REAL-TIME CONTEXT AWARE REASONING IN ON-BOARD INTELLIGENT TRAFFIC SYSTEMS

An Architecture for Ontology-based Reasoning using Finite State Machines

Arjan Stoter, Simon Dalmolen, Eduard Drenth, Erik Cornelisse and Wico Mulder
Logica, Prof. Keesomlaan 14, Amstelveen, The Netherlands

Keywords: ITS, HMI, Ontology, FSM, Reasoning, Context-awareness, Inference engine.

Abstract: In-vehicle information management is vital in intelligent traffic systems. In this paper we motivate an architecture for ontology-based context-aware reasoning for in-vehicle information management. An ontology is essential for system standardization and communication, and ontology-based reasoning allows context-awareness, inference and advanced reasoning capabilities. However, the amount of computational power it requires often conflicts with the computational limitations of on-board units, as well as the high demand for timeliness and safety. Our approach uses ontology-based reasoning and a finite state machine (FSM). By combining ontology and FSM, we illustrate how a heavy-weight reasoning-solution could be applied in a light-weight computational environment.

1 INTRODUCTION

The role of Intelligent Traffic Systems (ITS) becomes increasingly important in traffic management. The advance of Intelligent Infrastructure Systems (IIS) and In-Vehicle Information Systems (IVIS) propose a challenge for Human Machine Interaction (HMI). Advanced Driver Assistance Systems (ADAS) such as adaptive cruise-control, and Advanced Traveller Information Systems (ATIS) such as navigation and hazard warnings can support the driver. However, as traffic places high demands on human information processing (Barfield & Dingus, 1998), supplying additional information to the driver can easily result in an information overload (Lui, 2001; Jamson & Merat, 2005). In-vehicle information management aims to prevent information overload, and is therefore vital in ITS.

In-vehicle information management is defined here as a process that manages in-car information originating from ADAS and ATIS, and presents it to the driver via a dynamic presentation layer. The current study describes a software architecture for in-vehicle information management, referred to as the HMI manager. The HMI manager orchestrates interaction between in-car devices and applications, and the user (in this case the driver). It determines which information is shown to the user by assessing relevance and priority of information.

The first part of this paper describes the role and requirements of the HMI manager. In the second part we discuss our approach. We consider ontology-based systems and how they can be used for information management. We discuss the amount of computational power required for ontology-based reasoning, and how this commonly conflicts with the computational limitations of on-board units. We suggest the use of finite state machines in combination with ontology-based reasoning, and argue how heavy-weight (ontology-based) reasoning could be applied in a light-weight computational environment.

The architecture proposed in the current study is part of the Open Platform Solution for ITS, developed within the Strategic Platform for Intelligent Traffic Systems (SPITS). SPITS is a unique collaboration between the Dutch government, companies and scientific institutes (see http://www.spits-project.com/).

2 HMI MANAGER

The HMI manager orchestrates interaction between the driver and in-vehicle applications (ADAS and

Stoter A., Dalmolen S., Drenth E., Cornelisse E. and Mulder W..
REAL-TIME CONTEXT AWARE REASONING IN ON-BOARD INTELLIGENT TRAFFIC SYSTEMS - An Architecture for Ontology-based Reasoning using Finite State Machines.
DOI: 10.5220/0003185506370642
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 637-642
ISBN: 978-989-8425-40-9

ATAS). It determines priority of information, and provides information to the user that is relevant in a given situation. We focussed on two use-cases: [1] user applications can provide information (that will be passed on to the user by the HMI manager), [2] an external device (e.g. sensor) gives feedback about its status and controls via the HMI manager. User input was outside the scope of the current study.

Two key-drivers were defined for the HMI manager: timeliness and safety. Timeliness relates to real-time response and deliverance of information. Safety relates to reliability and predictability of the information-management process. In addition the following capabilities were defined: presentation of information, prioritisation of information, and the ability to remember information, states, and modes.

Figure 1 shows the capability-overview of the HMI manager. Three components were defined: the PresentationManager, the MentalState, and the PriorityManager. Together these are the core components of the HMI manager.

The PresentationManager component contains logic and configuration of all presentation-specific information like style and layout. The MentalState component collects, stores and shares information received from user applications as well as input from external devices. The PriorityManager contains priority logic, and determines which information is presented to the user, in which order. It does so depending on the content of the MentalState.

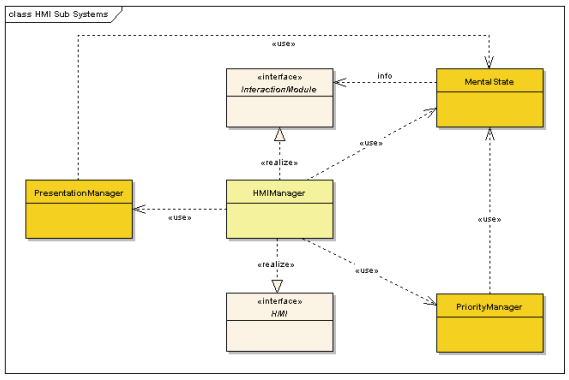

Figure 1: Capability overview of the HMI manager. The PresentationManager contains logic and configuration of all presentation-specific information. The MentalState component collects, stores and shares information received from user applications. The PriorityManager determines which information is presented to the user, and in which order, depending on the content of the MentalState.

The HMI manager was placed in between an application layer and presentation layer (see figure 2). The application layer contains applications that belong to ADAS and ATIS, which supply real-time (event driven) information related to safety, navigation, road conditions, traffic congestions, and so on. (User input was outside the scope of the current study, hence the one-way arrows in figure 2.)

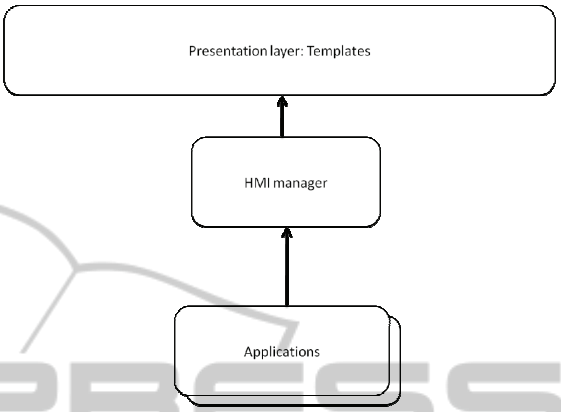

Figure 2: Position of the PM in IVIS. The PM is located in between the application- and presentation layer.

The top-layer in figure 2 is the presentation layer. Components in this layer present information to the driver. The presentation layer is dynamic; the templates of the presentation layer can change depending on the information it has to show. The templates are loaded upon request from the PresentationManager, located inside the HMI manager.

Note that the HMI manager is event-driven. Information is sent to the HMI manager on occurrence of events.

2.1 Information Management

Information is sent by applications as information messages. These messages contain several properties, two of which are importance and urgency. These are set by the applications in the current setup. Importance of a message depends on the message type. For instance, a safety-critical message will have higher importance than infotainment messages. Urgency relates to time criticality; how vital it is that information is shown at the time the event occurs. For example, a message related to fuel level will become more urgent as the fuel-level drops.

The PriorityManager uses importance- and urgency values, and the MentalState (see figure 1) to determine message priorities. The MentalState stores and shares information received from user applications and represents the context of the information prioritisation process.

2.1.1 Context-awareness

Context-awareness (in software) is used to describe collaboration between components in order to deal with data that describe real-world, complex situations. It involves semantic data and often semantic Web technology (e.g. OWL) (Feki & Mokhtari, 2006), and use of external data to build a model of the world (an awareness) (McCann et al, 2004; Shaou-Gang et al 2007).

Picture a sunny day as you drive along the country side. As the scenery passes, the fuel level drops. Though there is no reason to be alarmed just yet (normal importance, low urgency), your navigation system knows you are about to pass a gas station (low importance, high urgency). In addition, it also notices that the next gas station is several miles away (low importance, low urgency). Now, each of these information elements individually would have little meaning, or low priority. But its combined result is rather important: refuel now or strand later. As a result, the event of the approaching gas station suddenly gets a high priority. This is what we mean by context-aware reasoning.

3 APPROACH

3.1 Ontology-based Systems

An ontology is described as a shared conceptualization of the types and relations of things that are in the world (Wang et al, 2009). In computing science, ontology's are used as a data model for semantic representation and conceptualization of knowledge domains (Zhou et al 2008). An ontology represents a domain-vocabulary containing all essential concepts, their relations, and axioms and constraints. Using an ontology it is possible to extract relevant domain terminology and to extend an existing concept hierarchy by adding new concepts (William et al 2009). Ontology's are widely accepted for deductive reasoning, inference, and context-based reasoning (Martín et al, 2008).

An ontology-based approach fits the capabilities of the HMI manager. First, an ontology is essential for system standardization and communication. The HMI manager should be able to interact with multiple applications, and new information entities should be easily added. Second, an ontology allows inference about relationships between entities, and as such provides a powerful solution to reason about incoming information messages. Finally, being able to infer about entities and their relations provides a powerful solution for contextual reasoning about messages.

3.2 Finite State Machine

An FSM is commonly described as a behavior model that contains a finite number of states and transitions, as well as actions. A state is a unique configuration of the machine. An FSM can be defined as equation 1.

$$M = \{I, O, S, \delta, \lambda\} \quad (1)$$

I, O and S are finite and nonempty sets of input symbols, output symbols, and states. Transition between states is described by the state transition function δ: S x I -> S and the output function λ: S x I -> O. The transitions between states are described in a transition table, which can be represented as a directed graph (see figure 4, right) (Lee & Yannakakis, 1996).

An FSM is a relatively light-weight process with a predictable number of outcomes. By nature it is a clean, low demanding solution to problems that can be modelled as a 'machine' with a state, where events will trigger the transition to another state (e.g. Lee & Yannakakis, 1996). In our study, an FSM meets the HMI manager's requirements of timeliness and safety (where safety is a predictable outcome).

4 IMPLEMENTATION DESIGN

Using an ontology and rule engine provides a powerful approach to model the HMI manager's reasoning. However, the computational power required by ontology-based reasoning commonly conflicts with computational limitations of on-board units, which in turn conflicts with the HMI manager's key-drivers of timeliness and safety.

We suggest a two-step approach that includes configuration-time and run-time. At configuration time we designed an ontology with classes, relations and rules. From there we generate a finite state machine that was used at runtime.

4.1 Configuration Time

4.1.1 Ontology Reasoning

OWL web ontology language was used for the knowledge base. The inference engine was created using semantic web rule language (SWRL). SWRL provides a data-driven approach in the sense that new "context" information will trigger the SWRL

inference rules. SWRL is based on OWL and RuleML, which can use the knowledge (in terms of instances) defined in OWL to develop rules (Lui et al, 2010). We used protégé to develop OWL and SWRL.

4.1.2 Managing Priorities

The PriorityManager (figure 1) has an inference engine to determine priority of information messages. Inference rules describe the relation between importance, urgency (see 2.1), and priority of a message. In the current setup, 3 values were used for importance and urgency: low, normal, and high. Below is a SWRL example of a rule, where message *x* with normal importance and high urgency will get a high priority:

```
ImportanceNormal(?x),
UrgencyHigh(?x) ->
PriorityHigh(?x)
```

Table 1 illustrates the relation between importance and urgency, and priority of an information message. The outcome of the table represents the priority level.

Table 1: Basic relations between importance (left to right) and urgency (top to bottom). The out-coming values in the table represent the priority level.

	Low Importance	**Normal Importance**	**High Importance**
Low Urgency	Low Priority	Low Priority	Normal Priority
Normal Urgency	Low Priority	Normal Priority	High Priority
High Urgency	Low Priority	High Priority	High Priority

Figure 3 shows the logical components inside the HMI manager at configuration time. At the bottom are applications that send messages. Next, messages enter the MentalState where they are instantiated according to concepts of an ontology. This in turn triggers the PriorityManager, which holds an inference engine. This engine uses the ontology structure to determine relations between concepts, and uses rules to make decisions about these concepts, given the MentalState. Finally, the message is labeled with a priority and placed on a message stack inside the PresentationManager.

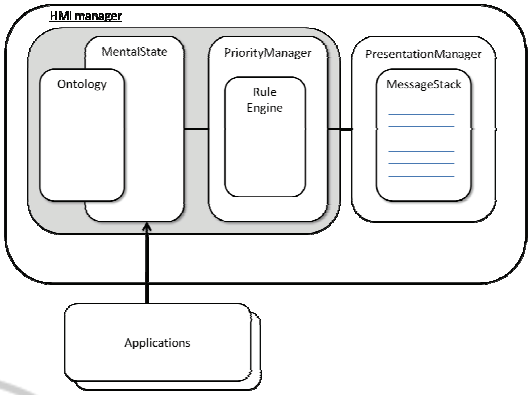

Figure 3: Logical overview of the HMI manager at configuration time. Applications send messages to the HMI manager where they enter the MentalState. Messages are instantiated in the MentalState according to concepts of the ontology. This in turn triggers the PriorityManager that holds a rule engine for the inference process.

4.2 Runtime

At configuration time the MentalState (plus ontology) and PriorityManager (with inference engine) are separate parts. At runtime these components are combined into an FSM (see figure 4). This is done by querying the ontology and creating a Cartesian product containing all possible inferences. That is, all possible messages and their priorities according to the SWRL rules. This resulted in a large amount of outcomes with all possible mental states (all combinations from the set of messages), where each message has a priority per state. Next, through a process of optimization an FSM is created with the optimal number of states.

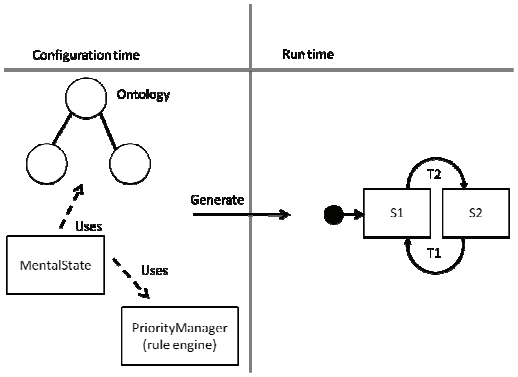

Figure 4: Transition between configuration time (left) and runtime (right) of the HIM manager. The MentalState with the Ontology, and the PriorityManager with the rule engine are transformed into a Cartesian product of machine states, containing all possible inferences. Next, through process of optimization an FSM is created with the minimum number of states.

5 DISCUSSION

Implementing context-aware reasoning in on-board units is a challenge. Ontology-based reasoning provides a powerful solution, but the computational power it requires conflicts with the capacities of on-board units. The solution presented in this study is the transformation of the ontology, rule engine, and MentalState into an FSM.

We encountered several challenges that require future research. These challenges relate to modelling the ontology and rule definitions, and the transition process between the knowledge base and FSM.

First, for modelling the ontology we are currently investigating the use of message classes. These classes are safety, infotainment, navigation, and car status. Rules in the rule engine would (for instance) state that messages belonging to safety always get a higher priority than those belonging to infotainment.

The second challenge lies in defining rules and relations between concepts, and how to make sure that some crucial part of the inference process is not overlooked. These issues are not trivial, as controlling the inference process to guaranty predictability and safety is crucial.

Finally, an FSM at runtime provides a light-weight solution that enables timeliness and predictability. However, the transition process of ontology-based reasoning into an FSM requires extensive research. Creating a Cartesian product of all inferences, followed by an optimization process has been promising. But defining the optimization-rules is a delicate process that requires follow-up studies. Also, it is expected that the size of the FSM's decision table is related to the ontology structure. As the FSM handles states we are currently investigating a state-driven structure for the ontology.

The issues described in the previous paragraph are currently addressed in collaboration with TNO, and prototypes have been created. The architecture will be implemented and tested inside a car simulator, and will be validated with participant experiments on simulated driver performance.

6 CONCLUSIONS

The architecture for the HMI manager described in the current study used ontology-based reasoning and an FSM. We believe that this approach has the potential to provide the best of both worlds, in that it places the power of an elaborate reasoning solution into a light-weight computation environment.

Our solution is capable of providing context-aware reasoning while maintaining timeliness and safety in real-time, demanding traffic situations. Extensive research will have to be performed on issues regarding the knowledge base and the FSM transition and optimization. But we believe that the present architecture of the HMI manager provides an important first step towards in-vehicle information management, as part of an Open Platform Solution for ITS.

REFERENCES

Barfield, W., Dingus, T. A., 1998. *Human factors in intelligent transportation systems.* Lawrence Erlbaum Associates. Mahwah, New Jersey.

Feki, M. A., Mokhtari M., 2006. *Context aware and ontology specification for assistive environment, HWRS-ERS Journal, International Journal of Human-friendly Welfare Robotic Systems*, Vol. 4, no. 2, pp. 29-32.

Jamson, A. H., Merat, N., 2005. *Surrogate in-vehicle information systems and driver behaviour: effects of visual and cognitive load in simulated rural driving.* Transportation Research Part F: Traffic Psychology and Behavior, Vol. 8, no. 2, pp. 79-96.

Lee, David, Yannakakis, Mihalis, 1996. *Principles and methods of testing finite state machine- a survey.*

Liu, C. H., Chang, K. L., Jason, J. Y., Chen, Hung, S. C., 2010. *Ontology-Based Context Representation and Reasoning Using OWL and SWRL*. 8th Annual Communication Networks and Services Research Conference.

Liu, Y. C., 2001. *Comparative study of the effects of auditory, visual and multimodal displays on driver's performance in advanced traveller information systems.* Ergonomics, Vol. 44, no. 4, pp. 425-442.

Martín, L., Anguita, A., Maojo, V., Bonsma, E., Bucur, A., I., D., Vrijnsen, J., Brochhausen, M., Cocos, C., Stenzhorn, H., Tsiknakis, M., Doerr, M., Kondylakis, H., 2008. *Ontology Based Integration of Distributed and Heterogeneous Data Sources in ACGT*. HEALTHINF 1INSTICC - Institute for Systems and Technologies of Information, Control and Communication, pp. 301-306.

McCann J.A., Kristofferson P., Alonso, E., 2004. *Building Ambient Intelligence into a Ubiquitous Computing Management System, International Symposium of Santa Caterina on Challenges in the Internet and Interdisciplinary Research.*, Amalfi, Italy (January, SSCCII-2004).

Shaou-Gang, M., Fu-Chiau S., Chia-Yuan H., 2007. *A Smart Vision-Based Human Fall Detection System for Telehealth Applications*. IASTED Telehealth Conference (Montreal, QC, Canada) (May- June 2007).

Wang, J., Zuo, W., He, F., Wang, Y., 2009. *A New Formal Description of Ontology Definition and Ontology Algebra*. Second International Symposium on Knowledge Acquisition and Modeling.

William L., S., Kristina L., W., Zhengxin C., 2009. *Constructing Domain Ontology from Texts: A Practical Approach and a Case Study*. Fifth International Conference on Next Generation Web Services Practices.

Zhou, L., Zhang, D., Chen, X., Zhang C., 2008. *A Method for Semantics-based Conceptual Expansion of Ontology*. Proceedings of the 2008 ACM symposium on Applied computing.