This makes a difference for dynamic facts based on the structure, not for the structure's definition.

```
<owl:DatatypeProperty rdf:ID="state">
 <rdfs:domain rdf:resource="#driver"/>
   <rdfs:range>
    <owl:DataRange>
     <owl:oneOf rdf:parseType="Resource">
       <rdf:first rdf:datatype="...">
         tired</rdf:first>...
     </owl:oneOf>
    </owl:DataRange>
   </rdfs:range>
</owl:DatatypeProperty>
```

is mapped to the structure
`struct(driver(driverId,state,...))`
and used in the dynamic fact
`driver{driverId:sfuchs,state:[tired],...}.`

R5: A *datatype property hierarchy* is mapped into a new structure and used as nested field inside the class structure.

```
<owl:FunctionalProperty rdf:about="#gpsPosition">
 <rdfs:domain rdf:resource="#ownVehicle"/>
  ...
</owl:FunctionalProperty>

<owl:DatatypeProperty rdf:ID="latitude"> ...
  <rdfs:subPropertyOf>
   <owl:FunctionalProperty rdf:ID="gpsPosition"/>
  </rdfs:subPropertyOf>
</owl:DatatypeProperty>
```

is mapped to a new structure
`struct(gpsPosition(latitude,...)).`
and nested as field
`struct(ownVehicle(vehicleId,g:gpsPosition,...)).`

R6: A *functional object property*, representing a simple relationship, is mapped into a structure field of the domain class.

```
<owl:FunctionalProperty rdf:ID="drives">...
  <rdfs:domain rdf:resource="#driver"/>
  <rdfs:range rdf:resource="#ownVehicle"/>
</owl:FunctionalProperty>
```

is mapped to
`struct(driver(driverId,state,drives,...)).`

R7: A *non-functional object property*, representing a complex relationship, is mapped into a new structure: the functor consists of "Domain Class Name + Property Name + Range Class Name" and the domain and range class IDs are fields.

```
<owl:ObjectProperty rdf:ID="execute">
  <rdfs:domain rdf:resource="#driver"/>
  <rdfs:range rdf:resource="#drivingManeuver"/>
</owl:ObjectProperty>
```

is mapped to
`struct(driverExecuteDrivingManeuver(driverId, drivingManeuverType,...)).`

R8: The special *Metainformation class*, assigned to every class via the object-property *hasAdditionalInformation*, is mapped into a new structure and is inserted as nested field into every other structure.

```
struct(metaInformation(source,quality,validity,
timespan))
...
struct(context(contextId,...,
meta:metaInformation)).
```

R9: An *enumeration* of a datatype property is mapped into a static fact with one field containing the enumerated values as list. The functor is made up of "e + datatype property name".

```
eDriverState([tired,drunk,stressed,distracted,
ill, medicated]).
```

An additional check-constraint is generated that validates a field value against the corresponding enumeration. This constraint is used later during the reasoning process to filter out invalid field values.

```
check_eDriverState(Value, Result):-
   eDriverState(List),
   ( member(Value,List)->
     Result=valid
     ;
     Result=invalid
   ).
```

The rules help generating the static framework for the knowledge base. This mapping has to be done once for initialization and afterwards every time, the ontology changes.
Traffic scenario descriptions for specific scenarios can now be described using individuals of the classes defined in the ontology. Individuals are mapped to dynamic facts using a similar mapping process:

- Every individual is translated to one dynamic fact.
- Every datatype property value is assigned to the corresponding structure field of the dynamic fact.
- For every enumeration datatype property, a list of values is constructed and assigned to the structure field.
- For every object property, the id of the corresponding relationship class is assigned to the structure field.
- References to nested object properties are resolved and assigned to the corresponding structure field.
- References to the related metainformation class are resolved and nested into the structure.

Example: The current description of the own vehicle is described with an individual of the "ownVehicle" class and a related meta-information individual, assigned to it with the object property *hasAdditionalInformation*.

```
<ownVehicle rdf:ID="ownVehicle_7">
  <speed rdf:datatype="&xsd;int">110</speed>
  <lineOfSight rdf:datatype="&xsd;float">450.0</lineOfSight>
  <steeringWheelAngle rdf:datatype="&xsd;int">2</steeringWheelAngle>
  <length rdf:datatype="&xsd;float">3.09</length>
  <gear rdf:datatype="&xsd;int">5</gear>
  <width rdf:datatype="&xsd;float">1.65</width>
  <hasAdditionalInformation rdf:resource="#MetaInf_ownVehicle_7"/>
  <drivesIn rdf:resource="#spatialContext_1"/>
</ownVehicle>

<MetaInformation rdf:ID="MetaInf_ownVehicle_7">
   <source rdf:datatype="&xsd;string">staticallyProvided</source>
   <source rdf:datatype="&xsd;string">onBoardSensing</source>
</MetaInformation>
```

The ontology-based description is then mapped to the assert-statement of a dynamic eclipse fact:

```
:-assert(
    ownVehicle{
      objId:ownVehicle_7,
      speed: 110,
      lineOfSight: 450.0,
      steeringWheelAngle:2,
      length: 3.09,
      gear: 5,
      width: 1.65,
      source:[staticallyProvided, onBoardSensing],
      drivesIn:spatialContext_1}).
```

The result of the overall transformation is a number of dynamic facts, the so-called dynamic knowledge base. In contrast to static predicates, dynamic facts can be added to resp. retracted from the knowledge base on demand, without recompiling the rest of the program. Complex rules are further added to the reasoning process for representation of traffic rules. The transformation from OWL to eclipse can be easily automated, using the mapping rules, as will be shown in the next section.

4. DISCUSSION OF THE PROPOSED APPROACH

In addition to the standard ontological tests provided by Protégé, the quality of the ontology was assessed using the ontology engineering criteria defined in [12]:

- **Reusability, standardization:** The model was developed with the main intent of providing a common domain-understanding and a framework for scene description of driving situations. The resulting machine-readable descriptions enable information exchange and reasoning for DAS. The model can be used for all tasks that need driving scenario descriptions (e.g. ACC, intersection assistance, etc.) on a tactical level.
- **Flexibility, extensibility:** New definitions can be added to the context-model without changing existing dependencies. Particularly stepwise refinement using OWL class hierarchies is easily possible, thus enabling applications to enhance the class-descriptions to the level of detail needed.
- **Genericity:** Our context-model is restricted to the driving domain and does not provide a domain-independent upper ontology.
- **Granularity:** The model consists of abstract objects representing a high-level description for the tactical level of the driving domain. Refinement to finer descriptions for the operative level is easy (see also flexibility).
- **Consistency:** No contradictions where found in the ontological content.
- **Completeness:** The empirical validation showed that the model is complete for high-level scene representation of driving scenarios. The rules necessary for efficient reasoning are implemented outside the model with a logic-based approach as discussed before.
- **Redundancy:** The context model does not contain redundant information.
- **Readability:** Labels of classes and properties where chosen with respect to understandability by human designers. Although the models main intent is information exchange between DAS, it should be understandable by human developers.
- **Scalability:** Cognitive and engineering scalability of the context-model is unproblematic, since the model contains a small number of possible objects. Reasoning scalability is unapplicable, because reasoning is done outside the model.
- **Language, formalism:** Description logics (more specific OWL) is used for scene representation. For the reasoning process we use a logic-based approach (based on the scene description).

We found that the model is sufficiently exhaustive to represent all relevant traffic objects in a driving situation, as well as their relationships to each other and the own vehicle. To summarize, the model fulfills the applicable ontological engineering criteria to a great extent, and is suitable for all tasks within a driver assistance system that need driving scenario descriptions on a tactical level. The model can be easily extended, without changing existing dependencies, using a step-wise refinement process exploiting OWL class hierarchies. Applications can enhance the class descriptions to a higher detail-level, any time, if needed.
For the empirical validation, we chose a number (about 120) of representative driving scenarios from both real-world

video-streams and a driving school's teaching resource, as a start. Most scenarios contained intersections or overtaking scenes, because our prototypical DAS will be implemented around these maneuvers. Intersection scenarios were chosen from both cities and rural roads, containing various numbers (including zero) and types of traffic participants with different relationships to the own vehicle. From the overall 120 scenarios, 50 where used for representation of overtaking maneuvers, 40 for intersections and the rest for various situations. Different scenarios were chosen for validating traffic objects (with/without signs, traffic lights and markings) as well as for various weather conditions (day/night/twilight) and driver states. For overtaking, scenarios where chosen from highway, city and rural roads, with a different number of lanes, with and without oncoming traffic, with different distance/speed combinations, line of sights and much more. We modeled every scenario description manually and comprehensively based on the developed context ontology, representing all relevant traffic objects and their relationships to the own vehicle that were present in the chosen scenario. Assumptions were made for values that will be provided by sensors in the real system and that were not known explicitly for the scenario: examples are the state of the driver, road surface temperature, technical features of the own vehicle and their state. A mapping component then was developed that automatically transforms the OWL-based individuals and relationships to dynamic facts, using the context ontology. The result set of facts is afterwards automatically compiled into the dynamic knowledge base of an ECLiPSe environment to check the syntactical correctness. Comparison for semantic equivalence was done manually against both the scenario description and the original traffic scene.
The ECLiPSe environment provides a tightly coupled interface to the programming languages TCL/TK and C++. A loose interface is available for Java. We chose the open-source script language TCL/TK for the development of the mapping component. The program takes the context-ontology and the OWL-based scenario descriptions as input and creates a dynamic fact file. First, the ontology structure is analyzed (identification of classes, enumerations, etc.). The results are then used for interpretation and transformation of the individuals file. Each scene description file has between 5 and 20 Kb, depending on the number of involved context objects. Although TCL/TK is an interpreted script language, this process only takes between 100 and 150 milliseconds on average on a standard laptop PC with Windows XP, a 2 GHz processor and 2 GB RAM (see fig. 3).

The mapping component is only one part of a prototype for an overtaking and intersection assistant: a rule-base is currently under development for deducing overtaking and intersection-crossing recommendations, using the dynamic knowledge base, created by the mapping component, as input. The rule-base is not complete yet, but first results look promising: for the average overtaking situation, the deduction process takes about 1 to 2 milliseconds on the laptop PC described above.

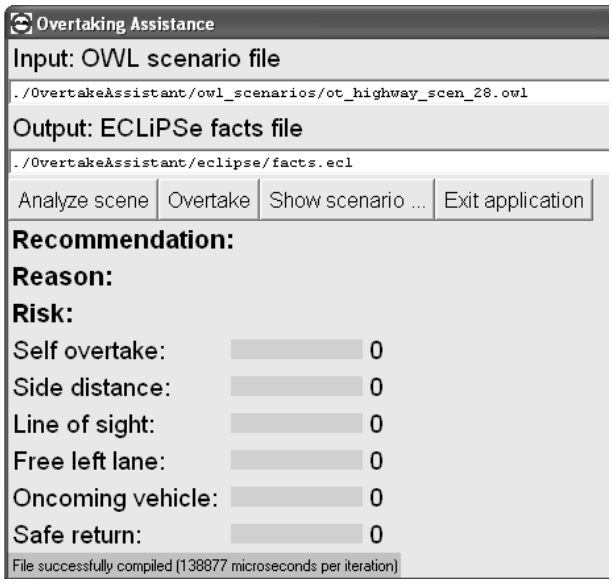

Figure 3: Mapping component for transforming the OWL scene description to a dynamic knowledge base

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an example that emphasizes the need for including the traffic context in the decision process of a DAS. Especially co-operative DAS demand a common domain understanding for scene representation to enable information exchange between vehicles. We developed an ontology-based driving context-model that is sufficiently exhaustive to describe a representative variety of traffic scenarios and serve as common basis for domain-understanding, information-sharing and decision-making. We further discussed the integration of modeled scenarios and traffic rules within a logic programming environment, because OWL's support for rule representation is still in it's infancy. The presented approach for information integration shows first promising results and demonstrates the feasibility of combining the advantages of ontologies with the reasoning power of logic-based languages. Future work will be concerned with a more comprehensive extension of the overtaking knowledge base that is able so adapt its decisions to the individual needs of the driver and that takes the meta-information of traffic objects into account during the reasoning process.

6. REFERENCES

[1] M. Baumann, T. Petzoldt, and J. Krems. Situation Awareness beim Autofahren als Verstehensprozess. In *MMI-Interaktiv*, volume 11, pages 43–57, 2006.

[2] N. Baumgartner, W. Retschitzegger, and W. Schwinger. Lost in Time, Space and Meaning - An Ontology-Based Approach to Road Traffic Situation Awareness. In *Third Workshop on Context-Awareness for Proactive Systems (CAPS'07)*, 2007.

[3] A. Benmimoun. The Driver as Archetype for Driver Assistance Systems? A Driver Model Based Approach for the Development of Situation-Adaptive DAS. In *13. Aachener Kolloquium Fahrzeug- und Motorentechnik*, 2004.

[4] H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *Knowl. Eng. Rev.*, 18(3):197–207, 2003.

[5] M. Endsley. Towards a theory of situation awareness in dynamic systems. *Human Factors*, 37:32–64, 1995.

[6] D. Gavrila. Pedestrian Detection from a Moving Vehicle. In *Computer Vision - ECCV 2000*, volume

LNCS 1843, pages 37–49, 2000.

[7] T. Gu, X. Wang, H. Pung, and D. Zhang. An Ontology-based Context Model in Intelligent Environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004.

[8] A. Heitmann, R. Guttkuhn, A. Aguirre, U. Trutschel, and M. Moore-Ede. Technologies for the monitoring and prevention of driver fatigue. In *Proceedings of the First International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design*, pages 81–86, 2001.

[9] K. Henricksen and J. Indulska. A Software Engineering Framework for Context-Aware Pervasive Computing. In *2nd IEEE Conference on Pervasive Computing and Communications (PerCom)*, 2004.

[10] I. Horrocks. OWL Rules, OK? In *W3C Workshop on Rule Languages for Interoperability*, 2005.

[11] I. Horrocks and P. Patel-Schneider. A proposal for an OWL rules language. In *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, pages 723–731, 2004.

[12] R. Krummenacher and T. Strang. Ontology-Based Context-Modeling. In *Third Workshop on Context Awareness for Proactive Systems (CAPS'07)*, 2007.

[13] A. Lattner, J. Gehrke, I. Timm, and O. Herzog. A Knowledge-based Approach to Behavior Decision in Intelligent Vehicles. In *Proceedings of the Intelligent Vehicles Symposium*, pages 466–471, 2005.

[14] J. Michon. *Generic Intelligent Driver Support - A comprehensive report on GIDS*. Taylor & Francis Ltd, 1993.

[15] A. Miene, A. Lattner, U. Visser, and O. Herzog. Dynamic-preserving Qualitative Motion Description for Intelligent Vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV'04)*, pages 642–646, 2004.

[16] J. Miura, T. Kanda, and Y. Shirai. An Active Vision-System for Real-Time Traffic Sign Recognition. In *Proceedings of the 2000 IEEE International Conference on Intelligent Transportation Systems*, pages 52–57, 2000.

[17] Q.Ji and X. Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, 2002.

[18] A. Rakotonirainy and F. Maire. Context-aware Driving Behaviour Model. In *Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles (ESV'19)*, 2005.

[19] A. Shashua, Y. Gdalyahu, and G. Hayun. Pedestrian detection for driving assistance systems: single-frame classification and system level performance. In *IEEE Intelligent Vehicles Symposium*, pages 1–6, 2004.

[20] Q. Sheng and B. Benatallah. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services. In *Proceedings of the International Conference on Mobile Business (ICMB'05)*, volume 00, pages 206–212, 2005.

[21] C. Simons. CMP: A UML Context Modeling Profile for Mobile Distributed Systems. In *Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS'07)*, page 289b, 2007.

[22] T. Strand, C. Linnhoff-Popien, and K. Frank. CoOL: A Context Ontology Language to enable Contextual Interoperability. In *Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003)*, volume LNCS 2893, pages 236–247, 2003.

[23] T. Strang and C. Linnhoff-Popien. A Context-Modeling Survey. In *First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004*, 2004.

[24] M. Toennis, G. Klinger, and J.-G. Fischer. Ontology-based Pervasive Spatial Knowledge for Car Driver Assistance. In *Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications Workshop (PerComW'07)*, pages 401–406, 2007.

[25] M. Trivedi and S. Cheng. Holistic Sensing and Active Displays for Intelligent Driver Support Systems. *Computer*, 40(5):60–68, 2007.

[26] Z. Wu, Q. Wu, H. Cheng, G. Pan, M. Zaho, and J. Sun. ScudWare: A Semantic and Adaptive Middleware Platform for Smart Vehicle Space. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):121–132, 2007.