Multimodal signal processing and interaction for a driving simulator : component-based architecture

Alexandre Benoit, Laurent Bonnaud, Alice Caplier, I. Damousis, F. Jourde, J.-Y. L. Lawson, L. Nigay, M. Serrano, D. Tzovaras

Journal on Multimodal User Interfaces, Vol. 1, No. 1, March 2007

MULTIMODAL SIGNAL PROCESSING AND INTERACTION FOR A DRIVING SIMULATOR: COMPONENT-BASED ARCHITECTURE

Alexandre Benoit, Laurent Bonnaud, Alice Caplier

GIPSA-lab, Grenoble, France
{benoit;bonnaud;caplier}
@lis.inpg.fr

Frédéric Jourde, Laurence Nigay, Marcos Serrano

LIG, Université Joseph Fourier, Grenoble, France
{frederic.jourde;laurence.nigay
;marcos.serrano}@imag.fr

Ioannis Damousis, Dimitrios Tzovaras

IT Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece
{damousis;tzovaras}@iti.gr

Jean-Yves Lionel Lawson

TELE Lab, Université catholique de Louvain, Belgium
lawson@tele.ucl.ac.be

ABSTRACT

In this paper we focus on the software design of 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 is 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 its software architecture based on multimodal signal processing and multimodal interaction components considering two software platforms, OpenInterface and ICARE.

KEYWORDS

Attention level – Component – Driving simulator – Facial movement analysis – ICARE – Interaction modality – OpenInterface – Software architecture – Multimodal interaction

1. INTRODUCTION

In the context of a multimodal driving simulator, we study component-based architecture using two platforms, namely OpenInterface [1] [2] and ICARE [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 [3]. One implementation of the ICARE model is defined using JavaBeans components [4].

We study the software design and development of a multimodal interactive system using the OpenInterface platform while the component architecture is along the ICARE conceptual model. The selected case study is a driving simulator. The project is multi-disciplinary and we aim at studying the integration of Signal Processing (SP) and Human-Computer Interaction (HCI) research results for developing multimodal systems. More precisely, the contribution of the project is threefold:

- We explain the integration of existing code (SP and HCI) as new components within the OpenInterface platform.
- We present the development of a new multimodal system with the OpenInterface platform by assembling components and by connecting them to an existing functional core (a driving simulator).
- We explicate the compatibility of the OpenInterface platform with the ICARE platform since some existing code is developed using the ICARE platform.

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 focus on the hypo-vigilance analysis and prediction. Before presenting the software architecture along the ICARE conceptual model, we recall the key points of the two platforms, OpenInterface and ICARE. We finally detail the software architecture that has been implemented followed by a discussion on the tradeoffs and differences with the initial ICARE architecture.

2. CASE STUDY: DRIVING SIMULATOR

2.1. 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 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 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.

Figure 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 ongoing 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.

Figure 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. The driver's face is monitored with a video camera in order to detect any sign of hypo-vigilance. This detection is based on the use of bio-inspired algorithms in order to analyze face movements, eyes blinking and yawning. A detailed analysis of eyes motion allows assessment of sleep prediction.

From the Human-Computer Interaction point of view, we focus 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). From the Signal Processing domain point of view, the study allows us to validate our bio-inspired algorithms for any facial motion analysis in terms of efficiency and computational rate. We also focus on the definition of a robust data fusion process in order to take a decision from head motion, eyes motion and mouth motion information.

2.2. Overview of the driving simulator

Starting from the programs developed during a first workshop at eNTERFACE 2005 [7], 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 that we used is the GPL program TORCS [8] and the multimodal interaction is developed using the two platforms OpenInterface and ICARE.

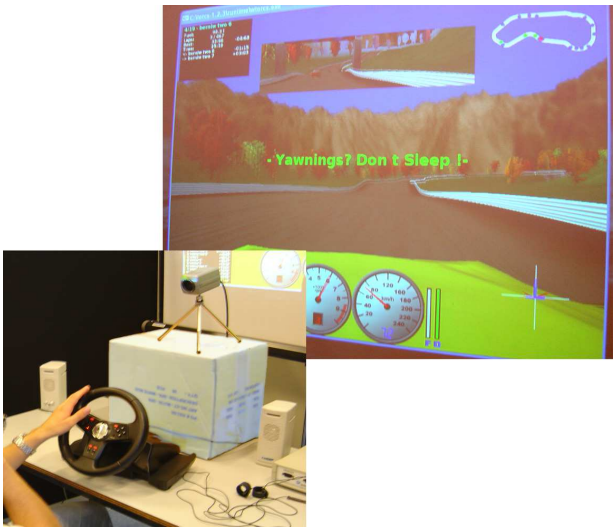

Figure 3: *Multimodal driving simulator: demonstrator in use.*

3. HYPO-VIGILANCE ANALYSIS AND PREDICTION

3.1. Hypo-vigilance detection with bio-inspired algorithms

At each time, the face of the driver is captured with a camera and three signs of hypo-vigilance are tracked: yawning, blinking (or eyes closure) and head motion. In order to do that, each frame of the driver's face video is processed:

- By a pre-filtering algorithm modeling the processes that are occurring at the human retina level;
- By a data fusion algorithm that extracts and fuses the information associated to each hypo-vigilance sign. A spectral analysis modeling the process that is occurring in the V1 visual area of the brain is used in order to extract the hypo-vigilance signs.

Figure 4 presents a global scheme related to the processes occurring at the retina level (first step) and at the V1 visual area level (second step).

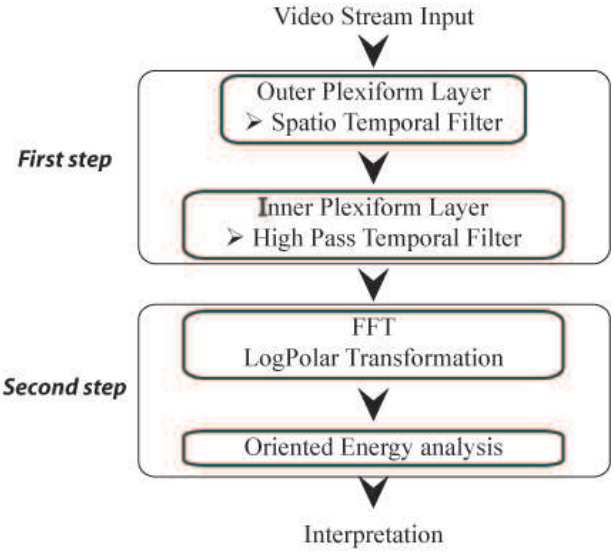

Figure 4: *Model of the retina.*

At the retina level, the visual information is double filtered. A spatio-temporal filter removes the noise and enhances the contours of the scene and a high pass temporal filter enhances the moving contours only. Figure 5 presents the transfer function of the spatio-temporal filter and figure 6 shows the impulse response of the high pass temporal filter. Figure 7 presents the results of the successive filtering steps on a sequence in which a head is moving horizontally. The output of the spatio-temporal filter shows the details of the image and in parallel, the output of the temporal high-pass filter extracts only the mobile contours. The expression of each transfer function is described in [9].

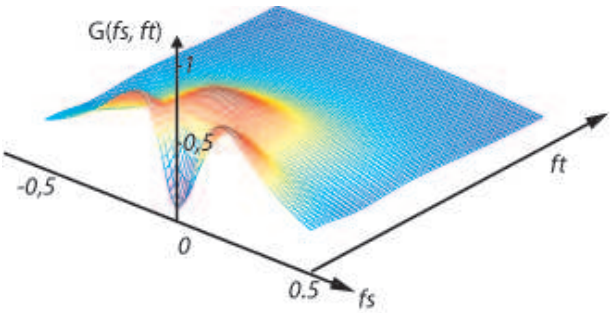

Figure 5: *Transfer function of the spatio-temporal filter.*

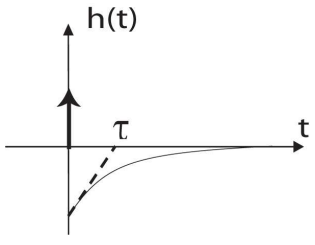

Figure 6: *Impulse response of the temporal high-pass filter.*

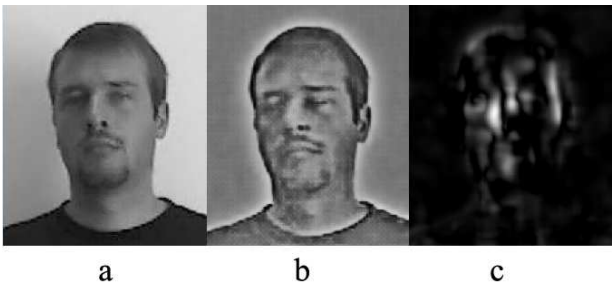

Figure 7: *Example of filtering on a horizontally moving head sequence (a), contours are extracted by the spatiotemporal filter (b) and moving contours are extracted after temporal high pass filtering (c).*

At the V1 area level, the scene is decomposed into some specific orientation and frequency bands in the Fourier domain. The amplitude FFT of each moving contour frame (figure 7-c) is computed in the log-polar domain. To compute the image spectrum in the log-polar domain, log polar Gabor filters are used:

$$G_{ik}(f,\theta) = \frac{1}{\sigma\sqrt{2\pi}}\left(\frac{f_k}{f}\right)^2 exp\left(-\frac{ln(\frac{f}{f_k})^2}{2\sigma^2}\right)\cdot cos\left(\frac{1+cos(\theta-\theta_i)}{2}\right)$$

In this formula, the GLOP (Gabor LOg-Polar) filter centered on frequency f_k in the θ_i orientation and with the scale parameter θ appears as a separable filter. Figure 8 gives an example of a log polar FFT. Each "pixel" of that frame represents the energy associated to a given band of orientation and energy. On this example, the spectrum analysis is carried out on the eye area in which the pupil is moving horizontally. The main energy is located around orientation 180° which represents the moving vertical contours of the analyzed area.

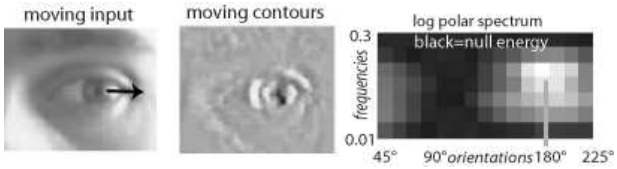

Figure 8: *Log polar spectrum of the moving contours of the eye.*

Figure 9 shows two other examples of results: the first one is related to an eye blink which is made of a vertical eyelid motion. This vertical motion creates an important energy increase on the horizontal orientations of the spectrum (90°). The second example shows the case of no motion which is related to a minimum energy on the spectrum because no moving contours are extracted.

As a consequence, the analysis of the log polar spectrum allows detecting the presence of motion (when no motion occurs, the global spectrum energy is null) and the direction of motion (the contours perpendicular to the motion direction are the contours with the highest energy in the log-polar spectrum) [10].

All the hypo-vigilance signs are derived from the analysis of the log-polar spectrum. Indeed, three such spectra are computed:

- one on a bounding box around the face,
- one bounding box around the eyes,
- one bounding box around the mouth.

As detailed in [11], in order to detect the eye blinks, the temporal evolution of the total spectrum energy $E(t)$ of each eye bounding box is analyzed. This energy evolution (as shown

in figure 10) presents energy drops when an eye motion occurs. Then, by selecting the energy drops related to vertical motions, eye blinks are detected, vertical motions being detected with the help of the log polar spectrum (figure 9-a).

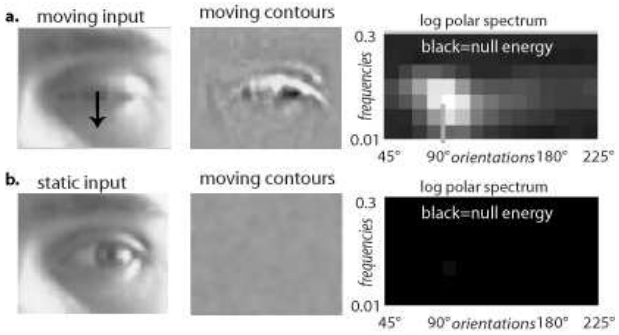

Figure 9: *Log polar spectrum of an eye that is closing (a), log polar spectrum of a static eye (b).*

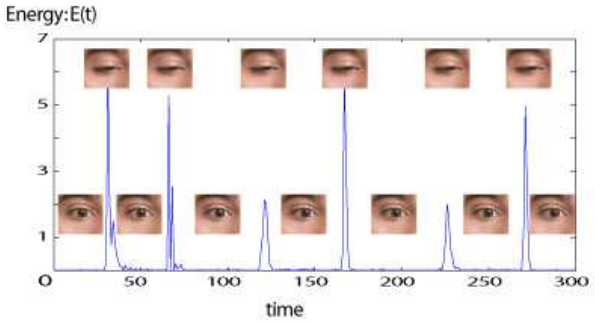

Figure 10: *Temporal evolution of the energy of the spectrum in an eye bounding box. Each energy increase is linked to an eye movement, by selecting only vertical motions, each eye blink is detected.*

In order to make the analysis of the face motion more efficient, the algorithm based on neuromorphic velocity filters described in [12] is used in order to compute the optic flow associated with the pixels of the face bounding box. As a result, the horizontal and vertical velocities of the face are estimated.

3.2. Sleep prediction

The aim 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 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 is a binary value 1 or 0 corresponding to warning or no warning.

The prediction 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 [13], [14] and the expertise gained in previous related projects such as AWAKE [15] 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.
- 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 [16], however for the extraction of these features a high speed camera capable of 200fps and special software is needed.

In figure 11, 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 (Long blink duration / Max interval / Blinking rate) on real time.

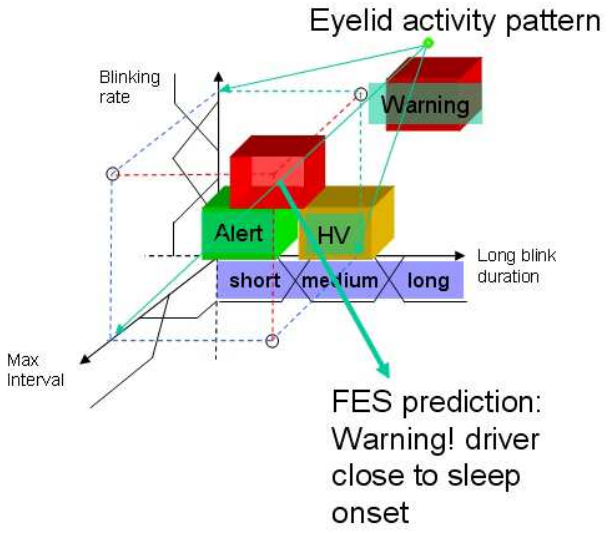

Figure 11: *A schematic of the FES premise space. Depending on which fuzzy rules are triggered by the eyelid activity pattern the output of the system is calculated in real time.*

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 [17]. 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 using this data led to a prediction accuracy of 80% for the training set of 30 drivers.

Having explained the algorithms for the hypo-vigilance analysis and prediction, we now focus on the software design. Two component platforms described in the following section have been used for the development of our case study. We then explain the corresponding designed component architecture.

4. COMPONENT PLATFORMS

4.1. 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 [1] and OpenInterface [2] projects, 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 12 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.

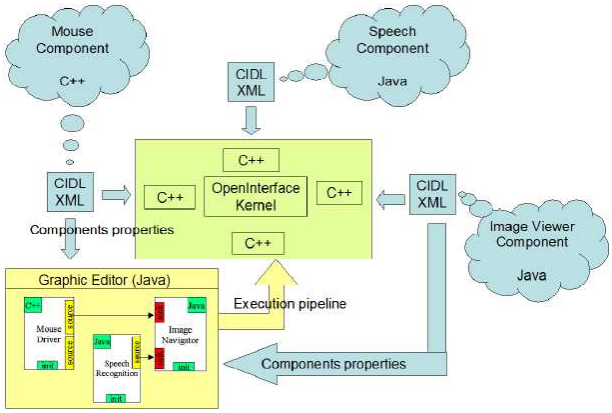

Figure 12: *Overview of the OpenInterface platform.*

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.

4.2. 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 [3, 4]. 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 [18]. 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 [19] as the coupling of a physical device d with an interaction language L: <d, L>. In [20], 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 [20]: 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. 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. For input multimodality, 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.

5. SOFTWARE ARCHITECTURE OF THE MULTIMODAL DRIVING SIMULATOR

In this section, we first present the overall architecture along the ICARE conceptual model followed by the implemented architecture. We finally conclude by a discussion of the tradeoffs and differences between the initial conceptual architecture and the implemented one.

5.1. ICARE conceptual architecture

In figure 13, 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. The overall software architecture is organized along four modules. The Functional Core (FC) implements domain specific concepts in a presentation independent way. The Functional Core Adapter (FCA) serves as a mediator between the Dialogue Controller and the domain-specific concepts implemented in the Functional Core. It is designed to absorb the effects of changes in its direct neighbours. Data exchanged with the Dialogue Controller are conceptual objects, that is perspectives on domain objects. Such perspectives are supposed to match the user's mental representation of domain concepts. They transform computational objects into abstractions driven by considerations for the user's conceptual model. At the other end of the spectrum, the interaction modality module is managing the input and output

concrete user interface. This module is developed using ICARE components. Finally the Dialogue Controller (DC) is the keystone of the model. It has the responsibility for task-level sequencing and is modality independent.

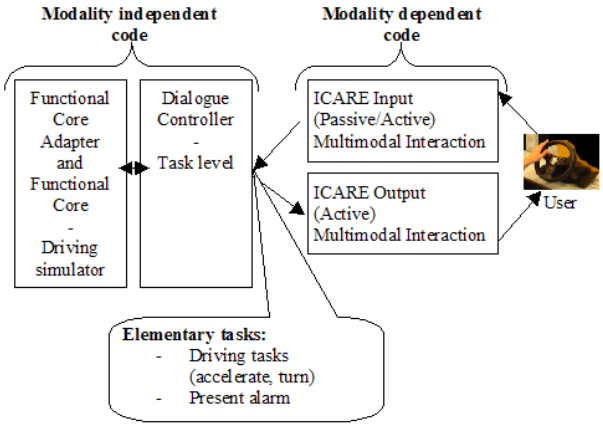

Figure 13: *Overall architecture of the multimodal driving simulator.*

As pointed out in figure 13, 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.

Because we reuse the GPL driving simulator TORCS [8] that we extend to be multimodal, some parts of the architecture of Figure 13 are already developed. Figure 14 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.

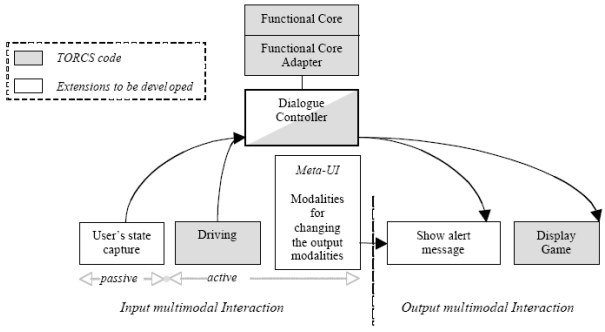

Figure 14: *TORCS code and extensions to be developed within our architecture.*

To better understand the extensions to be developed, figure 15 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 16. 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 16) 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 [21].

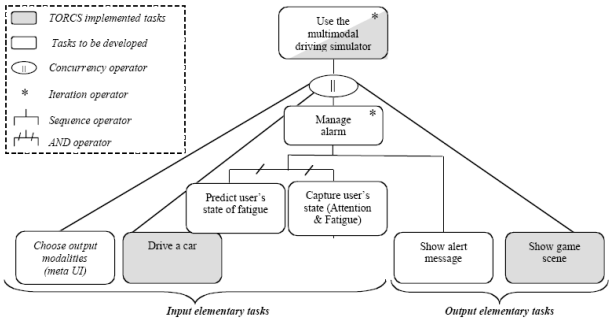

Figure 15: *Hierarchical Task Analysis (HTA): Task tree corresponding to the Dialogue Controller.*

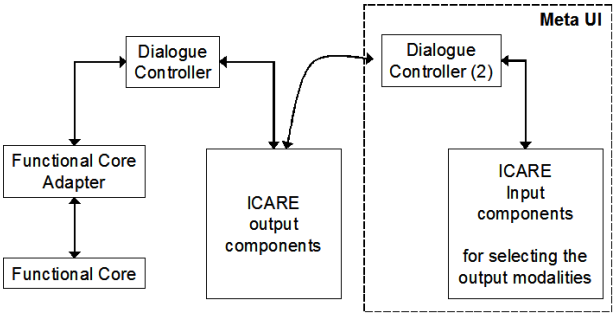

Figure 16: *Meta User Interface: ICARE components within an overall software architecture of an interactive system and the meta UI that enables the selection of equivalent modalities by the user (from [20]).*

To obtain the final ICARE architecture, for each elementary task of figure 15, an ICARE diagram is defined. Figure 17 presents the four ICARE diagrams designed for the four elementary tasks to be developed.

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

- For inputs, 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 18.

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 17 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.

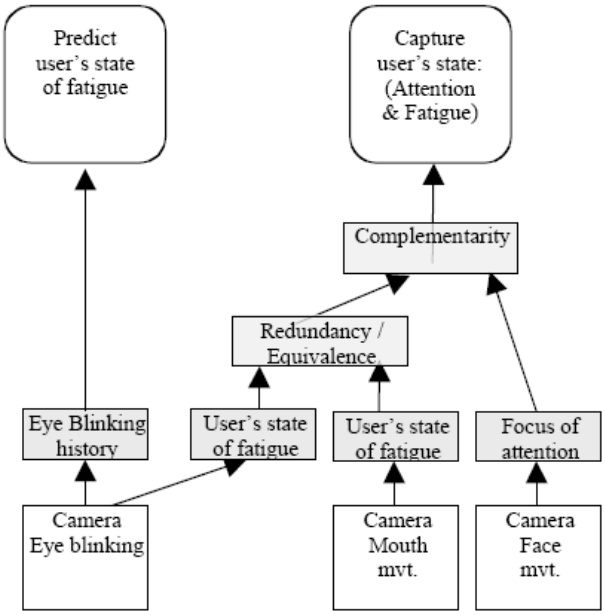

Figure 18: *Combined modalities for capturing and detecting user's state.*

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.

5.2. 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 17 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 19. First, due to time constraints, the Complementarity component of figure 17 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".
- 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 18 for capturing the user's state (Attention & Fatigue). Figure 20 presents the implemented processes of these two implemented OpenInterface components.

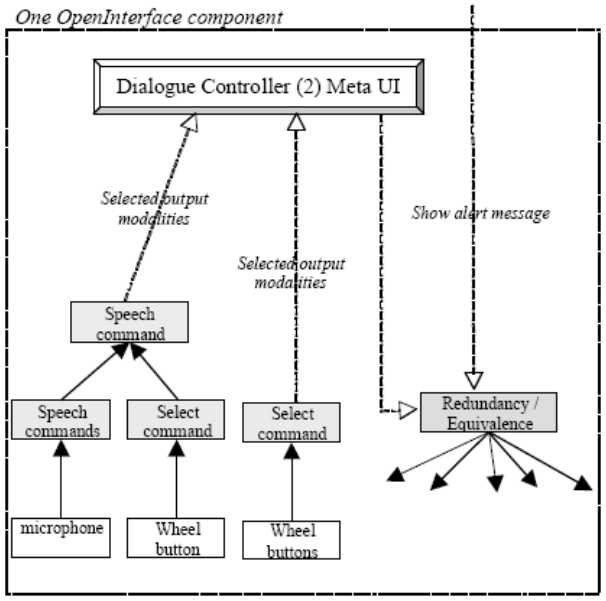

Figure 19: *Implemented ICARE components for the output modalities and meta User Interface. All the ICARE components are encapsulated in one OpenInterface component.*

The video analysis system for capturing user's state is composed of two OpenInterface components: the Retina 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.

5.2.1. Description of the Retina component

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 (see section 3).

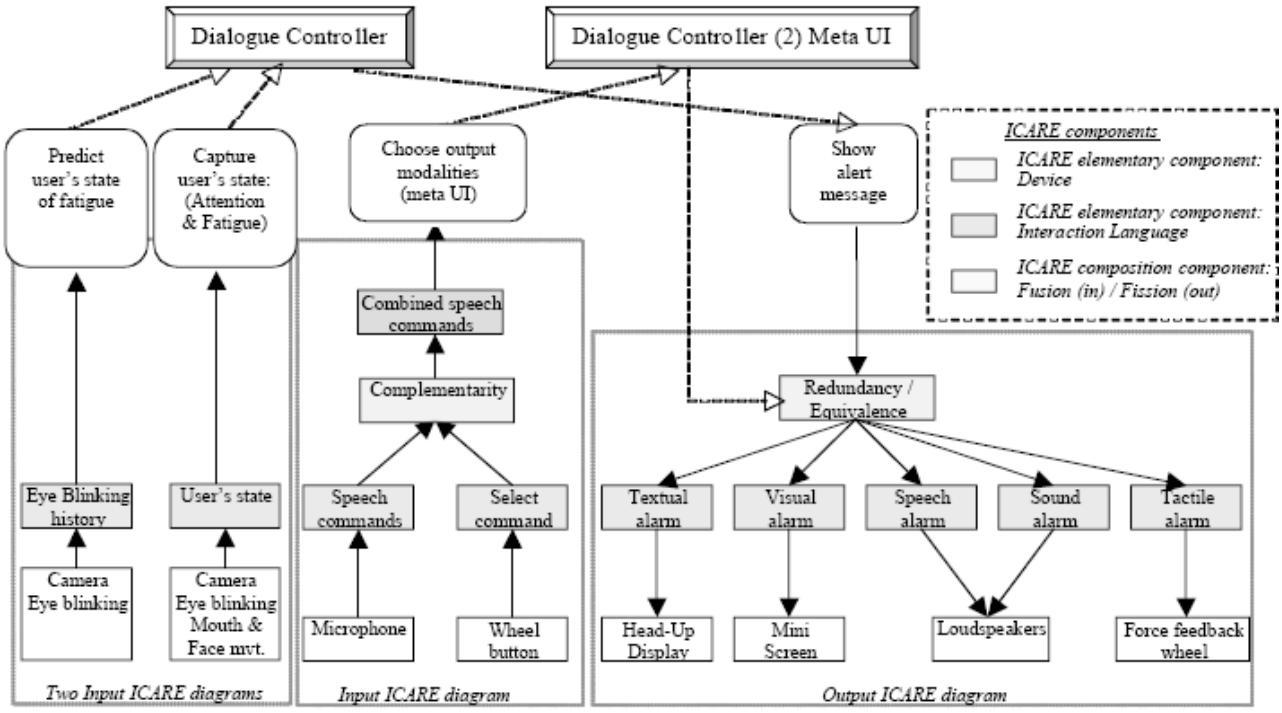

Figure 17: *ICARE diagrams for the elementary task of figure 7.*

5.2.2. 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 [22]. 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 and the Spectrum Analysis module (see section 3). This module provides alarms for different face motions: the global head motion, eyes and mouth motions (opening/closing).

The User State component 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 than 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.

5.2.3. Description of the Sleep Prediction component

The sleep prediction algorithm is integrated in the sleep prediction component (see section 3). This component was developed in C++ and was delivered in the form of a dll for integration.

5.3. Discussion: tradeoffs and compatibility between ICARE & 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 18 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 18, 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. For outputs, only one OpenInterface component implements all the ICARE diagrams. More OpenInterface components could have been defined corresponding to the ICARE software architecture. To do so, we need to extend the OpenInterface platform by defining new connectors corresponding to the ICARE output composition components.

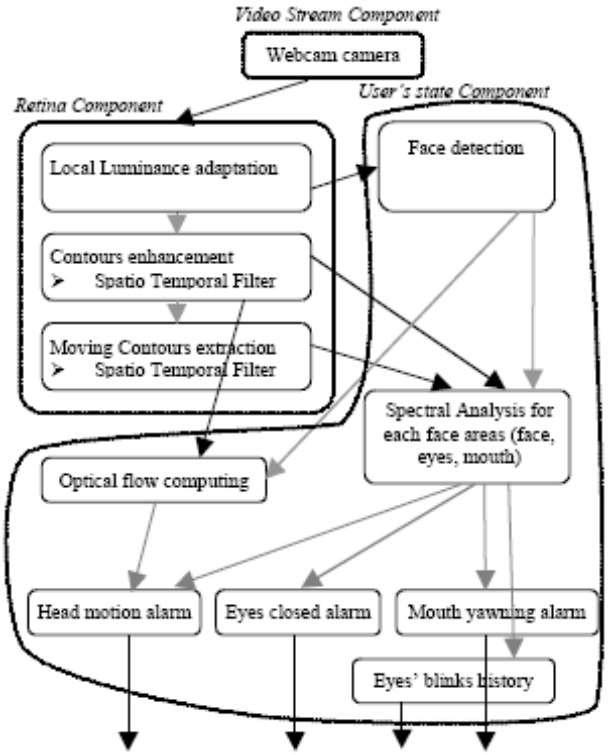

Figure 20: *Implemented OpenInterface components for capturing the user's state. Starting from the input provided by the Video Stream component, two OpenInterface components, namely Retina component and User's State component, have been implemented for providing four outputs: Focus of attention (head motion), duration of eyes closed, yawning and eye blinking history.*

6. CONCLUSION

By considering the case study of a driving simulator, we focused on designing and developing 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. We explained how to define components from existing code from both Signal Processing and Human-Computer Interaction research results. We also showed the compatibility of the two platforms 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. To do so, new native OpenInterface connectors need to 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 in performing some usability experiments and in studying the benefit of our component architecture in quickly modifying multimodal interaction and retesting the interaction as part of an iterative user centered design method.

7. ACKNOWLEDGMENTS

Many thanks to G. Serghiou for reviewing the paper. This work is partly funded by SIMILAR, the European NoE on Multimodal Interfaces (FP6-507609), during the eNTERFACE'06 Workshop in Dubrovnik-Croatia and by the OpenInterface European FP6 STREP focusing on an open source platform for multimodality (FP6-035182).

8. REFERENCES

[1] SIMILAR, European Network of Excellence, WP2-OpenInterface platform. http://www.similar.cc. 49, 53

[2] OpenInterface European STREP project. http://www.oi-project.org. 49, 53

[3] 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*, pp. 1325–1328, ACM Press, 2004. 49, 53

[4] J. Bouchet, L. Nigay, and T. Ganille, "ICARE Software Components for Rapidly Developing Multimodal Interfaces", in *Proc. ICMI'04 conference*, pp. 251–258, ACM Press, 2004. 49, 53

[5] J.-F. Kamp, *Man-machine interface for in-car systems. Study of the modalities and interaction devices*. PhD thesis, ENST, Paris, 1998. 50

[6] M. Tonnis, C. Sandor, G. Klinker, C. Lange, and H. Bubb, "Experimental Evaluation of an Augmented Reality Visualization Car Driver's Attention", in *Proc. ISMAR'05*, pp. 56–59, IEEE Computer Society, 2005. 50

[7] A. Benoit, L. Bonnaud, A. Caplier, P. Ngo, L. Lawson, D. Trevisan, V. Levacic, C. Mancas-Thillou, and G. Chanel, "Multimodal Focus Attention Detection in an Augmented Driver Simulator", in *Proc. eNTERFACE'05 workshop*, pp. 34–43, 2005. 50

[8] TORCS Driver Simulator. http://torcs.sourceforge.net. 50, 54

[9] W. Beaudot, *The neural information processing in the vertebrate retina: A melting pot of ideas for artifficial vision*. Computer science, INPG, Grenoble, December 1994. 51

[10] A. Benoit and A. Caplier, "Head nods analysis: interpretation of non verbal communication gestures", in *IEEE ICIP*, (Genova, Italy), 2005. 51

[11] A. Benoit and A. Caplier, "Hypovigilence Analysis: Open or Closed Eye or Mouth? Blinking or Yawning Frequency?", in *IEEE AVSS*, (Como, Italy), 2005. 51

[12] A. Torralba and 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, February 1999. 52

[13] I. Damousis and D. Tzovaras, "Correlation between SP1 data and parameters and WP 4.4.2 algorithms", tech. rep., SENSATION Internal Report, November 2004. 52

[14] A. H. Bullinger, "Criteria and algorithms for physiological states and their transitions", tech. rep., SENSATION Deliverable 1.1.1, August 2004. 52

[15] D. Esteve, M. Gonzalez-Mendoza, B. Jammes, and A. Titli, "Driver hypovigilance criteria, filter and HDM module", tech. rep., AWAKE Deliverable 3.1, September 2003. 52

[16] M. Johns, "The amplitude-Velocity Ratio of Blinks: A New Method for Monitoring Drowsiness", tech. rep., Epworth Sleep Centre, Melbourne, Australia, 2003. 52

[17] I. G. Damousis, D. Tzovaras, and M. Strintzis, "A Fuzzy Expert System for the Early Warning of Accidents Due to Driver Hypo-Vigilance", in *Artificial Intelligence Applications and Innovations (AIAI) 2006 Conference*, (Athens, Greece), June 7-9 2006. 52

[18] Sun Microsystems, *JavaBeans 1.01 specification*, 1997. http://java.sun.com/products/javabeans/docs/. 53

[19] L. Nigay and J. Coutaz, "A Generic Platform for Addressing the Multimodal Challenge", in *Proc. CHI'95 conference*, pp. 98–105, ACM Press, 1995. 53

[20] L. Nigay and J. Coutaz, *Intelligence and Multimodality in Multimedia Interfaces: Research and Applications*, ch. Multifeature Systems: The CARE Properties and Their Impact on Software Design, p. 16. AAAI Press, 1997. 53, 54

[21] 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. 54

[22] Machine Perception Toolbox (MPT). http://mplab.ucsd.edu/grants/project1/free-software/MPTWebSite/API/. 56