A Framework for Personalised HMI Interaction in ADAS Systems

Yannis Lilis[1], Emmanouil Zidianakis[1], Nikolaos Partarakis[1], Stavroula Ntoa[1]
and Constantine Stephanidis[1,2]
[1]*Institute of Computer Science, FORTH, N. Plastira 100 Vassilika Vouton, GR-700 13, Heraklion, Crete, Greece*
[2]*Department of Computer Science, University of Crete, Voutes Campus GR-700 13 Heraklion, Crete, Greece*
{*lilis, zidian, partarak, stant, cs*}*@ics.forth.gr*

Keywords: ADAS Systems, Personalisation, Adaptation, Human Machine Interaction.

Abstract: Personalisation features of Advanced Driver Assistant Systems (ADAS) can improve safety and driving experience. However, they are typically developed in an ad-hoc, application-specific and vehicle-specific manner, resulting in tightly coupled implementations that are difficult to extend, while disallowing reuse of personalisation code or even personalisation logic across different setups. In this context, this paper proposes a framework for supporting personalised HMI interaction in ADAS systems, developed in the context of the H2020 ADAS&ME project. The framework is based on a rule engine that uses a customisable and extensible set of personalisation and adaptation rules, provided by automotive domain and HMI experts, and evaluates them according to the driver, vehicle and environment to produce HMI activation and GUI personalisation and adaptation decisions. Personalised HMI modality selection is realised by taking into account all available input and output modalities of the vehicle and maintaining bindings for their activation. At the same time, GUI personalisation is handled automatically through a GUI toolkit of personalisable and adaptable user controls that can be used for developing any GUI application requiring personalisation features. The paper presents the design and development of the framework and validates it by deploying it in two case studies.

1 INTRODUCTION

Within the automotive domain, research efforts have recently focused on improving driving safety through the development of preventive support systems, such as adaptive cruise control, automatic emergency braking, lane keeping assist, collectively referred to as Advanced Driver Assistance Systems (ADAS).

To achieve their goal, ADAS systems utilise a variety of Human-Machine Interaction (HMI) elements for interacting with the driver. In this context, a prominent direction for further improving safety and the overall driving experience is to offer personalised and adaptive interaction (Mueller, 2014), taking into account aspects of the driver, the vehicle and the environment to deliver custom tailored interaction through intelligent decision making. For example, an ADAS system may take into account the driver's distraction or lack of experience to trigger proactive actions earlier. Also, the methods for informing the driver about an upcoming road condition may depend on driving context or environmental conditions. For example, in case of bad lighting conditions (e.g., sunlight), an auditory message would be preferred over a visual one.

A lot of research has focused in the field of HMI personalisation in the context of ADAS systems (Amditis et al. 2001; Recarte and Nunes 2003; Héléne et al. 2005; Hassel and Hagen 2006; Brouwer et al. 2009; Fischer and Nürnberger 2010, Garzon and Poguntke 2011; Garzon 2012). However, in most cases the personalisation features are deployed in an ad-hoc manner, focusing on specific applications and vehicle setups. This means that incorporating additional HMI personalisation would require significant effort, leading to a system that is difficult to extend. From a software engineering perspective, ad-hoc solutions also result in tightly coupled implementations where personalisation code is mixed with application logic code, hindering maintainability and disallowing reuse of personalisation code or even domain personalisation logic across different setups.

This paper argues that a comprehensive solution capable of delivering personalisation without being bound on specific setups or particular HMI elements would reduce the extra effort required for realising personalisation, thus greatly improving the adoption of HMI personalisation within ADAS systems. In this context, this paper proposes a framework for supporting personalised HMI interaction in ADAS systems. The framework is based on a rule engine that uses

a customisable and extensible set of personalisation and adaptation rules, provided by automotive domain and HMI experts, and evaluates them according to the current driver, vehicle and environment parameters to produce HMI activation and GUI personalisation and adaptation decisions. Personalised HMI modality selection is realised by taking into account all available input and output modalities of the vehicle and maintaining bindings for their activation. GUI personalisation is handled automatically through a GUI toolkit of personalisable and adaptable user controls that can be used for the development of any GUI application requiring personalisation features. The paper discusses the design and development of the framework and validates it by deploying it in two case studies.

2 BACKGROUND AND RELATED WORK

2.1 ADAS Systems and HMI Elements

ADAS are technologies used to make motor vehicle travel safer by automating, improving or adapting some or all of the tasks involved in operating a vehicle (Craig, 2012). The aim of ADAS systems is to reduce risk exposures by notifying the driver about potential problems, or automating some tasks to relieve a driver from manual control of a vehicle. To this end, ADAS systems provide technologies that monitor the driver state, alert drivers about potential collisions or accidents, implement safeguards and take over control of the vehicle to improve the safety of the driver and the passengers (Tigadi et al., 2016). To deliver such functionality, ADAS systems use input from multiple sensor sources such as ultrasound, automotive imaging, LiDAR, radar and cameras used for image processing by computer vision algorithms (Lu et al., 2005), as well as radio and satellite communications for vehicle-to-infrastructure or vehicle-to-vehicle monitoring (Piao and McDonald, 2008).

HMI elements are essential for ADAS systems as they serve both as a communication bridge between the vehicle and the driver and as a mean for the driver to access information and services provided via the vehicle-to-vehicle or vehicle-to-infrastructure communication. The efficacy of ADAS technologies for driver safety relies on the provision of distraction-free methods for driver-vehicle interaction, with multiple OEMs having already adopted the design and deployment of HMI elements in this direction.

ADAS information exchange is mostly supported by touch input devices as well as visual and auditory forms. Alternative input and output modalities have also been considered, for instance, HMI input may be provided through analysis of implicit gestures and poses. Touch-free HMI elements support interaction without requiring drivers to move their hands from the steering wheel. On the other hand, HMI output is provided explicitly through visual, acoustic and haptic modalities, as well as implicitly using ambient light, background sound and smooth force feedback on the steering wheel, pedals or handle bars. Visual feedback is provided by displays, like clusters on the dashboard, but also through head-up displays (HUDs) projecting information on a screen on the windscreen of the car in the driver sight direction, maintaining their focus on the road (Tonnis et al., 2007). Acoustic feedback can also be used for safe notifications, as audible warnings are more effective than visual ones in case of urgent situations since they induce a quicker reaction. Finally, haptic interaction is typically used on the steering wheel or on the seat and can be less annoying than auditory (Lee et al., 2004).

2.2 Personalised Interaction with HMI Elements in Automotive Applications

The evolution of HMI technologies and the increasing amount of information in the automotive domain has introduced new layers of interaction complexity. Initially, HMI elements were limited only to the primary commands devoted for driving. With the massive introduction of ADAS systems and the emergence of multi-modal HMIs, drivers encounter an increasing information flow. However, drivers are not always capable of perceiving and understanding the plethora of messages due to their physiological state (i.e., tired) and complex traffic environment (Héléne et al., 2005). As a result, HMI technologies need to be context aware in terms of driver, vehicle and environmental state as well as be personalised and adapted to user's characteristics, needs and expectations. There are two main applications for personalisation: HMI element personalisation and personalisation of driver assistance systems. A third application area in hybrid electric vehicles is the prediction of the driving range. Research studies mainly focus on the infotainment area for personalisation of the HMI elements. For example, (Garzon and Poguntke, 2011) present an in-car-entertainment system that automatically adapts to the personal needs of the driver. Various efforts have been made to increase driver's performance and satisfaction through personalised HMI technologies. For instance, (Hassel and Hagen, 2006) proposed a method to build a dialogue system in an

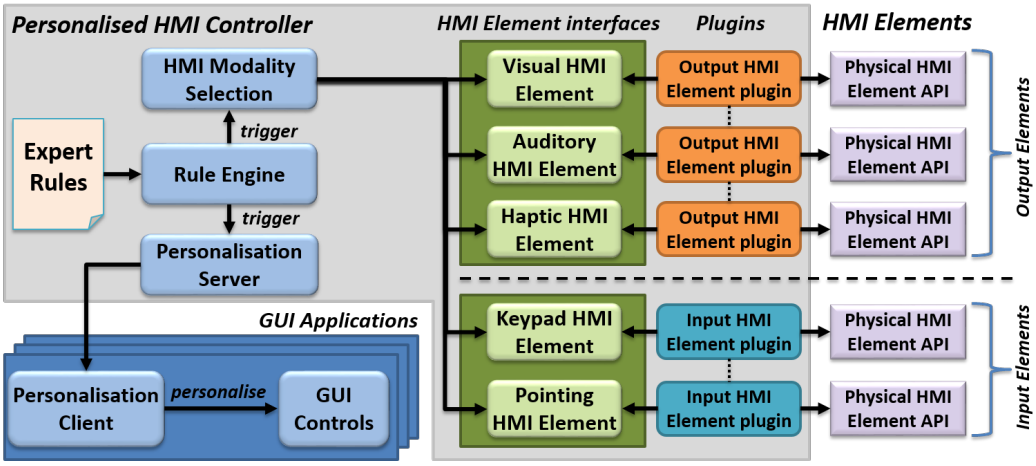

Figure 1: High-level architecture of the personalised HMI framework.

automotive environment that automatically adapts to the user's experience with the system. Another example of personalised interaction with in-car information systems is COMUNICAR (Amditis et al., 2001). It aimed to develop an integrated, in-vehicle multimedia HMI able to harmonize the messages coming from the ADAS systems, the telematics services, and the entertainment functions. Similarly, the AIDE project (Brouwer et al., 2009) investigated the integration of different ADAS systems and in-vehicle information systems that take into account the driver and the traffic conditions to adapt presented information.

(Garzon, 2012) introduces two approaches to simplify the execution of a preferred entertainment feature by personalising a list of context-dependent shortcuts or by automatically executing regularly used features. The myCOMAND case study explores an interactive user interface (UI) that provides access to various information items aggregated from Web services (Fischer and Nürnberger, 2010). It was created to gain insights into applicability of personalisation and recommendation approaches for the visual ranking and grouping of items using interactive UI layout components (e.g., carousels, lists). Mercedes-Benz User Experience offers an intuitive operating experience that learns user preferences via artificial intelligence (Werner, 2018). The system also includes intelligent voice control with natural speech recognition.

The domain of navigation systems is also important, as they are complex, with many functions, and in some cases coexisting with infotainment systems of a car and other components. According to (Recarte and Nunes, 2008), during stressful situations, the HMI of the driver navigation system can be made adaptive to reduce the mental workload of the driver, depending on the driver's characteristics.

3 PERSONALISED HMI FRAMEWORK

The Personalised HMI framework (Figure 1) is an expert system that applies rule-based reasoning to produce the personalisation and adaptation decisions for selecting and activating the most appropriate HMI elements and personalising GUI applications based on the driver profile, current state, driving context and environmental situation. The framework operates in a distributed environment, with the Personalised HMI Controller acting as a server and the HMI modalities and GUI applications acting as clients. It is part of an HMI framework developed in the ADAS&ME project (Knauss et al., 2018). To provide the necessary context, important ADAS&ME components and their interaction with the personalised HMI framework are presented before elaborating on the framework itself.

3.1 Components and Interaction

Apart from the Personalised HMI framework, other important components include the Personalisation System, the Driver Monitoring, the Environmental Monitoring and the Decision Support System (DSS).

The Personalisation System provides information about the driver (e.g., characteristics, preferences), while the Driver and Environmental Monitoring components provide information about the current driver state and environment. All this information is organised in a comprehensive ontology (Lilis et al., 2017) and constitutes the input parameters used for expressing the personalisation and adaptation logic.

The DSS acts as the "brain" of the vehicle and decides the appropriate interaction or transition strategy based on the situation, thus initiating interaction

with the driver. In particular, the DSS decides to inform, notify or alert the driver as well as request handovers or trigger automation functions when necessary. Interaction strategies involving HMI modalities are realised through the personalised HMI framework so as to support personalised interaction. Activation of automated functions similarly passes through the personalised HMI framework to be presented to the driver in a personalised manner.

3.2 HMI Modality Selection

Based on the vehicle HMI elements, the available input devices, either explicit (e.g., touch, gestures, speech, dashboard controls) or implicit (gaze/head tracking, physiological parameters) should be considered for choosing the best input method. Similarly, the available output devices (e.g., touch screens, speakers, vibration motors, AR displays) and their respective modality types (i.e. auditory, haptic, and visual) should be considered for choosing the best output method. To achieve this functionality, the personalised HMI framework classifies modalities into categories and supports multimodality.

3.2.1 Output Modalities

Output HMI elements are classified in three categories: visual, auditory and haptic. The proposed framework features corresponding HMI element interfaces so as to manage and access physical HMI elements installed in the vehicle in a uniform, device-independent manner. For each physical HMI element, an HMI element plugin is developed to wrap its low-level API to the appropriate HMI element interface. These plugins are instantiated upon system start-up based on configuration that matches the HMI element setup of the vehicle, and then registered to an HMI element holder. Through the HMI output element interfaces, the HMI elements of the vehicle can be triggered, activated or deactivated, while it is also possible to query for their status and capabilities. The HMI element status can be used to detect failing HMI elements so as to select alternative interaction modalities. HMI element capabilities can also be taken into account in specifying the adaptation and personalisation logic. For example, to visualize a notification message with a lot of content, the system would take into account the screen size of each visual HMI element and possibly opt for a big display.

3.2.2 Input Modalities

Input HMI modalities are categorised in keypad and pointing elements. The first category covers elements with hardware buttons that post distinct codes per key press (e.g., dashboard buttons), but also composite systems that process input streams and produce high-level command (e.g., speech recognition). The second category abstracts over physical pointing methods like relative or absolute pointing (e.g. touch).

Support for multimodality is based on an architectural split between *virtual input*, i.e. at the level of the HMI element interface, and *physical input*, i.e. at the level of the physical input device, again wrapped as an HMI element plugin. Each HMI input element interface abstracts over multiple physical input elements of its category and operates as a high-level input channel triggering interaction commands, while physical input plugins map device commands to interaction commands. Application logic handles input at the interaction command level, is independent from physical elements and supports any vehicle setup.

Input modality selection is also subject to change based on the personalisation and adaptation logic. For example, voice commands would be preferred over dashboard buttons when driving at high speed. On the contrary, voice recognition would be restricted in a loud environment so as to reduce recognition errors.

3.3 GUI Personalisation

Personalising GUI applications typically involves developing dedicated interface screens and associated interaction on top of the typical GUI. This supports personalisation at any level of granularity, but entails extra development effort, and may lead to code replication as well as tight-coupling of application logic with personalisation logic. For many GUI applications, a more course-grained personalisation approach that supports specific GUI elements and specific forms of personalisation, is often sufficient.

In this context, the proposed framework features a GUI toolkit of adaptive and personalisable user controls to be used in GUI application development. The toolkit integrates personalisation capabilities and abstracts these features from developers, enabling easier integration and robust rapid prototyping.

3.3.1 GUI Toolkit User Controls

The GUI toolkit features custom user controls (widgets) that extend their native counterparts with personalisation support, including both low-level (e.g., labels, buttons, checkboxes, etc.) and high-level (e.g., tab groups, list views) widgets. Both user control categories maintain the native user control interface so that once instantiated (via code or visual designer), the user control can then be used instead of its native counterpart in GUI development code.

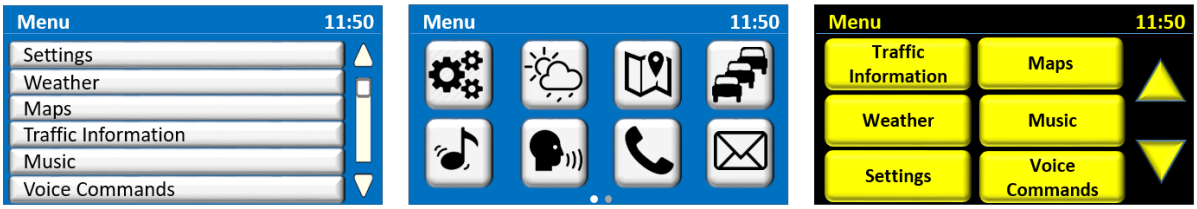

Figure 2: Alternative menu incarnations; default style (*left*), for computer experts (*middle*), for visually impaired users (*right*).

Low-level widgets are typically personalised in terms of visual style aspects such as fonts and colours. Supporting specific styles for particular widgets, or widgets collections is achieved in a way similar to CSS styling, i.e. assigning ids or classes to widgets to allow matching them against personalisation rules. High-level widgets are designed and developed based on notion of adaptive component hierarchies. Abstract user interface tasks are hierarchically decomposed to sub-tasks, which at the lowest level are matched by physical interface designs. Multiple physical designs are available per task, allowing for a polymorphic matching process that yields the best combination of physical designs based on the personalisation parameters. Technically, an abstract GUI control is used that is instantiated at runtime through a factory method to produce the concrete personalised GUI control. For example, Figure 2 shows alternative incarnations of a personalised menu widget.

The toolkit is extensible, allowing to introduce custom, application specific user controls building on top of the native ones. The toolkit facilitates the development of such widgets also as task-based adaptive component hierarchies, guiding the developer to introduce application specific tasks and develop the alternative matching physical interfaces.

3.3.2 Personalising User Controls

Personalisation of user controls is achieved through the *Personalisation Server* and the *Personalisation Client*. The former operates as a server within the central *Personalised HMI Controller*, while the latter is automatically included by the toolkit within each GUI application that maintains a list of all GUI toolkit user controls instantiated in the application. Upon launching the GUI application, the Personalisation Client registers itself to the Personalisation Server to be notified about any personalisation decisions. When notified, the Personalisation Client propagates these decisions to each user control by invoking their custom personalisation interface, styling them as needed and triggering the appropriate physical interfaces to match active tasks. This is done transparently and involves no developer effort.

3.4 Rule Engine for Triggering Personalisation Behaviour

The central component of the framework is a rule engine that uses a set of personalisation and adaptation rules provided by domain experts, i.e. driver behaviour and HMI experts. The rules are specified in ACTA (Zidianakis et al., 2017), a general purpose finite state machine description language that supports event-driven sequential logic as well as rule-driven workflows. The specified rules are evaluated according to the current input data to produce HMI activation decisions, handled by the HMI modality selection component, as well as GUI personalisation decisions, propagated by the GUI Personalisation Server to all GUI applications running in the vehicle.

As mentioned, interaction is initiated by the DSS, with each request handled by the Personalised HMI framework by directly triggering corresponding rules, thus following a well-defined interface dictated by the interaction strategies. The scheme is flexible enough to also support finer-grained interaction including personalising the start-up process (e.g., put on the news or music), selecting the destination and the preferable route (e.g., based on history), or stopping at a gas station (e.g., based on route proximity).

4 CASE STUDIES

4.1 Simulated Car Environment

This case study was selected as it enabled experimenting with a wide variety of HMI elements (developed as software plugins) without involving any hardware integration issues. The HMI elements included: (i) a central stack screen; (ii) a head-up display (HUD); (iii) a LED strip; (iv) an audio system; (v) a steering wheel with vibration; and (vi) a seat with vibration.

Many personalisation and adaptation parameters were considered covering driver preferences (e.g., GUI visual styles), experience (e.g., driving or computer experience), disabilities (e.g., low vision, hearing impaired), current state (e.g., sleepy, distracted)

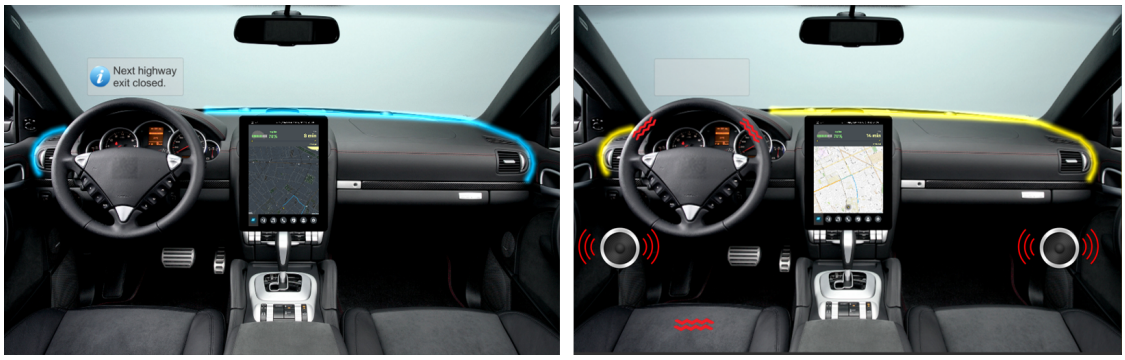

Figure 3: Examples of personalised and adapted HMI interaction; issuing an information message for a typical driver in low lighting conditions (*left*), issuing a warning message for a visually impaired driver in normal conditions (*right*).

and state history (e.g., usually sleepy), as well as driving context (e.g., current speed, automated or manual driving) and environment (e.g., environment noise, visual conditions). Based on these parameters, a set of rules was defined and developed to trigger personalised HMI interaction. Figure 4 shows an excerpt of these rules, while Figure 3 presents examples of the respective HMI interaction realised for specific parameter combinations. As shown, information messages are associated with a blue flashing colour on the LED. Driver alerts instead use a yellow LED colour, accompanied by a vibration either on the steering wheel, if the driver's hands are on it, or the seat otherwise. Both information messages and alerts also issue a message to be presented to the driver. This is further personalised considering driver disabilities; for a typical driver the message is presented on the HUD, while for a low vision driver spoken text is used instead. The rule excerpt also covers a case of GUI personalisation that takes into account the lighting conditions to appropriately activate a dark or a light visual mode.

```
@msg := params.context.message
when (LowLighting) { GUI.SetMode("darkMode"); }
when (!LowLighting) { GUI.SetMode("lightMode"); }
State "Idle" {
 LED.off();
 when(ShouldInformDriver) { NextState = "Inform"; }
 when(ShouldAlertDriver) { NextState = "Alert"; }
}
State "Message" {
 when (params.driver.lowVision)
  { speakers.SpeakText(@msg); NextState = "Idle"; }
 when (!params.driver.lowVision)
  { HUD.DisplayText(@msg); NextState = "Idle"; }
}
State "Alert" {
 LED.Flash("yellow");
 when (params.driver.handsOnSteeringWheel)
  { steeringWheel.Vibrate(); NextState = "Message"; }
 when (!params.driver.handsOnSteeringWheel)
  { seat.Vibrate(); NextState = "Message"; }
}
State "Inform"
  { LED.Flash("lightblue"); NextState = "Message"; }
```

Figure 4: Excerpt of personalisation and adaptation rules for case study 1, matching the interaction of Figure 3.

Overall, the framework was quite effective in supporting personalised HMI. It achieved decoupling of personalisation logic from application code and promoted a clear interface between realising DSS interaction strategies and triggering HMI elements. The rule language also enabled non-technical HMI experts to contribute in the development process and facilitated the prototyping of personalised interaction.

4.2 Electric Car

The personalised HMI framework has been deployed and integrated in Use Case B of ADAS&ME (Figure 5), which aims to mitigate range anxiety in electric vehicles. The HMI elements in this case study consisted of: (i) a central stack screen; (ii) an audio system; and (iii) a pair of A-pillar LED strips. Again, various parameters were considered to derive the personalisation and adaptation rules, with focus on accessibility issues (e.g. colour blindness, low vision, auditory impairments, etc.). Parameters also included driver characteristics such as language, preferences and experience, driver state, (i.e. anxiety), driving context (e.g., low battery) and environment (e.g., low lighting).

Since the central stack screen was the main HMI element of the case study, particular focus was put on GUI personalisation and adaptation. Many interface screens were personalised with Figure 7 illustrating

Figure 5: Integration of the personalised HMI framework and the central stack GUI application in an electric vehicle.

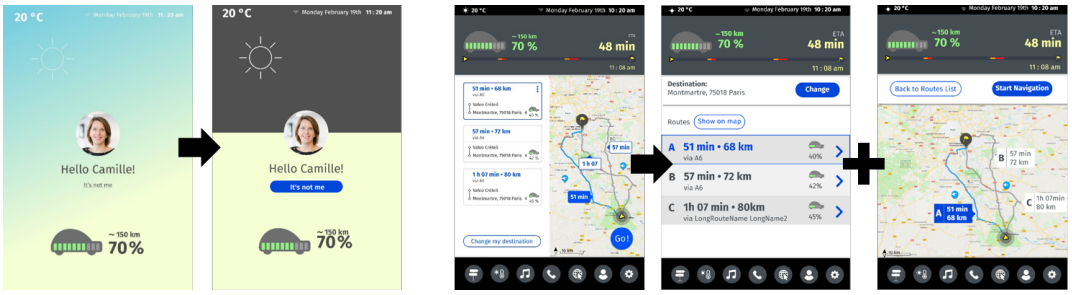

Figure 6: GUI personalisation and adaptation examples: changing visual styles (*left*), layout and presentation (*right*).

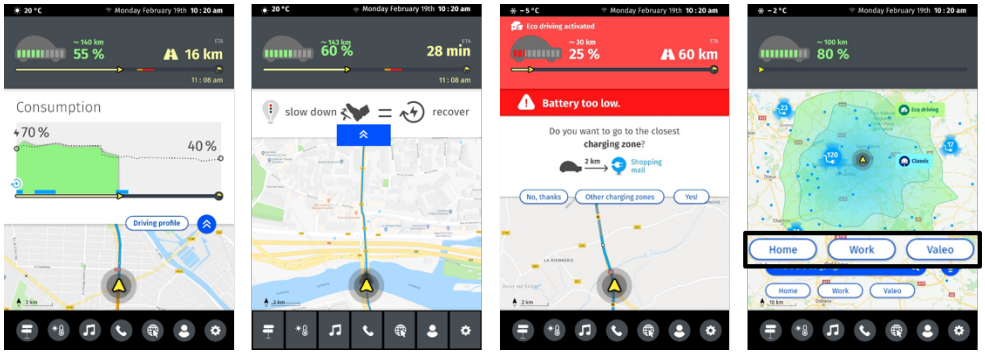

Figure 7: HMI modality personalisation and adaptation examples: adaptation for range anxiety (*left*); personalised interface and notification for inexperienced electric vehicle driver with motor impairments (*middle left*); adaptive interface suggestions based on electric vehicle range (*middle right*); personalised input controls based on recent/frequent driver destinations (*right*).

two indicative examples. The first one (left) shows the personalisation of visual styles such as colours, styles and fonts, used extensively for accessibility issues and driver preferences. The second one (right) shows the adaptation of layout and presentation for elderly drivers, drivers with low vision or drivers with high cognitive load. As shown, the joint task of selecting and reviewing the route is split into two separate tasks, each performed in a different screen so as to focus only on the specific task information, and allow presenting it with a larger font-size and clearer layout, facilitating interaction for the target driver groups. This category of adaptations was realised by introducing application specific tasks and developing the alternative physical interfaces matching the tasks. Finally, the middle left part of Figure 6 also presents a visual style adaptation for drivers with mild motor impairments that maximizes the effective touch area.

The central stack screen was also used as an input and output modality, featuring also HMI modality personalisation and adaptation examples, as shown in Figure 6. When the driver is detected to have range anxiety the HMI shows the expected and actual consumption to calm the driver (Figure 6, left). Then, notifications with electric vehicle information (e.g., slow down to recover battery) are presented for inexperienced drivers (Figure 6, middle left). Adaptive interface suggestions are also provided based on electric vehicle range, e.g., to suggest going to the closest charging station when battery level is too low (Figure 6, middle right). Finally, personalised input controls are used for destination selection based on drivers' recent or frequent destinations (Figure 6, right). HMI modality personalisation examples were not limited to the central stack screen; louder sounds were also used for drivers with hearing disabilities, while varying colours and patterns were adopted in the LED strips to convey driving context information.

Again, the framework proved to be very effective, especially in GUI personalisation. Personalisation for common UI controls such as round or square buttons, or changes in text size, were specified only once based on GUI toolkit styling and didn't involve repeated effort despite occurring across different screens. In order to personalise application-specific GUI parts (e.g., route selection) some development was naturally required, as would also have been without the framework. Nevertheless, the structured approach of the GUI toolkit for specifying tasks and developing the matching interfaces clearly separated original and adapted interface code from interaction code, facilitating code organisation. It also promoted reusability, enabling to reuse the widget in different aspects of the application or even across applications.

5 CONCLUSIONS

This paper presented a framework for personalised interaction in ADAS systems, taking into account the driver profile and state as well as the situational and environmental context. The framework relies on a rule engine that uses a customisable and extensible set of personalisation and adaptation rules to produce the decisions for selecting the most appropriate HMI elements and personalising GUI applications.

Personalised HMI modality selection is realised by taking into account all input and output modalities of the vehicle, classifying them into categories associated with HMI element interfaces and maintaining bindings for their activation. Multimodal input is also supported by separating virtual input and the related application interaction commands from physical input, and by allowing to connect multiple physical input devices to each HMI element interface.

GUI application personalisation is achieved through an extensible GUI toolkit of adaptive and personalisable user controls that is offered by the framework to be employed in the development of applications requiring personalisation features. The toolkit integrates personalisation and adaptation capabilities thus abstracting these features from developers of software for automotive applications.

The framework has been developed and adopted in two case studies to validate its applicability. As the ADAS&ME project progresses, the framework will be further integrated in demonstrator vehicles and eventually evaluated in the project pilot sites along with the HMI interaction and its personalisation and adaptation features. Future work includes the exploration of a machine learning approach for triggering HMI personalisation that also considers driver feedback and responses to prior system actions.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 688900 (ADAS&ME). The original design for the HMI of Use Case B of ADAS&ME was conducted by Valeo.

REFERENCES

Amditis, A., Bekiaris, E., Montanari, R., Baligand, B., et al. (2001). An innovative in-vehicle multimedia HMI based on an intelligent information manager approach: the Comunicar design process. In *8th World Congress on Intelligent Transport Systems*.

Brouwer, R. F., Hoedemaeker, M., and Neerincx, M. A. (2009). Adaptive interfaces in driving. In *FAC 2009*, pages 13–19. Springer Berlin Heidelberg.

Craig, J. (2012). Map Data for ADAS. In *Handbook of Intelligent Vehicles*, pages 881–892. Springer London.

Fischer, P. and Nürnberger, A. (2010). myCOMAND automotive user interface: Personalized interaction with multimedia content based on fuzzy preference modeling. In *UMAP 2010*, pages 315–326.

Garzon, S. and Poguntke, M. (2011). The personal adaptive in-car HMI: integration of external applications for personalized use. In *UMAP 2012*, pages 35–46.

Garzon, S. R. (2012). Intelligent In-Car-Infotainment Systems: A Contextual Personalized Approach. In *IE 2012*, pages 315–318.

Hassel, L. and Hagen, E. (2006). Adaptation of an automotive dialogue system to users' expertise and evaluation of the system. *Language resources and evaluation*, 40(1):67–85.

Héléne, T. V., Thierry, B., et al. (2005). Development of a driver situation assessment module in the AIDE project. *IFAC Proceedings Volumes*, 38(1):97–102.

Knauss, A., Diederichs, F., Wilbrink, M., et al. (2018). An HMI Framework for Driver/Rider States Adaptive Transition and ADAS. In *25th ITS World Congress*.

Lee, J. D., Hoffman, J. D., and Hayes, E. (2004). Collision warning design to mitigate driver distraction. In *CHI 2004*, pages 65–72. ACM.

Lilis, Y., Zidianakis, E., Partarakis, N., Antona, M., and Stephanidis, C. (2017). Personalizing HMI Elements in ADAS Using Ontology Meta-Models and Rule Based Reasoning. In *UAHCI 2017*, pages 383–401.

Lu, M., Wevers, K., and Van Der Heijden, R. (2005). Technical feasibility of advanced driver assistance systems (ADAS) for road traffic safety. *Transportation Planning and Technology*, 28(3):167–187.

Mueller, M. (2014). Deficiency drive. *Vision Zero International*.

Piao, J. and McDonald, M. (2008). Advanced driver assistance systems from autonomous to cooperative approach. *Transport Reviews*, 28(5):659–684.

Recarte, M. A. and Nunes, L. M. (2008). Mental workload while driving: effects on visual search, discrimination, and decision making. *Journal of experimental psychology: Applied*, 9(2):119.

Tigadi, A., Gujanatti, R., and Gonchi, A. (2016). Advanced Driver Assistance Systems. *International Journal of Engineering Research and General Science*, 4(3):2091–2730.

Tonnis, M., Lange, C., and Klinker, G. (2007). Visual Longitudinal and Lateral Driving Assistance in the Head-Up Display of Cars. In *ISMAR 2007*, pages 91–94.

Werner, K. (2018). Five Short Display Stories from CES 2018. *Information Display*, 34(2):28–34.

Zidianakis, E., Antona, M., and Stephanidis, C. (2017). ACTA: A general purpose Finite State Machine (FSM) description language for smart game design. In *IHCI 2017*, pages 143–150. IADIS Press.