Multimodal Signal Processing and Interaction for a Driving Simulator: Component-based Architecture

A. Benoit, L. Bonnaud, A. Caplier
Institut National Polytechnique de Grenoble, France, LIS Lab.

Y. Damousis, D. Tzovaras
Centre for Research and Technology Hellas – Thessaloniki, Greece, IT Institute

F. Jourde, L. Nigay, M. Serrano
Université Joseph Fourier, Grenoble 1, France, CLIPS Lab.

L. Lawson
Université Catholique de Louvain, Belgium, TELE Lab.

***Abstract*— After a first workshop at eNTERFACE 2005 focusing on developing video-based modalities for an augmented driving simulator, this project aims at designing and developing a multimodal driving simulator that is based on both multimodal driver's focus of attention detection as well as driver's fatigue state detection and prediction. Capturing and interpreting the driver's focus of attention and fatigue state will be based on video data (e.g., facial expression, head movement, eye tracking). While the input multimodal interface relies on passive modalities only (also called attentive user interface), the output multimodal user interface includes several active output modalities for presenting alert messages including graphics and text on a mini-screen and in the windshield, sounds, speech and vibration (vibration wheel). Active input modalities are added in the meta-User Interface to let the user dynamically select the output modalities. The driving simulator is used as a case study for studying software architecture for multimodal signal processing and multimodal interaction using two software component-based platforms, OpenInterface and ICARE.**

***Index Terms*— Attention level, Component, Driving simulator, Facial movement analysis, ICARE, Interaction modality, OpenInterface, Software architecture, Multimodal interaction.**

I. INTRODUCTION

THE project aims to study component-based architecture using two platforms, namely OpenInterface [1] and ICARE [2] [3], for combining multimodal signal processing analysis and multimodal interaction. OpenInterface is a component-based platform developed in C++ that handles distributed heterogeneous components. OpenInterface supports the efficient and quick definition of a new OpenInterface component from an XML description of a program. By so doing, any program can be included as an OpenInterface component and can then communicate with any other existing OpenInterface component. As opposed to OpenInterface, ICARE is a conceptual component model for multimodal input/output interaction [2]. One implementation of the ICARE model is defined using JavaBeans components [3].

In this project, we study the development of a multimodal interactive system using the OpenInterface platform while the component architecture is along the ICARE conceptual model. The selected case study for this project is a driving simulator [4].

The structure of the paper is as follows: first we present the selected case study by explaining the rationale for selecting this interactive system from a multimodal interaction point of view and by giving an overview of the interactive system. We then recall the key points of the two platforms, OpenInterface and ICARE before presenting the software architecture along the ICARE conceptual model. We then detail the software architecture that has been implemented followed by a discussion on the tradeoffs and differences with the initial ICARE architecture.

II. CASE STUDY: DRIVING SIMULATOR

A. Rational for selecting a driving simulator

The case study is a driving simulator. Indeed, facing the sophisticated sensing technology available in modern cars, multimodal interaction in cars constitutes a very challenging

This report, as well as the source code for the software developed during the project, is available online from the eNTERFACE'05 web site: www.enterface.net.

This research was partly funded by SIMILAR, the European Network of Excellence on Multimodal Interfaces, during the eNTERFACE'06 Workshop in Dubrovnik, Croatia.

domain. The key issue in terms of interaction design is that the main task of the user is the driving one, a critical task which requires a driver to keep her/his eyes on the road. A driving task relies on local guidance that includes sub-tasks involving control of the vehicle and knowledge of the environmental situation. In this context of a driving task, our application domain, our goals are:

- to capture a driver's focus of attention,
- to capture a driver's state of fatigue,
- to predict a driver's state of fatigue,
- to design and develop an output multimodal user interface for presenting alert messages to the driver.

Several projects focus on User Interfaces (UI) in cars and involve various interaction technologies such as trackpad fixed on the steering wheel [5], dedicated buttons, mini-screens as well as head-up display (HUD) technology. For example HUDs are used for displaying icons and texts, usually found on the dashboard of a car, in the windshield as shown in Figure 1.

Fig. 1. In-car HUD (from [5]).

We distinguish two main classes of UI studies in cars: design of interactive dashboards that nowadays include a screen (e.g., graphical user interface for controlling the radio and so on) and Augmented Reality (AR) visualizations. Several on-going projects focus on Augmented Reality (AR) visualizations for the driver using head-up display (HUD) technology. For example for displaying navigation information or for guiding the driver's attention to dangerous situations, transparent graphics (e.g., transparent path of the route) are directly projected onto the windshield [6] as shown in Figure 2, making it possible for the driver to never take her/his eyes off the road.

Fig. 2. In-car Augmented Reality: Guiding driver's attention to dangerous situation. The arrow indicates the position of imminent danger (from [6]).

Complementary to these projects, our task focuses on supporting the driving activity by monitoring and predicting the state of the driver (attention and fatigue). Instead of focusing on external dangers (e.g. a potential collision with a car coming from behind as in Figure 2), the project aims at detecting dangerous situations due to the driver's fatigue state and focus of attention. From the Human-Computer Interaction point of view, the project focuses on multimodal input and output interaction that combines passive input modalities (implicit actions of the driver) for detecting dangerous situations as well as active modalities (explicit actions of the driver) for perceiving alarms (output active modalities) and for changing the output modalities (input active modalities).

B. *Overview of the driving simulator*

Starting from the programs developed during a first workshop at eNTERFACE 2005 [4], the overall hardware setting of the driving simulator includes:

- 3 PCs: one under Windows for the driving simulator, one under Linux for capturing and predicting the driver's states (focus of attention and state of fatigue), and one on Windows for the output user interface developed using the ICARE platform (JavaBeans component).
- 1 LOGITECH webcam sphere
- 1 LOGITECH force feedback wheel
- 1 video-projector
- 2 loudspeakers

Figure 3 shows the system in action. For software, the driving simulator we used is the GPL program TORCS [7] and the multimodal interaction is developed using the two platforms OpenInterface and ICARE.

Fig. 3. Multimodal driving simulator: demonstrator in use.

III. COMPONENT PLATFORMS

A. *OpenInterface platform*

OpenInterface is a component-based platform developed in C++ that handles distributed heterogeneous components. OpenInterface supports the efficient and quick definition of a new OpenInterface component from an XML description of a program. Although the platform is generic, in the context of the SIMILAR project, the OpenInterface platform is dedicated to multimodal applications. We define a multimodal application as an application that includes multimodal data processing and/or offers multimodal input/output interaction to its users.

Figure 4 gives an overview of the platform. Each component is registered in OpenInterface Platform using the Component Interface Description Language (CIDL) and described in XML. The registered components properties are retrieved by the Graphic Editor (Java). Using the editor the user can edit the component properties and compose the execution pipeline (by connecting the components) of the multimodal application. This execution pipeline is sent to the OpenInterface Kernel (C/C++) to run the application.

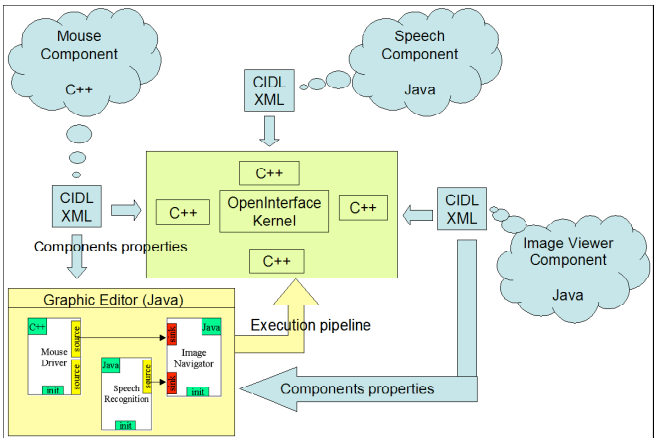

Fig. 4. Overview of the OpenInterface platform.

OpenInterface is designed to serve three levels of users: programmers, application designers (AD) and end-users. Programmers are responsible for the development and integration of new components into the platform. The application designers focus on end-user's needs and are aware of the resources provided by the platform. The AD will use the graphical editor to assemble components in order to develop a multimodal application. End-users interact with the final application whose components are executed within the platform.

B. *ICARE platform*

ICARE (Interaction CARE -Complementarity Assignment, Redundancy and Equivalence-) is a component-based approach which allows the easy and rapid development of multimodal interfaces [2] [3]. The ICARE platform enables the designer to graphically manipulate and assemble ICARE software components in order to specify the multimodal interaction dedicated to a given task of the interactive system under development. From this specification, the code is automatically generated. The currently developed ICARE platform that implements a conceptual component model that describes the manipulated software components, is based on the JavaBeans technology [8]. The ICARE conceptual model includes:

1. Elementary components: Such components are building blocks useful for defining a modality. Two types of ICARE elementary components are defined: Device components and Interaction Language components. We reuse our definition of a modality [9] as the coupling of a physical device d with an interaction language L: <d, L>. In [10], we demonstrate the adequacy of the notions of physical device and interaction language for classifying and deriving usability properties for multimodal interaction and the relevance of these notions for software design.

2. Composition components: Such components describe combined usages of modalities and therefore enable us to define new composed modalities. The ICARE composition components are defined based on the four CARE properties [10]: the Complementarity, Assignment, Redundancy, and Equivalence that may occur between the modalities available in a multimodal user interface. We therefore define three Composition components in our ICARE conceptual model: the Complementarity one, the Redundancy one, and the Redundancy/Equivalence one. Assignment and Equivalence are not modeled as components in our ICARE model. Assignment and Equivalence are not modeled as components in our ICARE model. Indeed, an assignment is represented by a single link between two components. An ICARE component A linked to a single component B implies that A is assigned to B. As for Assignment, Equivalence is not modeled as a component. When several components (2 to n components) are linked to the same component, they are equivalent. As opposed to ICARE elementary components, Composition components are generic in the sense that they are not dependent on a particular modality.

The two ICARE composition components, Complementarity and Redundancy/Equivalence have been developed in C++ as connectors within the OpenInterface platform.

In the following section, examples of ICARE component assemblies are provided in the context of the multimodal driving simulator.

IV. Software Architecture of the Multimodal Driving Simulator

In this section, we first present the overall architecture along the ICARE conceptual model that we defined at the beginning of the project followed by the implemented architecture developed during the workshop. We finally conclude by a discussion of the tradeoffs and differences between the initial conceptual architecture and the implemented one.

A. *ICARE conceptual architecture*

In Figure 5, we present the overall software architecture of the entire multimodal driving simulator in order to highlight the scope of the code organized along the ICARE conceptual model. As pointed out in Figure 5, within the architecture, we identify two types of link between the ICARE components and the rest of the interactive system:

- For inputs, the connection between the ICARE Input components and the rest of the interactive system is at the level of the elementary tasks. From explicit or implicit actions performed by the driver (i.e., the user) along various modalities, the ICARE components are responsible for defining elementary tasks that are

independent of the used modalities. Such elementary tasks are then transmitted to the Dialogue Controller. One example of an driving task is the "accelerate" task.

- For outputs, the Dialogue Controller is sending elementary presentation tasks to the ICARE output components that are responsible for making the information perceivable to the driver along various output modalities. One example of an elementary task is the "present alarm" task.

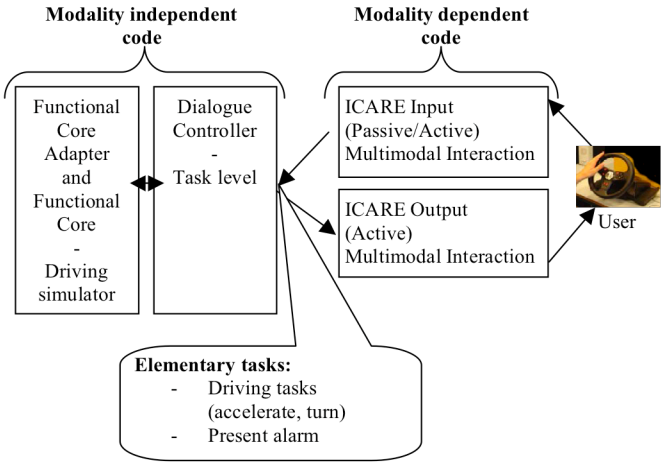

Fig. 5. Overall architecture of the multimodal driving simulator.

Because we reuse the GPL driving simulator TORCS [7] that we extend to be multimodal, some parts of the architecture of Figure 5 are already developed. Figure 6 shows the code that we need to develop along with the existing TORCS code. All the modalities for driving (input modalities based on the steering wheel and the pedal) and for displaying the graphical scene are reused and not developed with ICARE components.

To better understand the extensions to be developed, Figure 7 presents the task tree managed by the Dialogue Controller. Within the task tree, the task "Choose output modalities" does not belong to the main Dialogue Controller of the driving simulator but rather belongs to a distinct Dialogue Controller dedicated to the meta User Interface (meta UI) as shown in Figure 8. Indeed the meta UI enables the user to select the modalities amongst a set of equivalent modalities. Such a task, also called an articulatory task, does not correspond to a task of the driving simulator itself. The meta UI includes a second Dialogue Controller (Dialogue Controller (2) in Figure 8) as well as ICARE input components for specifying the selection. The selection is then sent by the second Dialogue Controller to the ICARE output components [11].

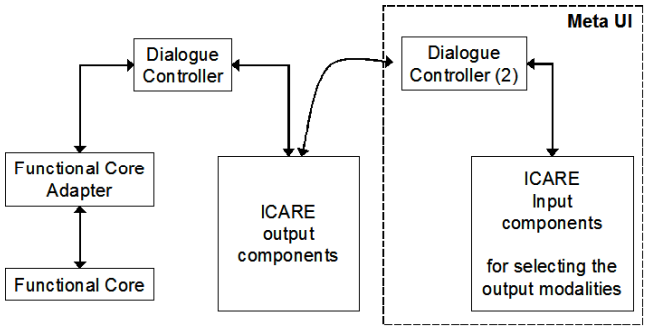

Fig. 8. Meta User Interface: ICARE components within an overall software architecture of an interactive system and the meta UI that enables the selection of equivalent modalities by the user (from [11]).

To obtain the final ICARE architecture, for each elementary task of Figure 7, an ICARE diagram is defined. Figure 9 presents the four ICARE diagrams designed for the four elementary tasks to be developed.

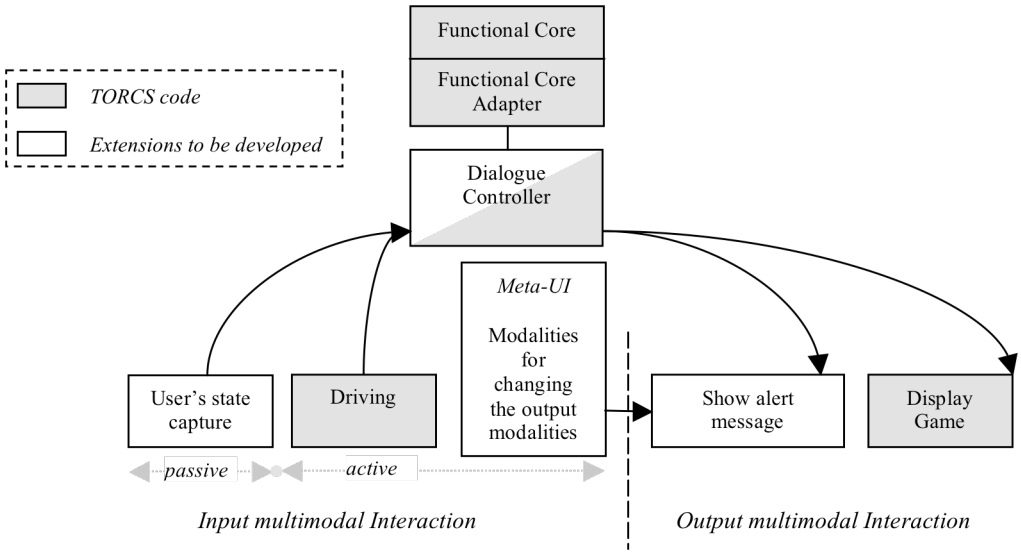

Fig. 6. TORCS code and extensions to be developed within our architecture.

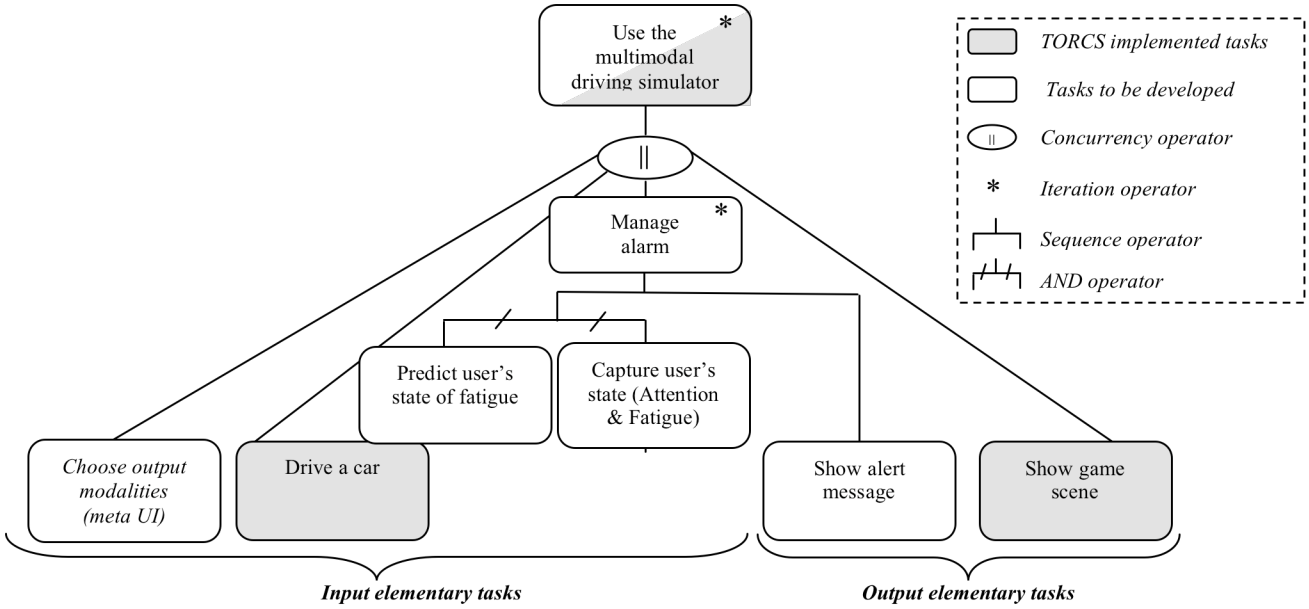

Fig. 7. Hierarchical Task Analysis (HTA): Task tree corresponding to the Dialogue Controller.

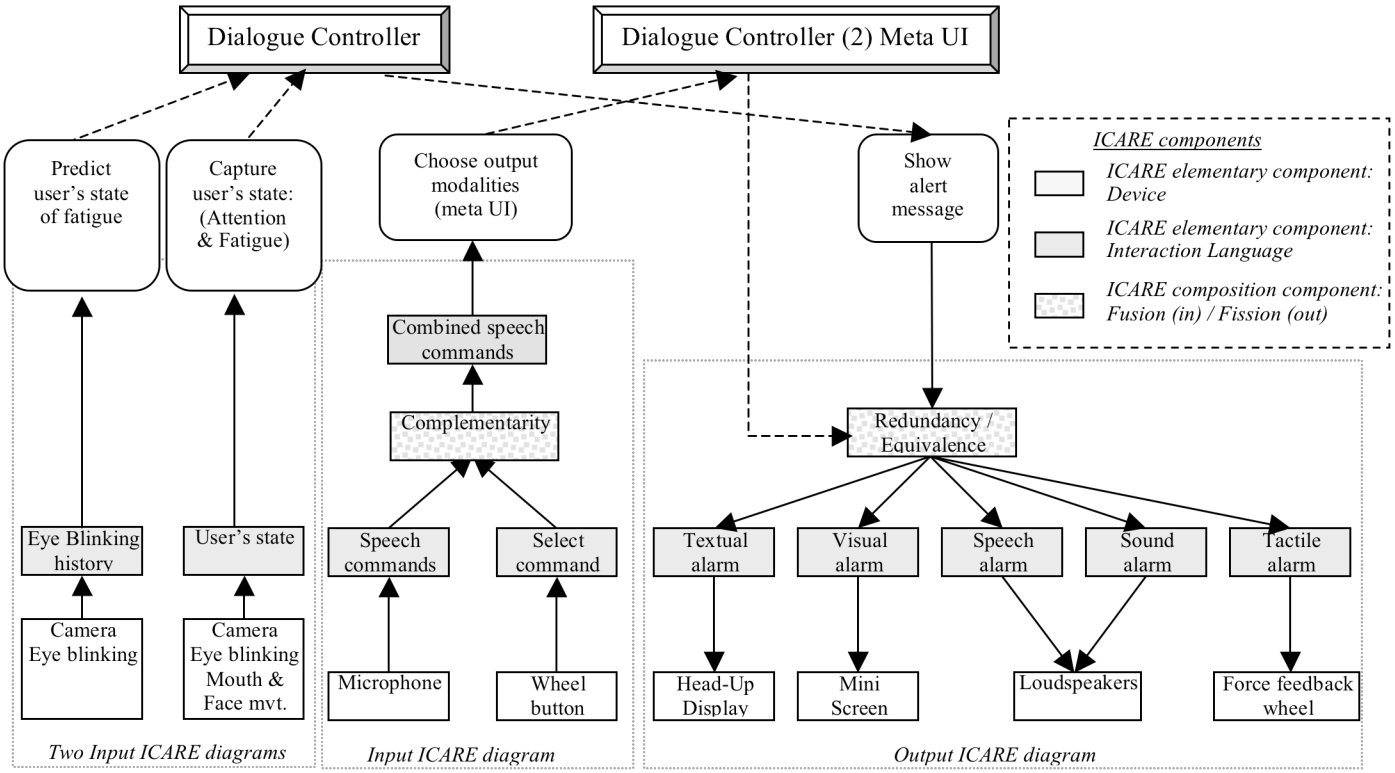

Fig. 9. ICARE diagrams for the elementary task of Figure 7.

The ICARE diagrams for the multimodal driving simulator include pure modalities and two composition components.

- For input, pure modalities made of a device and an interaction language components are used for the two tasks; (i) capture the user's state of fatigue and attention and (ii) predict the user's state of fatigue. These two modalities are passive input modalities. The modality for capturing the user's state is based on eye blinking and mouth movement (yawning) for detecting the state of fatigue and on face movement for capturing the focus of attention. Instead of one pure modality, we can also define three modalities, one for the state of

fatigue based on mouth movement, one for the state of fatigue based on eye blinking and one for the focus of attention. The three modalities will then be combined by two composition components as shown in Figure 10.

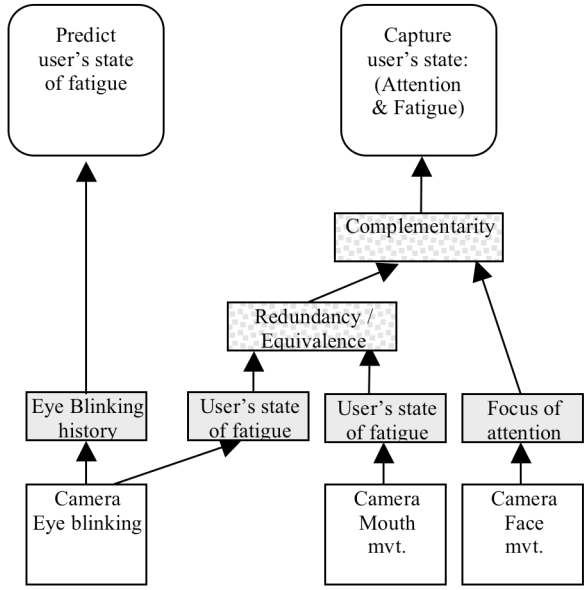

Fig. 10. Combined modalities for capturing and detecting user's state.

For selecting the output modalities the user issues speech commands such as "windshield screen voice beep tactile" for selecting all the output modalities. For using this combined modality, the user first selects a wheel button then issues the voice command, and finally selects again the button to indicate the end of the speech commands. As shown in Figure 9 two pure modalities, speech and button, are combined by a Complementarity composition component. Finally an Interaction Language component is responsible for combining all the recognized words between the two button press events. The output of this component is a list of selected modalities that is sent to the second Dialogue Controller of the meta User Interface.

- For outputs, five pure modalities made of a device and an interaction language component are defined for presenting an alarm. Such modalities are combined by a Redundancy/Equivalence composition component. This composition component implies that the five modalities can be used all together in a redundant way or that only a sub-set of the modalities (1 to 5 modalities) can be used in a redundant way.

Having presented the ICARE overall software architecture of the multimodal driving simulator, we now present the implemented architecture and in particular which components of the architecture have been implemented in OpenInterface.

A. Implemented architecture

We first describe the implemented OpenInterface components and then explain in the following section the differences between the ICARE conceptual architecture and the implemented architecture. We have developed six OpenInterface components:

- One OpenInterface component is dedicated to the video stream. Such a component is not explicit in the ICARE architecture since it represents a supplementary layer of the physical device driver.
- One OpenInterface component is implementing the software interface to be able to send messages to the TORCS code.
- One OpenInterface component implements all the ICARE diagrams for the task "Show alert message" of Figure 9 as well as the meta User Interface. This component has been implemented with ICARE JavaBeans components. The final implemented ICARE diagram is presented in Figure 11. First, due to time constraints, the Complementarity component of Figure 9 has not been used for developing the combined active modalities based on speech and a steering wheel button. Second, we decided to add a new modality for choosing modalities using dedicated buttons on the steering wheel. The two modalities are then equivalent for the task "Choose output modalities".

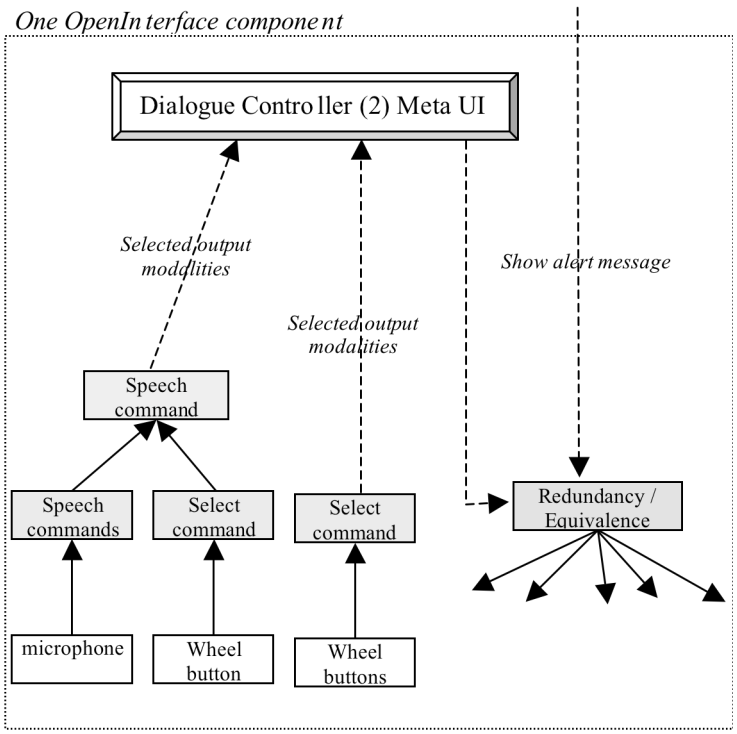

Fig. 11 Implemented ICARE components for the output modalities and meta User Interface. All the ICARE components are encapsulated in one OpenInterface component.

- One OpenInterface component corresponds to the "Eye Blinking history" for predicting the user's state of Fatigue.
- Two OpenInterface components correspond to the ICARE diagram of Figure 10 for capturing the user's state (Attention & Fatigue). Figure 12 presents the implemented processes of these two implemented

OpenInterface components.

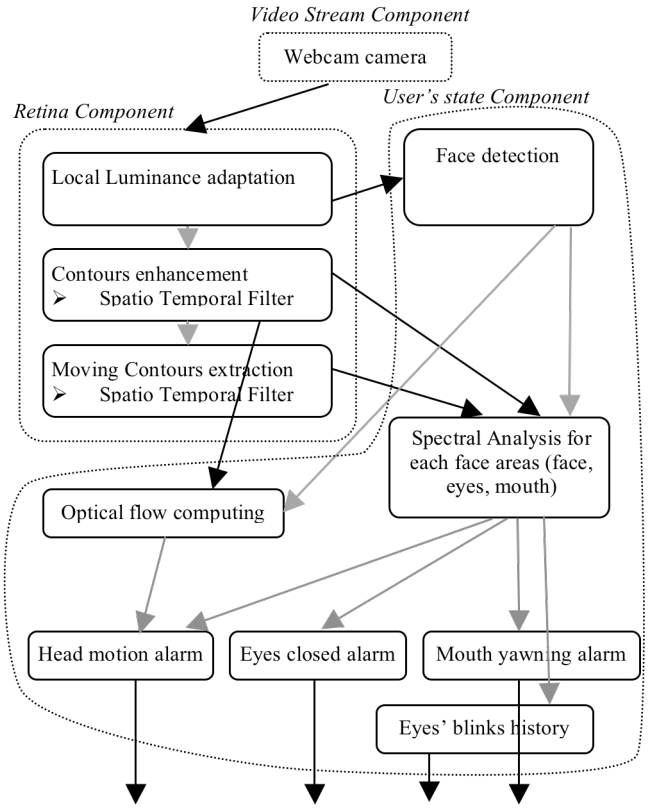

Fig. 12 Implemented OpenInterface components for capturing the user's state. Starting from the input provided by the Video Stream component, two OpenInterface components, namely Retina component and User's State component, have been implemented for providing four outputs: Focus of attention (head motion), duration of eyes closed, yawning and eye blinking history.

The video analysis system for capturing user's state is composed of two OpenInterface components: a prefiltering component that enhances the input data and extracts different information. The second component computes the user face analysis and outputs different indicators related to the user's state.

Retina Component description

Once a frame is acquired from the video stream component, it is processed by the Retina component. This component is a filter coming from the modeling of the human retina [12, 13]. It provides three outputs for each frame:

- a gray picture close to the input frame but with a corrected luminance. This output allows a better extraction of the details of the picture in the dark areas by enhancing locally the sensitivity to the luminance.
- a picture of all the contours in the input frame. This output contains only the contours of the input. It is robust against spatio-temporal noise and luminance variations. It allows a description of the details of the input such as eyes and mouth contours which are used by the fatigue detection.
- a picture of all the moving contours. This output only reports energy on the areas in which contours are moving. It allows event detection and motion description [14].

As an illustration, Figure 13 shows the three outputs of this component according to a frame input. The data provided by this Retina component are directed to the different modules of the User's State component.

Fig. 13 Illustration of the outputs of the Retina component.

Description of the User State component

The first module of the User State component provides the position of the head in the visual scene, and is made of the Machine Perception Toolbox [14]. This module accepts as input a gray level picture. Nevertheless, luminance variations on the face can make this module fail. Then, in order to make it more robust, its input is the corrected luminance output of the Retina component instead of the Video Stream Component.

Once the face is detected, two modules work in parallel. The first is the Optical Flow Computing module, which computes the velocity on the face area with the help of neuromorphic velocity filters [15]. This module provides the horizontal and vertical estimated velocities. The second module is the Spectrum Analysis module. It consists of the log polar spectrum analysis of both contour output and moving contour output of the Retina component. This module is based on the modeling of the primary visual cortex area V1. As explained in [13] [16], by analyzing the temporal response of the log polar spectrum of the moving contours response of the face, it is possible to retrieve motion event alarms and motion orientation when motion is occurring. Finally, this module provides alarms for different face motions: the global head motion, eyes and mouth motions (opening/closing). Also, by analyzing the temporal evolution of the Retina component contour output, it is possible to evaluate the state "Open" or "Close" of the eyes and the mouth yawning.

Outputs generation

The User State module provides different outputs that are used by the components presenting the alarms.

Three outputs are alarms related to the estimation of the driver fatigue level. An alarm is sent when the user closes his eyes for more that a specified duration (we experimentally fix it to 200ms). Another is sent when the driver yawns.

Also, an alarm is generated when the user moves his head longer than a specified period (we experimentally fix it to 300ms). The generation of this alarm is based on the data provided by the Optical Flow Computing module and the global head spectrum analysis. Once a head motion event is detected by the Spectrum Analysis module, the velocity data coming from the Optical Flow module and motion orientation

coming from the Spectrum Analysis module are fused to generate the appropriate alarm in the event that the information is redundant.

These alarms are developed to signal user fatigue dynamically. In order to provide a long term prediction of hypo-vigilance, we generate a last output which is a list of the duration of the eye blinks encountered in the last 20 seconds. This output is sent to the hypo-vigilance prediction component.

Sleep prediction component

The aim of this component is to provide the driver with a warning several minutes before he/she loses control of the vehicle due to extreme hypo-vigilance or sleep. The prediction is made based on the 20 second eyelid activity history of the subject. Specifically the input of the component is the start and end timestamps of the blinks as these are registered by the video analysis system.

The output of the component is a binary value 1 or 0 corresponding to warning or no warning.

The prediction of the component is calculated via the fuzzy fusion of several features that characterize the blinking behavior of the driver (Fuzzy Expert System). These features that were selected based on literature review [17], [18] and the expertise gained in previous related projects such as AWAKE [19] are the following:

- ***Long blinks duration***: the blinks in the 20 second window are filtered and only the ones lasting more than 0,3s are kept. If the number of long blinks is larger than 2 the sum of their durations is the long blink duration feature. Else the LBD = 0.
- ***Maximum interval between blinks*** is defined as the interval between the end of the current blink and the beginning of the next (t1[blink+1] - t3[blink]).
- ***Blinking rate.***

Although these features are not the most efficient ones they were the only ones that could be extracted given the input data and the camera used for video acquisition (30fps). Features that take into account velocity characteristics of the blinks are reported to have greater accuracy [20], however for the extraction of these features a high speed camera capable of 200fps and special software is needed.

In the following figure a schematic representation of the fuzzy system's premise space is shown. The features form a three dimensional space and their partitioning using three fuzzy sets per input leads to the formation of 27 fuzzy rules. Each fuzzy rule has a different output thus giving us the ability to model 27 different blinking behaviors prior to the sleep onset. The final output/prediction of the system is calculated by combining the outputs of the fuzzy rules that are triggered by the eyelid activity pattern (LBD | Max interval | blinking rate) on real time.

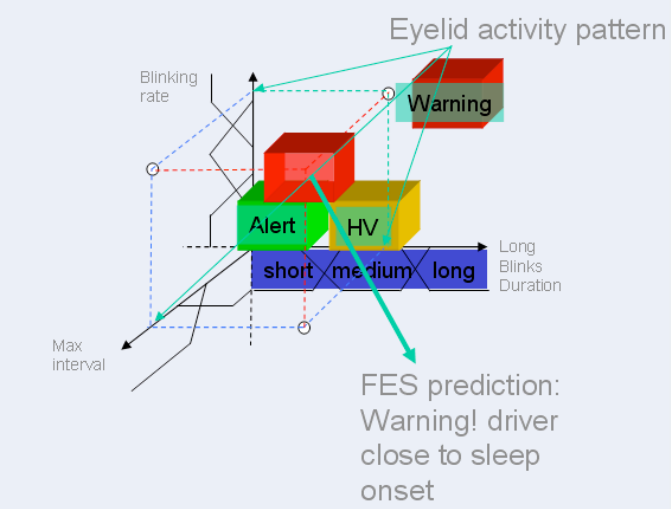

Fig. 14 A schematic of the FES premise space. Depending on which fuzzy rules are triggered by the eyelid activity pattern the output of the system is calculated in real time.

For the training of the fuzzy system's parameters data from 30 drowsy drivers were used, namely the blinking history of the subjects and the timestamps of the accidents during the driving sessions.

The method that was used for training was a real-coded genetic algorithm and the fitness function was chosen so as to maximize the correct predictions ratio and minimize the number of alarms so as to be as unobtrusive to the driver as possible [21]. Even though the training of the FES parameters with a GA takes a substantial amount of time that can reach one hour, once the parameters are trained the system generates its output instantly for online operation. Tests that were carried out during the workshop using this data led to a prediction accuracy of 80% for the training set of 30 drivers.

The component was developed in C++ and was delivered in the form of a dll for integration.

B. Discussion: tradeoffs and compatibility between ICARE and OpenInterface

There is no direct 1-1 mapping between the ICARE component architecture and the implemented OpenInterface component architecture. Nevertheless we demonstrated the compatibility and feasibility of the approach.

For the user's state capture, the two implemented OpenInterface components define large components. The componentization as described in Figure 10 would have been a difficult task since the code is developed in Matlab. Matlab had been initially used for exploring solutions. For providing final components after a feasibility phase made in Matlab, it would be useful to fully redevelop the final version in C++. Moreover we did not define one component for each feature used in the image, as advocated by Figure 10, for efficiency reasons because this would involve duplication of the video stream input.

For the output multimodal interface, we show the benefit of the ICARE approach, that is, that it allows us to quickly add a new equivalent modality for selecting the output modalities

within the meta User Interface and that it allows us to reuse components such as the Device component Loudspeakers for lexical feedback from the speech recognizer.

More OpenInterface components could have been defined corresponding to the ICARE software architecture: this was not pursued simply due to time constraints.

V. CONCLUSION

By considering the case study of a driving simulator, we focused on designing a software component architecture for multimodal interfaces from the Human-Computer Interaction domain, and how to implement it using the OpenInterface as well as the ICARE platforms. The compatibility of the two platforms is evident since several ICARE components are encapsulated within one OpenInterface component.

In future work, we first need to integrate the user's state prediction component within the demonstrator. We also plan to define new OpenInterface components particularly for the developed output multimodal interfaces. Moreover new native OpenInterface connectors could be defined corresponding to the ICARE output composition components. This work has already been done for the input ICARE composition components although we did not use them in this case study.

Moreover we would like to use new passive modalities for capturing the stress level of the user based on biological signals analysis. We are currently defining the corresponding OpenInterface components. We plan to integrate the stress level within our demonstrator as part of the meta User Interface for automatically selecting the output modalities in addition to allowing the user to select them.

Finally we would be interested to perform some usability experiments and to study the benefit of our component architecture in quickly modifying multimodal interaction and retesting the interaction as part of an iterative user centered design method.

REFERENCES

[1] *SIMILAR, European Network of Excellence*, WP2, OpenInterface platform. www.similar.cc

[2] J. Bouchet and L. Nigay, "ICARE: A Component-Based Approach for the Design and Development of Multimodal Interfaces", in *Proc. CHI'04 conference extended abstract*, ACM Press, 2004, pp. 1325-1328.

[3] J. Bouchet, L. Nigay and T. Ganille, "ICARE Software Components for Rapidly Developing Multimodal Interfaces", in *Proc. ICMI'04 conference*, ACM Press, 2004, pp. 251-258.

[4] A. Benoit et al., "Multimodal Focus Attention Detection in an Augmented Driver Simulator", in *Proc. eNTERFACE'05 workshop*, 2005, pp. 34-43. www.enterface.net/enterface05/

[5] J-F. Kamp, "Man-machine interface for in-car systems. Study of the modalities and interaction devices", Ph.D. dissertation, ENST, Paris, 1998.

[6] M. Tonnis, C. Sandor, G. Klinker, C. Lange, H. Bubb, "Experimental Evaluation of an Augmented Reality Visualization Car Driver's Attention", in Proc. ISMAR'05, IEEE Computer Society, 2005, pp. 56-59.

[7] TORCS Driver Simulator: torcs.sourceforge.net

[8] JavaBeans 1.01 specification, Sun Microsystems 1997. java.sun.com/products/javabeans/docs/

[9] L. Nigay, J. Coutaz, "A Generic Platform for Addressing the Multimodal Challenge", *in Proc. CHI'95 conference*, ACM Press, 1995, pp. 98-105.

[10] L. Nigay, J. Coutaz, "The CARE Properties and Their Impact on Software Design", *in Intelligence and Multimodality in Multimedia Interfaces*, 1997.

[11] B. Mansoux, L. Nigay and J. Troccaz, "Output Multimodal Interaction: The Case of Augmented Surgery", *in Proc. HCI'06 conference*, Springer-Verlag and ACM Press, 2006, to appear.

[12] W. Beaudot, "The neural information processing in the vertebrate retina: A melting pot of ideas for artifficial vision", PhD Thesis in Computer Science, INPG (France) december 1994.

[13] A. Benoit , A. Caplier "Head nods analysis : interpretation of non verbal communication gestures " IEEE, ICIP , Genova, Italy, 2005

[14] Machine Perception Toolbox (MPT) http://mplab.ucsd.edu/grants/project1/free-software/MPTWebSite/API/].

[15] A. Torralba, J. Herault "An efficient neuromorphic analog network for motion estimation." IEEE Transactions on Circuits and Systems-I: Special Issue on Bio-Inspired Processors and CNNs for Vision. Vol 46, No. 2, February 1999.

[16] A. Benoit , A. Caplier "Hypovigilence Analysis: Open or Closed Eye or Mouth ? Blinking or Yawning Frequency ?" IEEE, AVSS, Como, Italy, 2005.

[17] Yannis Damousis, Dimitrios Tzovaras: Correlation between SP1 data and parameters and WP 4.4.2 algorithms, SENSATION Internal Report, November 2004.

[18] Alex H. Bullinger et al "Criteria and algorithms for physiological states and their transitions, SENSATION_Del_1_1_1.doc", SENSATION Deliverable 1.1.1, August 2004

[19] A. Giralt et al. "Driver hypovigilance criteria, filter and HDM module", AWAKE Deliverable 3.1, September 2003.

[20] Johns, MW The amplitude-Velocity Ratio of Blinks: A New Method for Monitoring Drowsiness.

[21] I. G. Damousis et al "A Fuzzy Expert System for the Early Warning of Accidents Due to Driver Hypo-Vigilance", presented at the Artificial Intelligence Applications and Innovations (AIAI) 2006 Conference, 7-9 June, 2006, Athens, Greece.

A. Benoit was born in 1980 in France. He graduated from Institut National Polytechnique de Grenoble (INPG). Currently he is a Ph.D. candidate in the Laboratoire des Images et des Signaux (LIS) of Grenoble. His research interests are in the areas of human motion and head motion analysis. His work is based on the human visual perception system. He is also teaching signal processing at the Master level.

L. Bonnaud was born in 1970 in France. He graduated from the École Centrale de Paris (ECP) in 1993. He obtained his PhD from IRISA and the Université de Rennes-1 in 1998. Since 1999 he is teaching at the Université Pierre-Mendès-France (UPMF) in Grenoble and is a permanent researcher at the Laboratoire des Images et des Signaux (LIS) in Grenoble. His research interests include segmentation and tracking, human motion and gestures analysis and interpretation.

A. Caplier was born in 1968 in France. She graduated from the École Nationale Supérieure des Ingénieurs Électriciens de Grenoble (ENSIEG) of the Institut National Polytechnique de Grenoble (INPG), France, in 1991. She obtained her Master's degree in Signal, Image, Speech Processing and Telecommunications from the INPG in 1992 and her PhD from the INPG in 1995. Since 1997, she is teaching at the École Nationale Supérieure d'Électronique et de Radioélectricité de Grenoble (ENSERG) of the INPG and is a permanent researcher at the Laboratoire des Images et des Signaux (LIS) in Grenoble. Her main interest concerns the analysis and the interpretation of human motion. She works in the domain of facial expressions classification, human postures recognition, Cued Speech language classification and head rigid or non rigid motion analysis.

Y. Damousis was born in 1974 in Thessaloniki-Greece. He received the Dipl. Eng. Degree and a Ph.D from the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki in 1997 and 2003 respectively. Currently he is a senior researcher at the Informatics & Telematics Institute of the Centre for Research and Technology Hellas in Thessaloniki. His research interests are in the areas of expert systems, optimization and fusion in AI applications. He is a member of the Technical Chamber of Greece.

D. Tzovaras received the Diploma in electrical engineering and the PhD degree in 2D and 3D image compression from Aristotle University of

Thessaloniki, Thessaloniki, Greece, in 1992 and 1997, respectively. He is a senior researcher in the Informatics and Telematics Institute of Thessaloniki. Prior to his current position, he was a senior researcher on 3D imaging at the Aristotle University of Thessaloniki. His main research interests include virtual reality, assistive technologies, 3D data processing, medical image communication, 3D motion estimation, and stereo and multiview image sequence coding. His involvement with those research areas has led to the coauthoring of more than 35 papers in refereed journals and more than 80 papers in international conferences. He has served as a regular reviewer for a number of international journals and conferences. Since 1992, he has been involved in more than 40 projects in Greece funded by the EC and the Greek Secretariat of Research and Technology. He is an associate editor of the EURASIP Journal of Applied Signal Processing and a member of the Technical Chamber of Greece..

F. Jourde was born in 1981 in France. He graduated in computer science from the University of Grenoble 1. He is currently working as a research associate at the CLIPS laboratory of Grenoble. His research interests focus on Computer-Human Interaction (HCI) and in particular his research studies centre on formal specification of multimodal user interfaces and formal tests of multimodal interaction based on Lustre, a synchronous programming language.

L. Nigay was born in 1965 in France. She is a Professor at Université Joseph Fourier (UJF, Grenoble 1) and at Institut Universitaire de France (IUF). Her research interests focus on the design and development of user interfaces. In particular her research studies centre on Multimodal and Augmented Reality (AR) user interfaces such as the component-based approach named ICARE (Interaction Complementarity, Assignment, Redundancy and Equivalence) for the development of multimodal and AR interfaces and new interaction modalities combining the real and the physical worlds such as tangible user interfaces, embodied user interface and mobile augmented reality. She has published more than 130 articles in conferences, journals and books. L. Nigay has received several scientific awards (including the CNRS Bronze medal in 2002 and the UJF gold medal in 2003 and again in 2005) for excellence in her research and is involved in many international scientific societies and events, as well as European research projects.

M. Serrano was born in 1981 in Spain. He graduated in computer science both from the Facultad de Informatica of the Universidad Politecnica de Madrid in Spain and from the École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (ENSIMAG) in France, in 2005. He is currently working as a research associate at the CLIPS laboratory of Grenoble. His research interests focus on Computer-Human Interaction (HCI) and in particular his research studies centre on output multimodal interaction for augmented surgery and multimodal interaction on mobile devices such as phone and PDA.

L. Lawson was born in 1982 in Bénin. He graduated from the Engineering School of Université Catholique de Louvain (UCL) and obtained his Master degree in Computer Science and Engineering in 2004. He is currently working as a research associate at the Communication and Remote Sensing Laboratory (TELE) of Université Catholique de Louvain on the development of OpenInterface, an open source component-based platform.