CONTEXT-AWARE PRIORITIZATION OF INFORMATION: AN ARCHITECTURE FOR REAL-TIME IN-VEHICLE INFORMATION MANAGEMENT

Arjan Stoter, Erik Cornelisse, Simon Dalmolen

arjan.stoter@logica.com, erik.cornelisse@logica.com, simon.dalmolen@logica.com
Logica. Prof. W.H. Keesomlaan 14, 1183 DJ Amstelveen, The Netherlands.
Phone: +31 88 564 0000, fax: +31 20 503 3018

Abstract: Human Machine Interfaces (HMIs) are the communication devices between in-vehicle applications and the driver. Frequently, independent HMIs are used for different applications which may cause information overload situations at the driver. Hence, a common HMI design that determines the application interface with an HMI Manager that provides rules for prioritizing information from different applications is needed. The design takes into account: application transparency, driver situation awareness, driver workload and surrounding external influences. By reducing distraction, especially when the situation requires more attention of the driver, the need for controlling the information flow emerges. This paper presents a driver-centred unified approach for in-vehicle information management.

INTRODUCTION

The advance of intelligent infrastructure systems (IIS) and in-vehicle information systems (IVIS) propose a challenge for human machine interface (HMI) inside road vehicles. Current applications often use independent interaction channels, which creates potentially high risks for information conflicts and information overload in the driver that compromise traffic safety (1, 2). Effective and safe HMI in road-vehicles requires a driver-centred unified approach for in-vehicle information management of IIS and IVIS: a process that orchestrates information-flows from different on-board applications, in order to optimize information delivery to the driver.

In a previous study, we described an approach for context-aware information-management inside road-vehicles, using a knowledge base (3). In the current paper, we elaborate on our approach and present an application-transparent design of an HMI-manager that manages (prioritizes) information from multiple on-board applications, according to vehicle situational-awareness and driver workload. The main goals of the HMI manager are (1) to manage graphical-, audio- and tactile channels for the interaction between the driver and user applications, and (2) to manage this information based on priority.

SCOPE

The HMI-manager was designed as a core service of an on-board unit (OBU). The OBU is part of the strategic platform for intelligent transportation systems (SPITS). The HMI-manager handles information-flow between driver and applications, and coordinates the presentation of information by interaction devices (switches, lights, displays, audio channels or any other kind of interaction hardware). The service-nature of the HMI-manager allows user-applications to provide information to the HMI-manager at a functional level, without the need for the HMI-manager to require any knowledge of the user-applications. Three key-capabilities were defined for the HMI Manager: (1) presentation of information, (2) prioritisation of information, and (3) the ability to remember information, states and modes.

The HMI architecture uses abstract layers, where each layer serves the layer above and without the need to know the details of the layer below (figure 1).

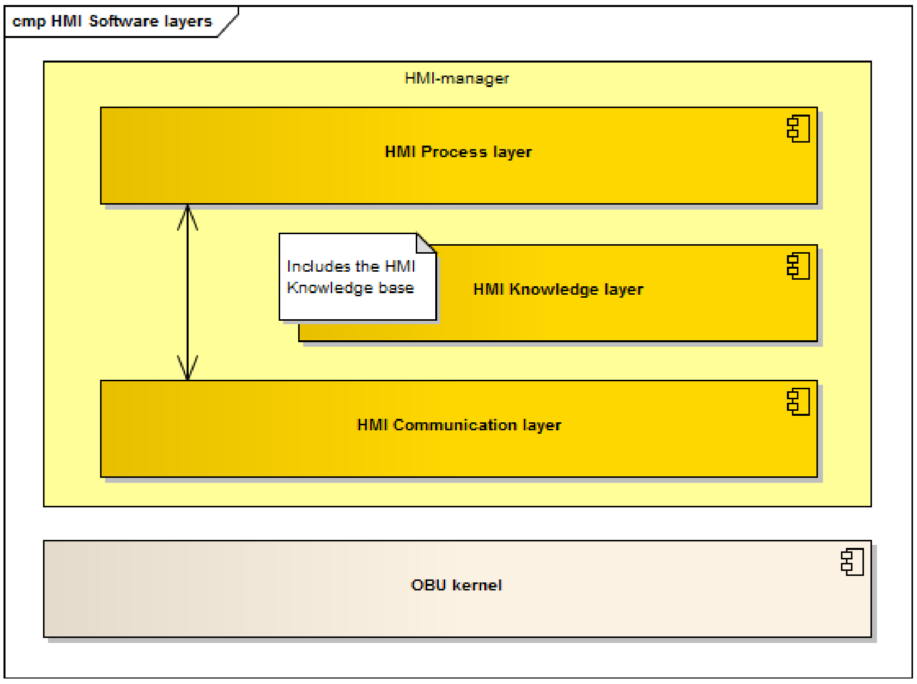

Figure 1. Layers of the HMI architecture. The HMI Communication layer handles the asynchronous inter-process communication, the HMI Knowledge layer stores configuration and received information, and the HMI Process layer contains the presentation and prioritisation-mechanisms.

The HMI Communication layer handles the asynchronous inter-process communication (IPC) with external devices, user applications and other OBU services. The “man-in-the-middle” design-pattern is used to decouple functional capabilities of the HMI-manager from OBU-specific implementation. Information from external devices and user-application will be transformed to a functional level in a uniform format and be stored in the HMI-Knowledge base.

The HMI Knowledge layer collects and stores configuration and received information. In addition it contains decision-rules that (1) determine how the HMI Manager reacts upon incoming information, and (2) determine relationships between information in order to allow context-based reasoning.

The HMI Process layer contains the actual presentation and prioritisation-mechanisms, which are triggered by the HMI Knowledge layer. User interaction that does not require context-awareness or priority control can by-pass the HMI Knowledge layer (the arrow in figure 1 between the HMI Communication layer and Process layer). E.g. a video stream from a rear-view camera can and should be displayed to the driver instantly.

IMPLEMENTATION

This section describes the composite structure of the HMI architecture in relation to the abstract layers in Figure 2. The following components were identified: (1) presentation and priority managers (HMI Process layer), (2) a context manager (HMI Knowledge layer), and (3) an adapter manager for communication with the OBU (HMI Communication layer). Figure 2 shows the composite structure of the HMI-Manager.

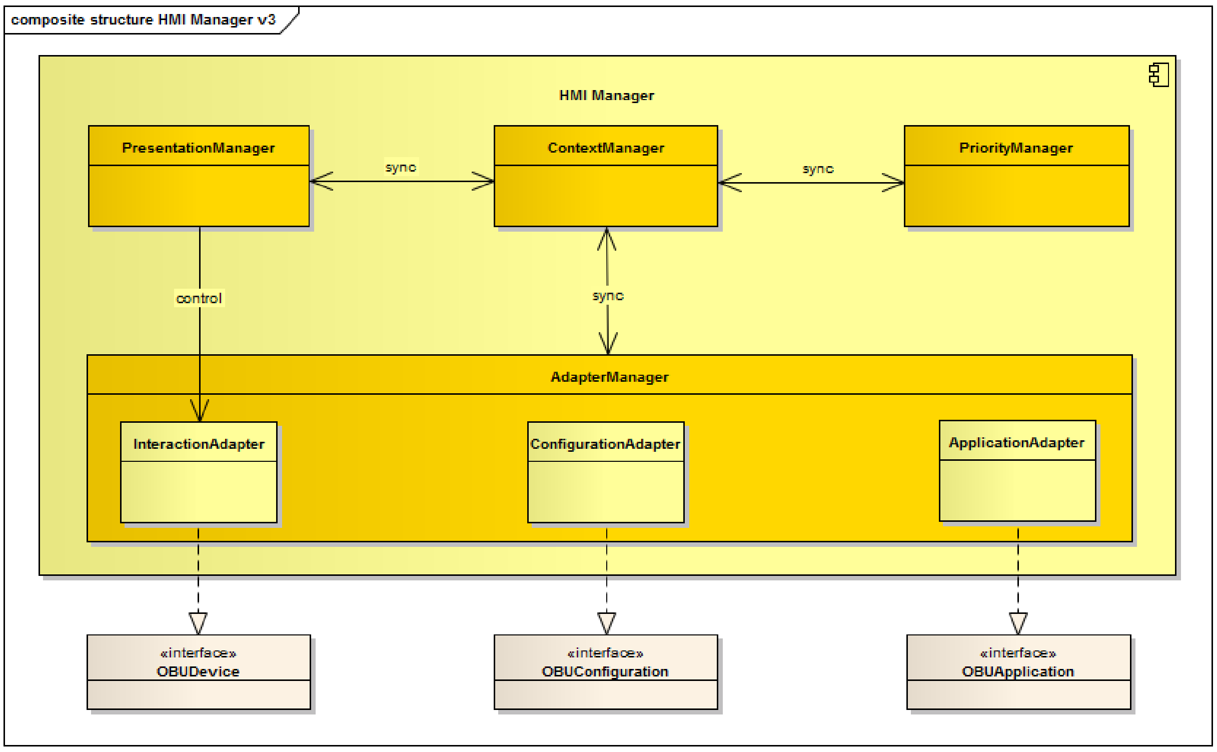

Figure 2. Composite structure of the HMI-Manager.

PROCESS LAYER

The Presentation manager (figure 2) was introduced as a specialised component of the HMI-Process layer to transform user-generated events into events for user applications and external devices, and vice versa. It contains presentation logic and uses presentation-specific configurations (e.g. style and layout), which are stored in the HMI Knowledge base. The Priority manager is the second specialised component within the HMI Process layer. It contains and handles priority logic to determine which information is presented, in what amount, as well as the order in which it is presented. We will elaborate on the prioritization process itself later on.

KNOWLEDGE LAYER

The Context manager (figure 2) implements the HMI Knowledge layer. It stores and shares the context of the vehicle and the driver. Information received from user-applications as well as (user) input from external devices will be made available (synchronised) by this component for other HMI components, such as the Priority Manager and the Presentation Manager. The Context manager contains an Event-manager that informs other components about new or updated information. The Context manager is described in more detail in the next section.

COMMUNICATION LAYER

The HMI Communication layer (figure 1) is implemented by 3 adapters and a coordinating manager (figure 2): (1) The Interaction adapter that implements interfaces with OBU devices, (2) The Configuration adapter, to connect to the OBU configuration services, and (3) the Application adapter, to implement a generic interface for user-applications (Figure 2). All three adapters are gateways to adapt the incoming events into the right format for the Context manager. An Adapter manager is required to orchestrate concurrent I/O of the HMI-manager.

PRIORITIZATION OF INFORMATION-ELEMENTS

The HMI-manager prioritises atomic information-elements according to urgency and importance. These concepts are explained in section 4.1. Atomic information-elements are the smallest pieces of functional information. For instance, congestion information may contain information about multiple traffic jams. Information about a single traffic jam is considered here to be the smallest functional element, which contains multiple attributes to specify its characteristics (e.g. location, length, duration).

Priority of information elements is ranked according to 4 categories: urgent & important (1), urgent & not important (2), not urgent & important (3), and not-urgent & not important (4) (see table 1).

Priority	Urgent	Not urgent
Important	1	3
Not important	2	4

Table 1. priority of an information element is ranked according to 4 categories: urgent & important (1), urgent & not important (2), not urgent & important (3), and not-urgent & not important (4).

Information elements ranked as priority category 1 will be shown immediately and unconditionally while presentation of those ranked as category 4 might be postponed until the situational context allows it.

IMPORTANCE AND URGENCY

Importance and urgency are used to determine priority of (atomic) information-elements. In order to achieve priority-ranking, atomic information-elements contain the following attributes:

- Event type [mandatory]
- Importance [optional]
- Timestamp [optional;
- Location [optional]
- Validity [optional]

Event type represents the category of information elements, where each element belongs to a specific type, e.g. vehicle status, navigation or entertainment. It is used to determine importance of information-elements.

Importance will be deduced from event-type by the priority manager. This could state that -for instance- information belonging to type “vehicle-status” will always have a high importance. The importance-attribute value can be overruled by user applications to differentiate between messages of the same event-type. For example, the importance of fuel-level information can change from “not important” to “important” when the fuel level passes a configurable threshold.

Timestamp is the absolute creation-time of the information element by an application. If omitted it will be set to the time it was received by the HMI-manager. The timestamp will be used to rank information-elements according to age, within each priority category. Recent information is considered to be more relevant than older information elements within the same priority category.

Urgency of (atomic) information-elements relates to its temporal nature: the relevance of information at a specific time and location. It is represented by the validity attribute. Validity (and hence urgency) is dynamically linked to situational context of the vehicle, and therefore directly linked to the location attribute.

Location of the vehicle can be absolute (expressed in latitude and longitude coordinates), relative (expressed as road attributes, e.g. road id and hectometre signs), and logical (referring to a place or area using geo-fence).

Validity of information-elements will increase as the vehicle approximates the location of the event. Since urgency is time related, the proximity will be expressed in seconds until the information gets “in-range”. Information becomes relevant or in-range when the proximity exceeds a configurable threshold. Based on absolute coordinates, the proximity is calculated as the distance divided by the current velocity. In case of a relative or logical location, the proximity is Boolean.

DRIVER WORKLOAD AND CONTEXT STATES

Workload of the driver was simplified to four discrete values represented as concentration modes for the driver:

A – Alerts, direct attention and focus on driving are required. Only urgent and important messages will be passed.

H – High attention of the driver, extra concentration is required. Only urgent messages will be passed.

N – Normal situation, information is presented in order of priority. No information will be hold back or postponed.

L – Low concentration is required, no limitations/restrictions. Information will be presented at order of reception, with no prioritisation.

A finite number of context states (CS) are used. These are categorised according to their cause or condition and translated into 1 of the 4 concentration modes for the driver. For the driver, the following conditions were taken in account:

Driver conditions	CS
• Preferences of the driver	
□ Quite mode, “Do not disturb me”	A
□ Hibernate mode “Wake me up only if urgent”	H
□ Normal mode (default mode)	H
□ Let me know everything when the situation allows it.	L

Concerning the vehicle, the following conditions were taken in account:

Vehicle conditions	CS
• Location based on the road situation	
▫ Ramp, crossings or within urban area	A
▫ Highway or secondary roads	N
• Speed	
▫ Velocity < 0 , Driving backwards	A
▫ Velocity < 10 km/hour. This is an indication that caution is required because the driver is searching for parking space, is trying to park the car or it pedestrians and cyclists might be nearby.	H
▫ Velocity > 10 km/hour	N
▫ Velocity = 0 km/hour , standing still	L
• Using brakes and 2 seconds afterwards	A
• Directions signs indicates changing lanes, fog lights or wipers turned on indicates reduced visibility	H

User-applications can raise the required attention of the driver by setting the environment/ambient condition to the applicable concentration mode [A], [H], [N], or [L]. For example:

Environment/ambient conditions	CS
• Collision warning, with optional information about direction (front, left, right, rear)	A
• Accident warning, including the location. If not in proximity or after passing the accident, the message is not relevant and shall not influence the context	H
• Tail of traffic jam	H
• Road works	H

The Context manager is aware of these conditions (context) and will be updated by the Priority manager about the current concentration mode derived from the most restrictive context state.

CONTEXT MANAGER

The Context manager uses a knowledge base to store concepts, relations, and facts. A concept represents an abstract or concrete object that belongs to a class or category that serves as metadata, specifying corresponding actions and values. Facts are specialised concepts with additional information about location, priority and validity information (figure 3). Based on the category and the current state, a corresponding action will be performed.

Information elements are stored as Facts and have a validity and priority. External devices and user-applications can only add Facts to the knowledge base. Adding an information element will result in changing the subject of that provided Fact. Facts cannot be changed and will be deleted by the Context manager when their validity expires.

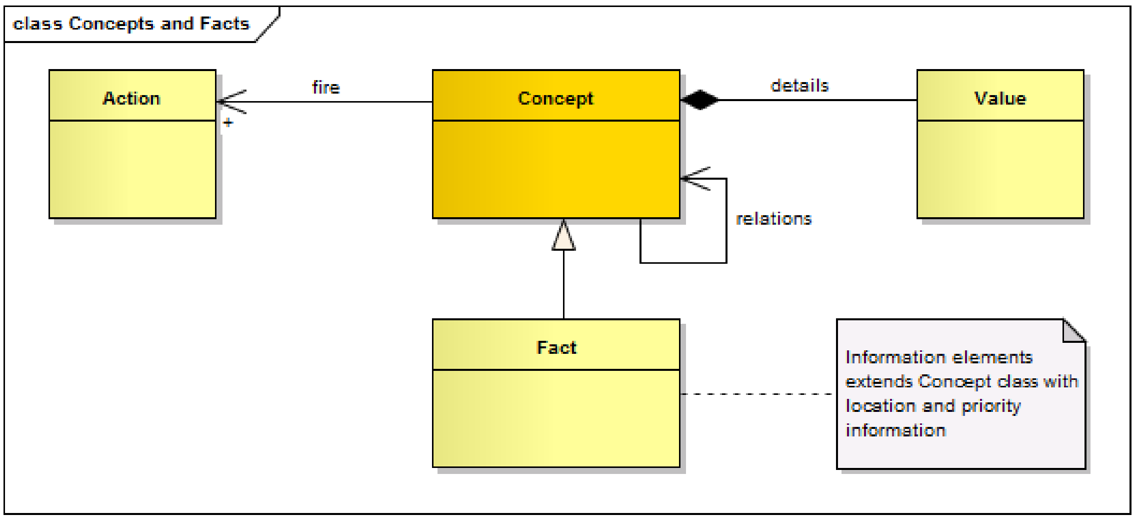
class Concepts and Facts
Action
fire
+
Concept
details
Value
relations
Fact
Information elements extends Concept class with location and priority information

Figure 3. Schematic presentation of the Context manager. A concept represents an abstract or concrete object that belongs to a class or category that serves as metadata, which also specifies a corresponding action and values. Facts are specialised concepts with additional information about location, priority and validity information.

USER INTERACTION

The HMI manager provides a mechanism to handle three interaction modalities: graphical, audio and tactile interaction with the user. First functional information is transformed into presentation information based on templates. Next, interaction channels are addressed according to the prioritised information. Finally, commands are given to external devices to present the information.

The control of the interaction modalities is designed around open standards to facilitate the Open Platform vision of SPITS and can be implemented with existing technology.

In the current study, user interaction for the HMI Manager was designed around the concept of layering, where information can be presented on top of other information. With use of layers, multiple sources can display their information in order of priority.

GRAPHICAL INTERACTION

The Context manager Graphical presentation encompassed the use of visual interaction-devices, such as displays. The HMI manager uses multiple graphical layers for presentation. Each layer supports transparency to allow visibility of layers beneath, and can be switched on or off to show ore hide all information allocated to that layer. Graphical layers are divided in two groups, the managed layers and the free layers. The free layers are directly addressable and located below the managed layers. The free layers are intended for fast presentation by user-applications without intervention of the HMI-manager (figure 4).

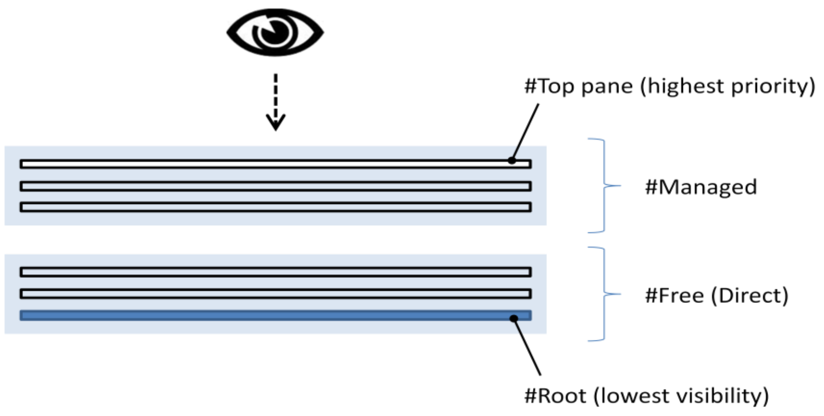

Figure 4. The HMI manager supports multiple graphical layers. Each layer supports transparency to allow visibility of layers beneath, and can be switched on or off to show ore hide all information allocated to that layer.

The managed layers are controlled by the HMI-manager. Instead of direct access, the managed layers are controlled by the Presentation manager, using templates that are invoked by functional information-elements. The top pane is reserved for information elements with the highest priorities.

AUDIO INTERACTION

In our current design, the HMI-manager supports hands-free bi-directional audio interaction. Voice recognition is outside the scope of the HMI-Manager. Audio interaction was divided into the following categories for presentation, processing and prioritisation purposes:

- Background noise: as result of the environmental state of the vehicle;
- Streams: user-requested continues audio streams, such as music and phone calls;
- Short messages: artificial spoken messages;
- Signals: short sounds, such as a repetitive alert or notification.

Similar to the graphical presentation (5.1), the audio channel can be divided in two priority channels. The HMI-manager supports a primary and a secondary audio channel to control the audio focus by regulating the volume. Notifications will be put on the primary audio channel while music, telephone calls and continuous audio streams will be addressed to the secondary channel.

Using a multi-directional speaker system, the driver can also be informed about the position-specific events in relation to the vehicle, e.g. parking sensors or dead-angle warnings (figure 5).

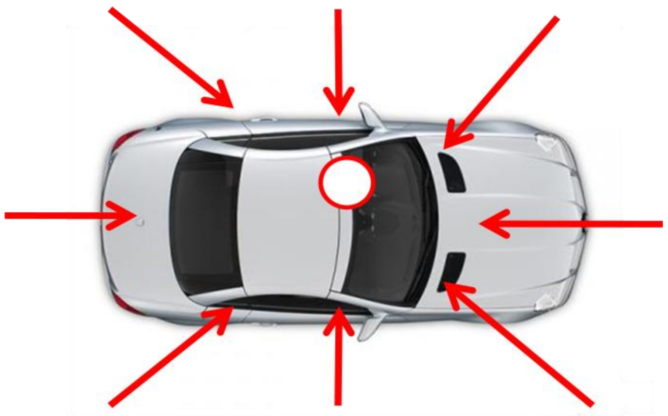

Figure 5. Using a multi-directional speaker system, the driver can also be informed about the position-specific events in relation to the vehicle.

TACTILE INTERACTION

Tactile devices represent the group of controls that can be touched, pressed or turned by the user working as a sensor. In some cases, tactile devices can also be an actuator to inform the user about a status. For example, giving force feedback on the steering wheel or gas pedal to give the driver a better feeling of the situation. Tactile and haptic technology using force-feedback have been successfully used in aviation and are considered as a potential future extension for the HMI Manager.

INTEGRATION

The HMI-manager is the central coordination point for user interaction. It integrates information from different applications into a uniform and consistent look and feel. The HMI-manager is designed to control the interaction of information instead of controlling applications, with the goal to provide a driver-centred unified approach for in-vehicle information presentation.

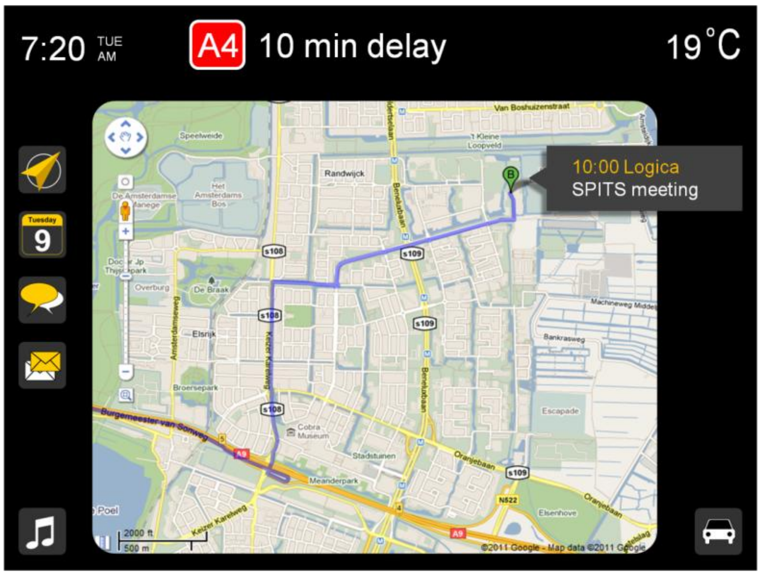

Figure 6: screenshot of the prototype in a "map-oriented view". Different views can be selected using the icons, to display information related to location, time, synchronous communication, asynchronous notifications, audio streams, and to access vehicle controls

A prototype of the HMI-manager has been developed in the SPITS project to demonstrate the usefulness and need for HMI management. Figure 6 shows a screenshot of the prototype showing the "map-oriented view". Different views can be selected using the icons (Figure 6, left), to display information related to location, time, synchronous communication (e.g. phone), asynchronous notifications (e.g. SMS or email), audio (e.g. current music streams), and to access vehicle controls (Figure 6, right).

Our prototype demonstrated a unified form of in-vehicle information presentation that implements information prioritization, originating from heterogeneous applications. The next step will be to implement the HMI-manager inside road vehicles for testing and validation in real-life traffic situations.

CONCLUSION

The innovative contribution of SPITS is the integration of prioritised information into Human Machine Interaction, in order to cope with an ever growing flow of in-vehicle information. By reducing distraction, especially when the situation requires more attention of the driver, the need for controlling the information flow emerges. A proposal for a generic application interface specification is one of the outcomes of the SPITS Architecture design for cooperative vehicles.

In the current paper, we presented a driver-centred unified approach for in-vehicle information management. We described an application-transparent design of an HMI-manager that manages (prioritizes) information from multiple on-board applications, according to vehicle situational-awareness and driver workload.

An architectural design of the HMI-manager was described. We introduced the HMI-manager as a central service of the OBU, and illustrated how information-elements from in-vehicle applications were transformed into a unified information-element format. Next, we described how these information-elements could be prioritised according to importance and urgency, and how these elements were presented to the driver accordingly, using different presentation channels.

A prototype was created to demonstrate the practical feasibility and usefulness of the HMI-manager. Future work will focus on implementation of the HMI-manager into real vehicles for testing and validation in real-life traffic situations.

REFERENCES

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

(2) Jamson, A. H., Merat, N. “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*. 2005. Vol. 8, no. 2, pp. 79-96.

(3) Stoter, A. J. R., Dalmolen, S., Drenth, E., Cornelisse, E., Mulder, W., 2011. “Real-time context aware reasoning in on-board intelligent traffic systems: an architecture for ontology-based reasoning using finite state machines”. *ICAART 2011 conference on Agents and Artificial Intelligence*, (Rome, 2011).