Eur. Transp. Res. Rev. (2014) 6:455–467
DOI 10.1007/s12544-014-0143-3

ORIGINAL PAPER

Argos-ESL: a versatile tool for planning and managing experiments on driving behavior

Antonio Pérez · M.Isabel García · Manuel Nieto · José L. Pedraza · Santiago Rodríguez · Juan Zamorano

Received: 30 October 2012 / Accepted: 26 June 2014 / Published online: 31 July 2014

Abstract
Purpose Argos is a long-term program started in 1990 to promote experimental research about driver behavior under realistic driving conditions. The development of the latest Argos platform has recently been completed and some experiments have been carried out based on this new complex and powerful tool.
System Description Argos allows recording multiple car parameters (such as speed), driver variables (such as the point of gaze), and environmental parameters (some of which are obtained by means of real-time signal processing, such as the distance to lateral road marks). All of these data can be used to replay driving sessions in the laboratory and to extract data associated with different driving session segments.
Paper contents This paper is focused on the description of the "Roadmap Program Interpreter", a tool 2 embedded in the Argos acquisition and control system which not only interacts with the equipment, but also with the driver and with the experiment supervisor. This tool is based on a new language, ESL –Experiment Specification Language– designed for the Argos project to help the experiment designer to formally specify complex, controlled, and repeatable experiments in a very efficient way. Argos-ESL has allowed the Spanish Traffic Agency (DGT, a governmental agency) as well as other local agencies to develop complex experiments in a fraction of the time previously needed by the DGT for designing similar driving experiments.

Keywords In-vehicle data recorder (IVDR) · Advanced Driver-assistance systems · Driver behavior · Feedback

A. Pérez (✉) · M. I. García · M. Nieto · J. L. Pedraza · S. Rodríguez · J. Zamorano
Department of Computer System Architecture and Technology, Technical University of Madrid Facultad de Informática, Campus de Montegancedo, sn 28660 Madrid, Spain
e-mail: aperez@fi.upm.es.

M. I. García
e-mail: mgarcia@fi.upm.es.

M. Nieto
e-mail: mnieto@fi.upm.es.

J. L. Pedraza
e-mail: pedraza@fi.upm.es.

S. Rodríguez
e-mail: srodri@fi.upm.es.

J. Zamorano
e-mail: jzamora@fi.upm.es.

1 Introduction

Car driver behavior is a crucial issue in road safety. This field, which has seen great research efforts by both governmental agencies and vehicle manufacturers, involves a combination of human, environmental, and vehicle related factors.

Many driver behavior studies are based on driving simulators, because they represent a safe option [1, 5, 16]. However, driving in real conditions can provide human behavior experts with complementary data about actual driver behavior, albeit at the cost of requiring the use of costly instrumented vehicles. Most often instrumented vehicles limit their functionality to logging several kinds of data, thus acting as simple data-loggers [17]. On-board data-acquisition systems are usually referred to as In-Vehicle Data Recorders (IVDR).

Many specific experiments can be designed to characterize driver behavior using instrumented vehicles

or driving simulators. Some primary examples can be classified based on their respective main subject of study:

- Effects of mental workload on driving performance (including research about how phone conversations or other sources of driver distraction affect driving.)
- Driver perception and reaction time in response to unexpected or surprising events.
- Several aspects of car-following behavior, such as the relationship between following distance and speed, lane following analysis and longitudinal driving behavior.

Belonging to the first class, Liang et al. [5] propose a method to detect driver distraction in real time using eye movements and driving data captured while the driver interacts with an acoustic communication in-vehicle system. Comte [1] focuses on attempting to reduce drivers' tendency to exceed speed limits. Her work falls within the second of the above-mentioned classes because its main subject is focused on driver reaction time. Also in this class Rakha et al. [12] use an instrumented vehicle equipped with a differential global positioning system (DGPS) and a data acquisition system to characterize the impact of driver age and gender as well as driver behavior based on the perception-reaction time (PRT) at high speed through signaled intersections. Finally, other authors use instrumented vehicles to acquire data to build a simulation model of driver behavior in car-following – see Ma [6] – or to predict the intention to change lanes by analyzing the driver's head movements and eye gaze –Doshi [2]. These studies belong to the last class in the above classification.

In general, repeating experiments with the same driver under the same conditions allows an experiment supervisor to draw conclusions about driver behavior reproducibility. Repeating the same experiment with different drivers or under different external conditions provides information about usual/unusual driver behavior characteristics. However, repeating experiments is not easy when working with a real car in standard conditions.

Argos is an instrumented vehicle developed at the Computer Architecture Department (DATSI) of the Technical University of Madrid (UPM) based on an embedded multicomputer that records mechanical and environmental parameters as well as video images synchronized on a time code basis [7, 10]. The system also includes several software applications allowing a follow-up analysis of the collected data. Argos has been used in different research projects to study the relationship between driver attention and speed control [14] and the impact of using mobile phones on road safety [8].

The last version of the Argos instrumented car [10] continuously records vehicle data and driver information such as the point of gaze, environmental information such as the distance to other vehicles and to the lateral road markings. Several video cameras provide synchronized information about the driving scene. A cursor superimposed on the main road video scene tracks the gaze direction in real time. A major improvement of the new Argos system is its ability to interact with the driver by means of light and audio stimuli activated at specific or scheduled time intervals.

This paper is focused on the description of the Roadmap Program Interpreter, a tool embedded in the Argos acquisition and control system which simultaneously interacts with the equipment, the driver, and the experiment supervisor. This tool is based on a new language, ESL –Experiment Specification Language– designed for the Argos project to help the experiment designer to formally specify complex, controlled, and repeatable experiments in the easiest possible way. Being able to program sessions with ESL is one of the main differences between the Argos system and other IVDR or instrumented vehicles.

The rest of this paper is organized as follows. The next section describes the instrumented Argos vehicle. Section 3 outlines the requirements to be considered while defining and developing driving experiments by means of the experiment Roadmap Program. Section 4 presents the Experiment Specification Language. Section 5 briefly describes an experiment example for driving sessions, defined through ESL. Finally, some concluding remarks are presented in Section 6.

2 The Argos system

In 1987, the Spanish Traffic Agency (DGT, a Public Administration agency) started the Argos Program whose main aim is to promote experimental research about driver behavior under realistic driving conditions. The first step of the program consisted in designing and implementing a platform to perform driving experiments involving an instrumented car and a laboratory. The main purpose of the car was to collect data related to both the driver and the car, as well as to environmental conditions. The experiments were conducted by the session supervisor (usually a psychologist) who could monitor the acquired data on a laptop computer and store predefined codes or comments. All the collected data had to be post-processed in the laboratory in order to reproduce the driving session and to extract any relevant information. A more detailed description can be found in [7, 9].

While the design of the platform proved to be suitable for the initially planned experiments and research, using the system and planning the experiments soon created the need for new features which the Argos car did not offer as it was a pure data recorder. This was particularly apparent in the lack of ways for the driver to effectively interact with the system.

As new and more complex experiments were designed [8, 13–15], new peripheral equipment was added to the system, most of it to provide some kind of interaction, such as push buttons included in the steering wheel body or a light barrier detection system. However, in most cases the associated data could only be stored synchronously to be post-processed with the rest of the session data.

Managing the experiments meant that the supervisors had to write detailed road maps on sheets of paper that included the configuration procedures, the steps to be followed, the conditions or the moments at which certain actions needed to be performed, the precise instructions to be given to the driver, etc.

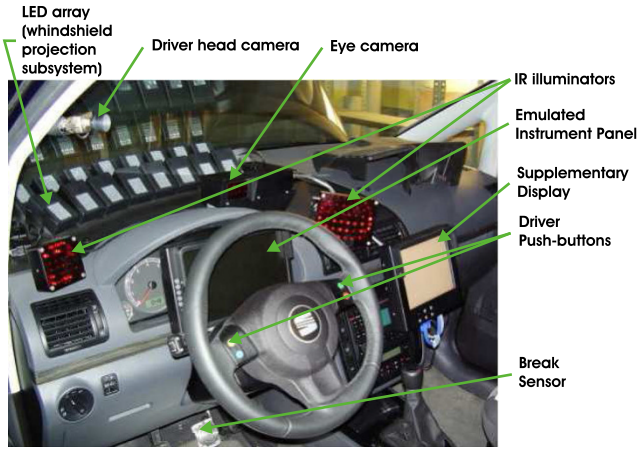

Fig. 1 Argos driving post

2.1 The new Argos car

The instrumented car was used for more than ten years and it helped DGT to make regulatory decisions on road safety. In 2003 DGT started a second project with the DATSI to develop a new platform. This new platform combines the wide experience acquired by the DGT experts and that of the platform designers based on their continuous feedback from the actual Argos car users. Thus, the system was completely redesigned including state of the art equipment and foreseeing a long useful life of the platform including continuous and easy updates with new features and peripheral equipment [10]. One of the added improvements is an experiment management tool, the Roadmap Program Interpreter, which interacts with the equipment, the driver and the experiment supervisor allowing the design of multistage and fully unattended experiments, i.e. experiments without any supervisor inside the car.

The new Argos system is an instrumented car with multiple sensors, cameras and devices. It has two main operational seats: the driver's seat and the supervisor's seat. The driver's seat includes several push-buttons in the steering wheel, a camera pointing at the driver, a camera pointing at the road and located as close as possible to the driver's head (scene camera), eye-tracking peripherals, a high-luminosity LED array to project text and graphics on the windshield, and an LCD display placed in front of the vehicle instrument panel that can be used to give instructions, to display video images, or to draw graphics replacing the instrument panel (see Fig. 1). The rear seat constitutes the main supervisor's post and consists of a wireless keyboard with an incorporated track-ball, three LCD displays, and two additional push-buttons.

Additionally, the front passenger's seat has a car pedal extension, an LCD display placed over the GPS navigator, and connectors for an auxiliary keyboard-mouse set and for tachistoscopic glasses (liquid crystal glasses whose lenses can be individually occluded). This post can be used in experiments involving tachistoscopic glasses, for security purposes in experiments on driving under fatigue, alcohol, drugs, etc, or as a secondary supervising post.

Argos has been designed as a multicomputer system built on top of four main PC subsystems interconnected by a Gigabit Ethernet (see Fig. 2). Additionally, two microcomputer-based subsystems were designed to perform specific tasks, the Sensor Management System (SMS) and the Stimuli Display System (SDS), and two commercial products were integrated in the system, the Eye Tracking System (ETS), that estimates the driver's point of gaze on the scene camera, and the Radar Scanning System (RSS). Argos software modules are depicted in Fig. 3 in which a summary of the hardware subsystems is represented in the top right corner, the main software modules in the top half and the auxiliary or specialized modules in the bottom half of the figure.

2.1.1 Control and Acquisition System (CAS)

The CAS manages the acquisition of all the numerical data gathered by the sensors distributed across the car, most of them provided by the SMS; it generates an overall system clock used to synchronize all the information stored in the system; and it manages the presentation of stimuli to the driver.

2.1.2 Image Storage and Management System (ISMS)

The main purpose of the ISMS is to capture, store, and distribute video images obtained from a high-resolution firewire scene camera and up to four additional PAL-format video inputs. Two of these inputs are connected to two video cameras, one of them pointing at the driver's face and a second one, located near the rear window, pointing at the road behind the vehicle; the third input receives video images from the ETS video output, and the fourth one is

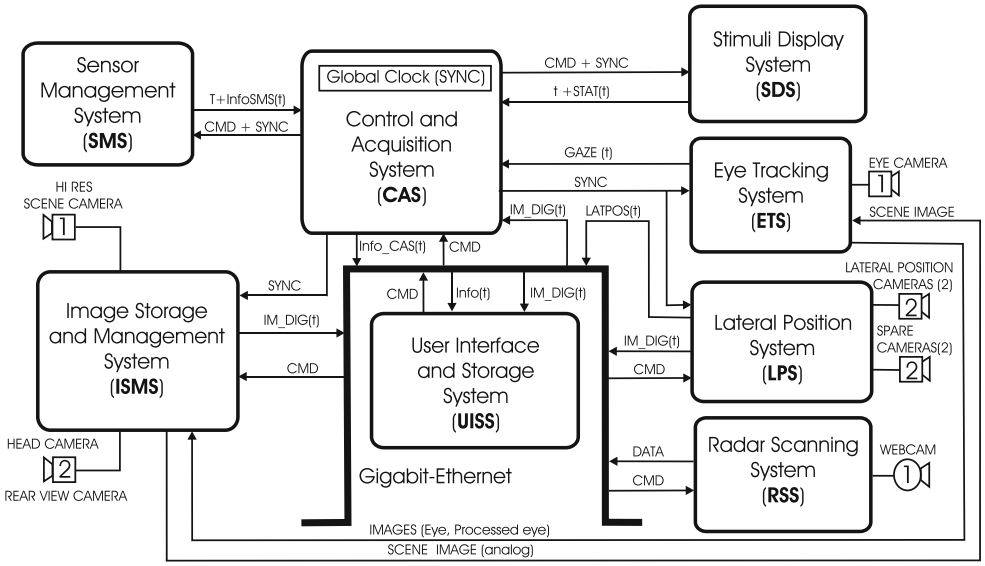

Fig. 2 Argos hardware architecture

provided to increase system flexibility allowing the experiment supervisor to connect an additional camera placed at any other point of the car depending on specific experiment requirements. For instance, the supervisor can place this extra camera over the driver's shoulder or head, pointing at the steering wheel or at the pedals.

Every video image frame is timestamped with the time code supplied by the CAS and stored in the ISMS along with a timestamp-based index so that video sequences can quickly be located. The experiment supervisor can select one or several video images to be shown on the displays connected to the CAS and to the UISS.

2.1.3 Lateral Position System (LPS)

The LPS has two main functions; the first is to estimate in real-time the distance to the road lane marks on both sides of the vehicle using image processing algorithms. It uses two PAL cameras installed in the place of the front fog lamps to capture the lane marks. In a different working mode, the LPS can also perform image processing algorithms to detect predefined visual beacons, as required in some experiments.

The second function of the LPS is to perform the same video image processing as described in the ISMS

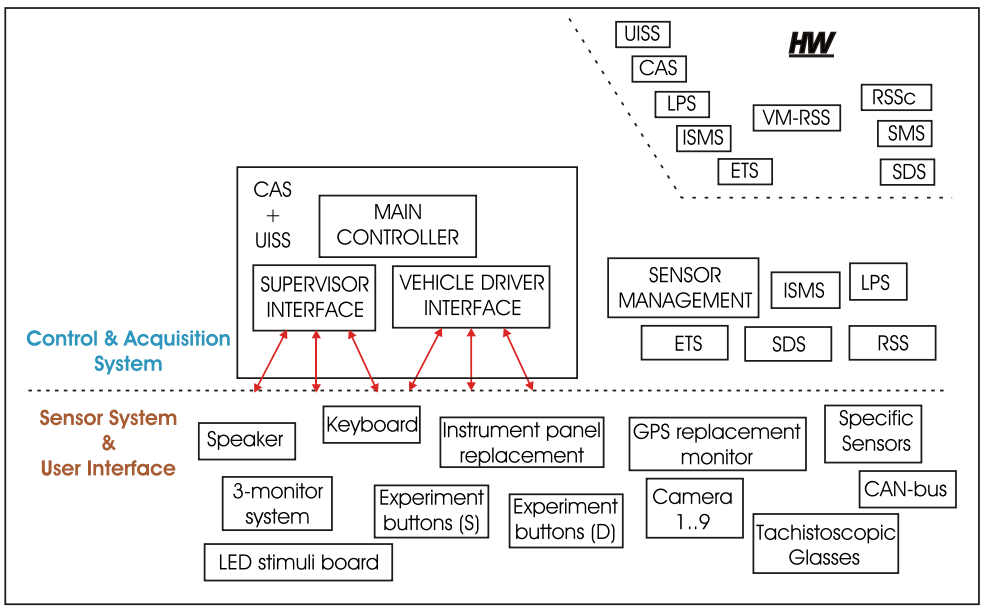

Fig. 3 Argos software architecture

subsection, i.e. to capture, store and distribute video images obtained from up to four additional cameras. Two are fixed and correspond to both lateral position cameras. The other two PAL inputs are not in use yet, but they are provided for future expansions of the system.

2.1.4 User Interface and Storage System (UISS)

The UISS is the Argos front-end that provides user access to the whole system through the supervisor's post to manage driving sessions, as well as for maintenance tasks, backups, etc. It coordinates all the computers and software modules and sends instructions to configure, start, manage, and end the driving sessions. The instructions may be generated by direct actions of the supervisor or by the Roadmap Program.

The UISS is responsible for storing all the information required for performing experiments and all the data collected by Argos during the driving sessions.

It executes the graphical user-interface application that allows the supervisor to monitor any signal and video image. If the scene image is selected, the UISS superimposes a mark on the coordinates of the driver point of gaze. The user interface also allows the supervisor to activate certain stimuli and to interact with the Roadmap Program Interpreter to control the experiments. After completing a session, the UISS allows the supervisor to replay the whole session or selected fragments, or to extract relevant selected data into text-formatted files.

Finally, the UISS executes the Roadmap Program Interpreter, which is a tool designed to manage complex experiments and fulfills the supervisor's role in unattended experiments. This tool is responsible for interpreting the Roadmap Program written in ESL. It monitors all the session data stated as relevant in the Roadmap Program and uses them to control the experiment. The monitored session data include the signals collected by the subsystems, the driver's and the supervisor's push-buttons, and the commands directly issued by the supervisor. On the other hand, the interpreter can trigger the activation of different kinds of pre-programmed or generated stimuli: luminous, acoustic, graphical, etc, so the interpreter can interact with the system, the driver and the supervisor. Graphics can be generated in real time, for example to reproduce the instrument panel of the car showing actual values collected by the system or others artificially altered by the Roadmap Program (see Fig. 4).

2.2 Exploitation of Argos

The new Argos was delivered to the DGT in 2008. Since then it has mainly been used in two projects co-funded by the European Commission under the Seventh Framework Programme of the European Union. The first one is called "Integrated Human Modelling and Simulation to support Human Error Risk Analysis of Partially Autonomous Driver Assistance Systems (ISi-PADAS)" [3]. Within this project the Foundation for the Research and Development in Transport and Energy (CIDAUT) has used Argos to gather information about the behavior of drivers of diverse age and gender under different real traffic scenarios, analyzing the influence of various factors related to distraction. Scenarios consisted of a categorization of driving maneuvers in the following situations: free driving, car following, lane change, overtaking, approaching a slower vehicle, and approaching a traffic light. In some particular scenarios, a distracting cognitive secondary task was given to the driver. About 20 drivers participated in the study in 80 minute driving sessions. As a result of these field tests, a set of enriched data about real driving behavior was generated and is currently being analyzed. CIDAUT has developed this research as part of the project to support the conception of a new driver assistance system, aimed at improving longitudinal driving by means of information, warning and intervention strategies.

Fig. 4 Reproduced instrument panel

The second project, "Promoting real Life Observations for Gaining Understanding of road user behaviour in Europe" (PROLOGUE), aims "to contribute to reducing the number of road casualties in Europe by further developing and testing the naturalistic observation methodology" [11]. At the time of writing, the Argos car is being used by the Institute of Traffic and Road Safety (INTRAS) of the University of Valencia, which are partners of the PROLOGUE consortium. As part of its research, a suite of unattended 2-3 hour sessions is being performed under real traffic conditions.

3 Roadmap Program requirements

Experiments usually have to be designed and fully controlled by the experiment supervisor, and performed by the

driver. The supervisor usually divides the experiment into several stages, each one oriented to achieve specific objectives, and the different stages can be connected in a complex way. This complex structure may require the supervisor to spontaneously introduce small changes depending on the specific experiments. The dynamic characteristics of the experiments require Argos to be equipped with an automated system that facilitates controlling experiments and modifying their stages.

These considerations led Argos designers to build a tool that would allow the supervisor to formally specify complex and controlled experiments, having to fulfill the following requirements:

- It has to be based on a powerful but easy-to-learn programming language, allowing the supervisor to program the experiment with a few sentences.
- It has to be embedded in the acquisition software, to enable on-line access to the recently acquired data.
- It has to be able to accept commands from the supervisor during the driving session. These commands have to be evaluated in the same way as data acquired during the driving session.
- It has to be able to influence the driver during the driving session by managing the tachistoscopic glasses, the emulated instrument panel, the stimuli board, and the acoustic stimuli.

4 ESL features

Before starting the experiment, the supervisor has to schedule the tasks to be performed by the driver during the driving session. These tasks are coded in a Roadmap Program written in ESL. The Argos designers provided the system with this imperative high level language assuming that although the supervisor is a computer user, not a software engineer, he/she would be familiarized with the use of this kind of languages, such as Visual Basic, embedded in very popular applications. The language is tightly typed in order to prevent the supervisor from introducing errors, thus avoiding assignments between variables belonging to different data types.

The Roadmap Program is divided into two sections: preamble and code. The preamble contains the complex types and variable definitions. The code section contains the ESL sentences that are fetched and executed. Several mechanisms have been provided to allow accessing the data being acquired during the driving session.

The program is checked before the first execution, as soon as it is loaded, in order to ensure that no syntax error is present. When the experiment starts, the first execution of the whole program initializes the devices to be used in the experiment and memory is allocated for storing variables. Further executions are triggered when a relevant data item is received by the UISS from any system. The code section is then completely executed in order to modify the driving session conditions if necessary. The following subsections detail the most important aspects of the language.

4.1 Data types

Basic data types are the types usually found in imperative high-level languages [4]: characters, integers (signed and unsigned), floating point numbers and strings.

In order to cope with data abstractions in the development of experiments, complex data types have been included in ESL. A complex data type is a set of basic data types, grouped in order to form an abstract data type. The definitions of complex data types have to be included in the preamble before the definition of the variables that will be used in the experiment. These types are used to model different vehicle devices and instruments to interact with the driver, for example, the RPM and speed gauges, the acoustic and visual stimuli, several icons to appear on the instrument panel, etc. Several complex data types are predefined to help the supervisor represent the emulated instrument panel, the visual and acoustic stimuli, and the tachistoscopic glasses. The following paragraphs describe the complex data types to model the experiment environment.

4.1.1 Emulated instrument panel data types

A set of predefined types is provided to graphically display data emulating the original car panel (see Fig. 4). Every symbol has the necessary fields to model the position in the panel where it has to be displayed, as well as its size and color. The types provided model the following symbols:

- Circles, triangles, and rectangles.
- Arrows. This data type includes the angle. In this way, left, right, up, and down arrows can be displayed.
- Icons. Some industry de-facto standard icons are included, such as low-fuel, low beam lights, or high beam lights.
- Text. Textual box can be displayed by selecting its font size, text color, and panel position.
- Bars. Vertical and horizontal bars can be displayed on the panel.
- Sectors. They can be used to show the coolant temperature and the fuel tank level.
- Gauges. Used to represent the speed and the RPM tachometer.

The last three types have additional similar fields and they include minimum and maximum values, number of

divisions in the scale, size, position on the panel, label, and color.

The supervisor can build different instrument panels by defining the `Instrument_Panel` complex type and including fields belonging to the types listed above. Figure 5 shows the definition of a simple instrument panel containing two gauges for engine RPM and car speed, and a text box. The code section contains the `speed` field initialization, setting 220km/h in a speed gauge with a radius of 180 pixels, in red color, at the (100, 100) position of the instrument panel, and a range of [0,220]km/h. The `rpm` field is built from the `speed` field and only `rpm` specific fields are modified in order to make compact programs. The RPM gauge displays the value 2000, at (100, 200) and the gauge is [0, 5000]RPM.

4.1.2 Visual stimuli data types

The high-luminosity LEDs are arranged in two rows and eight columns of 5×7 cells (see Fig. 1), each one controlled independently and devoted to projecting a character or a shape on the windshield. This approach implies that the UISS has to send reverse characters to the cells. Every cell is modelled by using a variable belonging to the `Visual_Stim` type specifying the complete stimuli board. Thus, the led array is modelled as an array of 16 elements of this type. This type contains the number of the cell and the five columns that represent the character.

The ESL provides the `led_set` function (described below) in order to define the characters to be displayed in the light stimuli board.

4.1.3 Acoustic stimuli and peripheral data types

Acoustic stimuli are modelled by using the three field `Acoustic_Stim` data type. The `file` field specifies the mp3 file containing the acoustic stimulus and the `start` and `length` fields specify the starting of the stimulus specified in seconds from the beginning of the file and how many seconds the stimulus has to be played, respectively.

A similar data type is also predefined for handling the tachistoscopic glasses.

4.2 Sentences, operators and functions

ESL contains the control structures of imperative programming languages such as C, Fortran or Visual Basic: `if`, `for`, and `while`. The arithmetic, logical and relational operators are identical to those of C language.

The ESL provides the supervisor with several embedded functions to facilitate experiment programming. Some functions are included to convert variables between types while others deal with strings: concatenation, comparison, searching for a substring in a string, etc. Random number generation is included in ESL to support simulation of random situations, and several random variable generators are available to the programmer. Several clock related functions are included allowing ESL to measure the elapsed time between two events with millisecond precision.

We will pay special interest to signal handling functions used for obtaining data being acquired and functions to interact with the driver, which are explained in depth in the next sections.

4.2.1 Signal handling functions

Signal data access is necessary for the ESL program to be able to take decisions depending on signal values. Hence the first requirement is to be able to select relevant signals. Two functions have been included for this purpose:

- `get_id_signal` associates a name to a signal descriptor. In this way, vehicle signals can be known by their assigned names.

```
struct {
   Gauge speed, rpm;
   Text text;
} Instrument_Panel; # Definition of type
Instrument_Panel panel;

begin_test:"Example"
# Gauge fields = {x, y, size, range_min, ...
          ..., range_max, color, value, tag}
   panel.speed = {100, 100, 180, 0, 220, 0x0f0000,
                  220, "Speed"};
   panel.rpm = panel.speed;
   panel.rpm.y = 200;
   panel.rpm.range_max = 5000;
   panel.rpm.value = 2000;
   panel.rpm.tag = "RPM";
   panel.text.value = "Ticketted!!";
```

Fig. 5 Definition and initialization of a basic instrument panel

– `select_signal` notifies the system that the ESL programmer wants to take decisions based on the values of the specified signal. The number of seconds of interest also has to be specified in order to allocate the necessary memory.

Several functions are provided in order to give the programmer more freedom to obtain the signal values:

– `get_signal` obtains the last acquired value of a signal.
– `get_signal_min`, `get_signal_max`, and `get_signal_avg` give the minimum, maximum, and average values during the relevant interval.
– Two additional functions to manage operator codes allow marking specific instants of the session or influencing the driver environment: `insert_code` records an operator code in the session files and `get_code` obtains the last operator code inserted in the session.

4.2.2 Interacting with the driver

Some functions have been included to be able to interact with or modify the driver environment through the instrument panel and the visual and acoustic stimuli. `update_panel` is used by the programmer to display symbols on the instrument panel. The programmer specifies the display where the symbols have to be displayed (GPS or the instrument panel) and a variable belonging to the `Instrument_Panel` type containing all the symbols to be used.

Visual and acoustic stimuli are played by using similar functions: `update_stimuli`. This function receives a `Visual_Stim` or `Acoustic_Stim` variable and plays the stimulus accordingly. Acoustic stimulus files are only sent once to the CAS by the initialization section.

An additional function has been included to manage visual stimuli so as to make it easy for the programmer to draw a character. `led_set` receives a `Visual_Stim` variable and four integers specifying a rectangle to be filled without modifying the bits that are outside the specified area.

Similarly, the `activate_peripheral` function is provided to modify the status of the tachistoscopic glasses.

4.2.3 Sample program

Figure 6 shows an example of obtaining the value of a signal (RPM) that has to be displayed on a gauge in the instrument panel. Thus, an instrument panel containing the RPM gauge is defined in the preamble. The initialization of this structure is completed inside the first `if` sentence of the program code. This sentence is only executed during the first execution of the program because the `initialized` variable contains the value 0. Inside this sentence, the program "notifies" the interpreter that the RPM signal is going to be used to take decisions, and to store RPM data for the last two seconds by calling the `select_signal` function.

When a data set is received from any system, the program is executed and the last RPM data are obtained and rounded to the nearest integer (lines 15 to the end of the code).

```
struct {
   Gauge rpm;
} Instrument_Panel;      # Definition of type
Instrument_Panel panel; # panel is an instrument panel
integer initialized, rpm, aux, prev_rpm;
float fl;

begin_test:"Update_RPM"
if (initialized == 0) then
    .......              # Code for initialization of rpm gauge
    rpm=id_senal("RPM");  # Get signal id
    select_signal(rpm,2); # Two seconds of history
    initialized = 1;      # This code will not be executed later
fi
fl=get_signal(rpm);       # Obtaining the signal value (float)
round(value, fl);         # Round it to the nearest integer
aux = prev_rpm - value;   # Compute the absolute value of the difference
if (aux < 0) then         # with the previous one
     aux = -aux;
fi
if (aux > 100) then
     prev_rpm = value;    # If rpm changes more than 100
     panel.rpm.value = value;
     update_panel(0,panel); # Sending panel updating
fi
end_test
```

Fig. 6 Obtaining the value of a signal and updating the instrument panel

The program checks the difference between the recently acquired data item and the previous value in lines 17 to 20. If the absolute value of this difference is significant (greater than 100), the instrument panel is updated in lines 22 to 24.

5 Example: estimating arrival time

Estimating arrival time is an experiment promoted by the Spanish Traffic Agency to study the speed and time-to-collision perception of young car drivers. The experiment was carried out by selecting participants from a chosen group of drivers of different gender and age to travel in the front passenger's seat. The sample was composed by 20 male and 20 female drivers aged between 20 and 30.

Participants were deprived of vision for a variable distance to a target located at the end of a straight lane. They were required to estimate the instant when the car would reach the target. Different constant speeds and different predefined distances were used.

In order to properly perform these experiments, a lane was marked by using 6 beacons and 1 target at the end. Beacons were situated at distances of 225, 150, 125, 100, 75, and 0 meters to the target. The Roadmap Program was in charge of controlling the instants when the passenger had to be blindfolded depending on the beacon signals and the predefined distances.Participants started the experiment deprived of vision, they recovered vision at 225 m and, at predefined distances (150, 125, 100, or 75 m), they were deprived of vision and asked to push a button when they judged that the vehicle reached the target position.

The Roadmap Program makes it easy to automate these experiments, as well as to control some additional Argos equipment that was also needed:

- tachistoscopic glasses to blindfold the passenger. They are activated or deactivated by the CAS when the Roadmap Program sends a command in response to the detection of beacons.
- beacon signals are generated by image processing algorithms running on LPS.
- the UISS executes the Roadmap Program but it is also in charge of monitoring several parameters such as the speed. Thus, if there are significant variations (20 %) the experiment is invalidated by the Roadmap Program and has to be repeated.
- the ISMS processes and stores data from the high-resolution scene camera in order to be reproduced in the laboratory.
- the SMS monitors the button that participants have to push when they think that the target has been reached.
- the CAS receives an mp3 file containing a sound that has to be played when the experiment is cancelled. The UISS decides when an experiment needs to be cancelled.

The Roadmap Program that manages this experiment uses four types of operator codes used as commands to configure the experiment:

- "c" code. The experiment has to be cancelled because the supervisor detects something wrong in the experiment.
- "vxxx" codes. Used to specify the constant speed for the experiment. For example, the code "v100" specifies that the target speed is 100 km/h and the possible variation for the speed ranges from 80 to 120 km/h.
- "b1, b2, b3, and b4" codes. Used to specify the beacon at which the participant is deprived of vision. They indicate the distance where the participant is blindfolded: 150, 125, 100, and 75 m respectively.
- "s" code. Used to specify the experiment starting instant, i.e. the participant is blindfolded until the first beacon is reached.

The Roadmap Program is in charge of performing every action to carry out the experiment, ensuring that the car speed is correct and that no beacon is missed by establishing a timeout for the experiment, i.e. if 30 seconds after starting the experiment the Roadmap Program has not received the signal indicating that the participant has pressed the button, the experiment is automatically cancelled. In order to achieve these objectives, the Roadmap Program is divided into four parts:

- the configuration segment is in charge of establishing the experiment parameters by using the operator codes;
- the evaluation segment is in charge of testing the beacon signal, activating the tachistoscopic glasses to blindfold the participant, and ensuring that the speed is in the range specified by the supervisor;
- the cancel segment is in charge of notifying the participant that the experiment has been cancelled.

The current status of the experiment is stored in an integer variable with four possible values:

- 0. The experiment has not started (the supervisor has not yet sent the "s" code).
- 1. The experiment has started, the participant is blindfolded and the program is waiting to reach the first beacon.
- 2. The experiment has started and the first beacon has been reached. The participant is not blindfolded and the program is waiting for the beacon the supervisor specified by using one of the "b1", "b2", "b3", or "b4" codes.

– 3. The participant is blindfolded again because the beacon has been reached and the program is waiting for the participant to press the button and for the last beacon to be reached.

In Fig. 7 the preamble and the initialization segments are shown. These segments contain the variable definitions and the `if` sentence that is executed during the first execution of the program. This sentence body contains the declaration of the set of signals that will be used in the experiment: the speed, the beacon to identify the current experiment phase; and the red button the participant has to press when he/she estimates the car is reaching the final beacon. The selection of these signals is notified to the ESL interpreter by calling the `get_id_signal` and `select_signal` functions. On the other hand, the participant of a cancelled experiment is notified by an acoustic stimulus, contained in an mp3 file that is delivered to the CAS during the first execution of the Roadmap Program by calling the `update_stimuli` function. The tachistoscopic glasses are initialized in transparent status so the participant is not blindfolded. The blindfolding interval is set to 30 seconds, which is only a timeout, because the tachistoscopic glasses are fully controlled by the Roadmap Program. Finally, the `status` and `cancel` variables are set to 0, the timeout for the experiment is set to 30 seconds and the experiment target speed is set to 60 km/h. The last sentence sets the `initialized` variable to 1, preventing the `if` sentence from being executed again.

In Fig. 8 the configuration segment containing the acquisition of user codes is shown. This code segment starts by recovering the last user code introduced. If no code is returned, no action is taken. Otherwise, user codes returned in the `usr_code` variable are checked. As soon as the "s" code is detected the experiment is started by blindfolding the participant, the tachistoscopic glasses are activated by calling the `activate_peripheral` function, the `status` changes to 1 (waiting for the first beacon), the number of registered beacons is set to 0 and the timeout to end the experiment is stored in the `end_experiment` variable in milliseconds.

If the "c" code has been acquired, the `cancelled` variable is set to 1 in order to cancel the experiment at the end of the Roadmap Program.

If the first character of `usr_code` is set to "b", one of the b1, b2, b3 or b4 codes has been introduced. So, the second character of `usr_code` is obtained and stored in the `target_beacon` variable.

```
Acoustic_Stim sound;
Perif_Tachistoscop tach;
integer initialized, error, new_beacon, ret_val, cancelled, change;
unsigned:64 end_experiment, current_time, timeout;
integer value, beacon, ant_beacon, speed, button, n_beacons, target_beacon;
integer min_speed, max_speed, target_speed, status;
float fl;
string buffer[200], usr_code[20];
begin_test:"Estimation";
if (initialized == 0) then
    beacon=get_id_signal("Beacon");
    select_signal(beacon,1);
    speed=get_id_signal("Speed");
    select_signal(speed,1);
    button=get_id_signal("Red_button");
    select_signal(button,1);
    sound.file="cancel.mp3";
    sound.start=0; sound.length=100;
    ret_val=update_stimuli(sound);
    tach.activation = 0;
    tach.length = 30;
    tach.active = false;
    ret_val=activate_peripheral(tach);
    tach.activation = 1;
    timeout=30;
    target_speed=60;
    min_speed=target_speed*8/10;
    max_speed=target_speed*12/10;
    status=0; cancel=0;
    n_beacons=0; error = 0;
    initialized=1;
fi
```

Fig. 7 Preamble and initialization code segment

```
value=get_code(usr_code);
if (value == 1) then
    if (strcmp(usr_code,"s") == 0) then
        n_beacons=0;
        status = tach.activation = 1;
        ret_val=activate_peripheral(tach);
        end_experiment=get_local_time()+ timeout * 1000;
    else
        if (strcmp(usr_code,"c") == 0) then
            cancelled=1;
        else
            substring(buffer, 0, usr_code, 1);
            if (strcmp(buffer,"b") == 0) then
                substring(buffer, 1, usr_code, 20);
                ret_val=str2int(target_beacon, buffer);
                if (ret_val == 0) then
                    if (target_beacon < 1 | target_beacon > 4) then
                        error = 1;
                        target_beacon = 0;
                    fi
                else
                    error = 1;
                fi
            fi
            if (strcmp(buffer,"v") == 0) then
                substring(buffer, 1, usr_code, 20);
                ret_val=str2int(target_speed, buffer);
                if (ret_val == 0) then
                    min_speed=target_speed*8/10;
                    max_speed=target_speed*12/10;
                else
                    error = 1;
                fi
            fi
        fi
    fi
    if (error == 1) then
        print("User code error. Please check it!",1);
        error = 0;
    fi
fi
```

Fig. 8 Configuration segment

If the first character of `usr_code` is set to "v", the target speed is established. So, the number that follows this character is obtained and the `min_speed` and `max_speed` variables are updated with the speed range. If there is any error in recovering the user code, the supervisor is notified.

Figure 9 contains the evaluation segment showing how the experiment is managed. If the status of the experiment is not 0, the last value of the "Beacon" signal is obtained. If this value has changed from 0 to 1, a new beacon has been reached and the `new_beacon` variable is set to 1. When a new beacon is detected and the experiment status is 1, the `activate_peripheral` function is called and the participant recovers vision while the experiment changes to status 2. If the experiment status is 2 or 3, the number of beacons detected, `n_beacons`, is incremented. For status 2, if the value of this variable reaches the `target_beacon` value the participant has to be blindfolded and the status is updated to 3. For status 3, the program waits for the participant to press the button and for the car to reach the target beacon. Both events are registered by including a user code by means of the `insert_code` function. The speed value is obtained by calling `get_signal`. If the integer speed value is greater than `max_speed` or lower than `min_speed` it means that the speed is out of range. This event is registered by recording a user code in the session. Finally, the current time is compared with the timeout stored in the `end_experiment` variable. If this timeout has been exceeded, the experiment is cancelled and the `timeout` user code is inserted.

The code finishes checking the `cancelled` variable. If it is set, the experiment has to be cancelled and an acoustic stimulus is played to notify the participant.

```
if (status > 0) then
    new_beacon = 0;
    fl=get_signal(beacon); round(value, fl);
    if (value != ant_beacon) then
        if (ant_beacon == 0) then
            new_beacon = 1;
        fi
        ant_beacon = value;
    fi
    if (new_beacon == 1) then
        if (status == 1) then
            tach.activation = 0; status = 2;
            ret_val=activate_peripheral(tach);
        else
            n_beacons = n_beacons + 1;
            if (status == 2) then
                if (n_beacons == target_beacon) then
                    tach.activation = 1; status = 3;
                    ret_val=activate_peripheral(tach);
                fi
            else
                if (n_beacons == 5) then
                    insert_code("target");
                fi
                fl=get_signal(button); round(value, fl);
                if (value == 1) then
                    insert_code("buttonpressed");
                    status = 0;
                fi
            fi
        fi
    fi
   fl=get_signal(speed); round(value, fl);
   if (value < min_speed | value > max_speed) then
       insert_code("wrongspeed"); cancelled=1;
   fi
   current_time=get_local_time();
   if (current_time > end_experiment) then
       insert_code("timeout"); cancelled=1;
   fi
fi
```

Fig. 9 Evaluation segment

6 Conclusion

Car accidents represent a notable percentage of deaths or serious injuries, which makes the study of driver behavior a crucial issue for understanding and improving road safety. Some studies can be carried out by means of driving simulators, but driving in real traffic conditions makes a great difference in obtaining significant results. However, there is a small number of instrumented vehicles and most of them are limited to logging vehicle or ambient data (IVDR).

The authors have designed and built a multi-sensor and interactive experimental vehicle, Argos, a complex system built around a multicomputer platform. It is based on state of the art technologies: last generation sensors, advanced human machine interfaces, image compression, video on disk recording, image processing and pattern recognition algorithms. These features, together with the fact that open software operating systems running on standard hardware PCs are the base of the main subsystems, complemented by major off-the-shelf components, ensure a long life expectancy for the system.

An innovative component of the Argos platform is the Roadmap Interpreter, a powerful tool that allows experiment designers to write experiment programs specifying the different driving session stages and the actions that have to be performed when relevant events occur. Experiment programs run under control of the Roadmap Program Interpreter that is embedded in the User Interface and Storage System, which allows to interact with the driver and the experiment supervisor. This tool-set has been used to design some experiments [10] such as the example described in this paper. This example includes sentences specifying interaction with both the experiment supervisor (by means of user codes that the

Roadmap Program manages as commands) and the driver or participant (through the tachistoscopic glasses and the steering wheel push buttons).

By incorporating the Roadmap Interpreter and the Experiment Specification Language, Argos has turned into a sophisticated tool reducing the usual time required to develop experiments and opening up possibilities for designing highly complex experiments, allowing designers to reuse previous work and to concentrate on the experiment target rather than the details of running the experiment. In fact, with the new version of the car (with ESL), the experiment setup takes a few hours instead of several days as was the case with the previous version.

References

1. Comte L (2000) New systems: new behaviour Trans Res F Traffic Psychology and Behaviour 3(2):95–111
2. Doshi A, Trivedi M (2009) On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes. IEEE Trans Intell Transp Syst 10(3):453–462. doi:10.1109/TITS.2009.2026675
3. ISi-PADAS Project Home Page (2011) http://www.isi-padas.eu
4. Kernighan BW, Ritchie D (1978) The C Programming Language. Prentice-Hall
5. Liang Y, Reyes ML, Lee JD (2007) Real-time detection of driver cognitive distraction using support vector machines. IEEE Trans Intell Trans Syst 8(2):340–350
6. Ma X. Andreasson I (2007) Behavior measurement, analysis, and regime classification in car following. IEEE Trans Intell Transp Syst 8(1):144–156. doi:10.1109/TITS.2006.883111
7. Nunes L, Recarte M (1997) Argos program: Development of technological systems and research programs for driver behavior analysis under real traffic conditions. In: Proceedings of the international seminar on human factors in road traffic 2 (ISHFRT 2), pp 630–640
8. Nunes L, Recarte M (2002) Cognitive demands of hands-free-phone conversation while driving. Trans Res F Traffic Psychology and Behaviour 5(2):133–144
9. Pastor L, de Miguel P, Pérez A, Rosales F, Rodríguez S, Cabañas A, Rodríguez A (1993) Sensor techniques using image processing for driver behaviour study. In: Nwagboso CO (ed) Automotive sensory systems, chap 9. Chapman and Hall, pp 185–209
10. Pérez A, García M, Nieto M, Pedraza JL, Rodríguez S, Zamorano J (2010) Argos: An advanced in-vehicle data recorder on a massively sensorized vehicle for car driver behavior experimentation. IEEE Trans Intell Transp Syst 11(2):463–473. doi:10.1109/TITS.2010.2046323
11. (2011) PROLOGUE Project Home Page. http://www.prologue-eu.eu
12. Rakha H, El-Shawarby I, Setti JR (2007) Characterizing driver behavior on signalized intersection approaches at the onset of a yellow-phase trigger. IEEE Trans Intell Trans Syst 8(4):630–640
13. Recarte M, Nunes L (2000) Effects of verbal and spatial-imagery tasks on eye fixations while driving. J Exp Psychol Appl 5(2):31–43
14. Recarte M, Nunes L (2002) Mental load and loss of control over speed in real driving towards a theory of attentional speed control. Transp Res F Traffic Psychology and Behaviour 5(2):111–122
15. Recarte M, Nunes L (2003) Mental workload while driving: effects on visual search, discrimination and decision making. J Exp Psychol Appl 9(2):119–137
16. Sekizawa S, Inagaki S, Suzuki T, Hayakawa S, Tsuchida N, Tsuda T, Fujinami H (2007) Modeling and recognition of driving behavior based on stochastic switched ARX model. IEEE Trans Intell Trans Syst 8(4):593–606
17. Toledo T, Musicant O, Lotan T (2008) In-vehicle data recorders for monitoring and feedback on drivers' behavior. Transp Res C Emerg Technol 16(3):320–331