SVG for Automotive User Interfaces

Sébastien Boisgérault, Mohamad Othman Abdallah, Jean-Marc Temmos

SVG for Automotive User Interfaces

Dr. Sébastien Boisgérault <Sebastien.Boisgerault@ensmp.fr>
Mohamad Othman Abdallah
<mohamad.othman_abdallah@ensmp.fr>
Jean-Marc Temmos <jtemmos@visteon.com>

Abstract

In car cockpits, a wide range of graphic displays, from the low-end multi-functional devices to the most advanced reconfigurable clusters, represent an increasing part of the on-board information systems. We address within the EDONA HMI project the modeling of such human-machine interfaces (HMIs) and the development of an integrated HMI design environment that would improve current development practices. In this article we specifically discuss modeling issues: we explain why the SVG format was selected as the basis of the HMI graphic content description and present domain-specific extensions, mostly related to the HMI functional description, that provide support for a consistent modeling of HMI components.

Table of Contents

Introduction

Among human-machine interfaces (HMIs) found in car cockpits, many devices such as head units, instrument clusters and control panels, usually contain a mix of mechanical systems and low-end displays such as segmented displays. On the other hand, larger screens of higher size and quality are used increasingly, starting with high-end vehicles, prototypes and concept cars. These displays may replace classic HMI systems – for example implement a virtual instrument cluster – and provide advanced features, such as a full reconfigurability, that the classic physical device could not offer.

The Visteon/3M X-Wave concept instrument panel is a great example of the high-end scenario. It demonstrates, among several technologies, innovative use of HMI displays, either as information

systems – the driver information cluster located behind the steering wheel and the multi-functional display located in the middle of the cockpit – or as control devices like the integrated center panel (ICP). Instead of classic buttons, the ICP is a touch-sensitive display. Proximity sensing is used to switch from a dead-front look to an illuminated HMI when the driver's hand nears the panel, allowing him to control systems such as audio, climate or navigation in a few clicks.

Figure 1. Visteon/3M X-Wave concept car cockpit

Figure 2. Inactive and active integrated center panel

The EDONA HMI subproject addresses the modeling of this wide range of HMI displays, from the low-end displays to the high-end devices used in the most advanced applications. The project aims to provide a model-based, integrated design tool chain that will support their entire design process. In this document, we first present the EDONA project and open platform and the specific context of HMI design for the automotive industry. Then we describe the aims and components of the EDONA HMI design tool chain as well as its integration into existing processes. Finally we turn our attention to HMI models and their structure, and present how the SVG standards may be used in their core graphic model and what kind of application-specific extensions we propose to obtain an adequate description of HMI models.

EDONA and Human-Machine Interface Design for the Automotive Industry

EDONA is an ongoing french project, started in september 2007, of the competitive cluster System@tic Paris-Région. It gathers the major national actors of the automotive industry under the leadership of Renault. This project promotes an approach based on shared development tools and standards for the design of embedded systems for the automotive. It aims to deliver an open technological platform hosting specialized development environments for the automotive industry. Such environments will be provided by EDONA sub-projects, including one dedicated to HMI design.

A suitable HMI design environment for this context has to take into account several characteristics of the automotive industry that we outline in this section

Interoperability

The automotive industry is structured around a large number of providers of cars components. The ability of these actors to easily exchange components specifications and designs and ultimately to seamlessly integrate these components into vehicles while maintaining a high productivity and quality level is a major issue. Selected standards contribute to the resolution of the interoperability issue: prominently, AUTOSAR, as an emerging automotive standard that targets integration of hardware and software component on ECUs (Electronic Computing Units) has a major role to play in this respect. But it focuses on the deployment stage ; on the other hand, standards for common models and formats as well as their support in software tools, integrated into consistent platforms, shall address the same issue at the design stage. In this matter, the development cycle of HMI components is no exception.

Safety-critical system design

As many other automotive embedded systems, the failure of a HMI system may generate high risk levels for the customers. Therefore, its development process shall guarantee that it produces software components that meet all necessary safety requirements. This issue is obviously shared with the production of HMIs in other safety-critical domains such as medical systems or avionic systems. The experience of the automotive industry with IEC61508 showed that non-automotive safety standards do not fully address either the real-time nature of embedded systems, or the automotive development and life cycles. The ISO26262 standard on the contrary specifically applies to safety-related E/E and software systems embedded in automotive and addresses hazards due to malfunctions. It embraces a customer risk-based approach for the determination of the risks at the vehicle level. This standard does not explicitly enforces the set of development methods to adopt in order to achieve safety-related goals. However, prior experiences in avionics, a related field where safety constraints are particularly important, have already shown the benefits of using a model-driven development process. Focusing the development of software component on their model – a formal specification of their behavior – and using code generation to actually produce the components, reduces ambiguity and lack of accuracy in the specifications, allows early detection of design errors, enables the application of formal verification methods and avoids the errors introduced by manual coding.

Embedded platforms

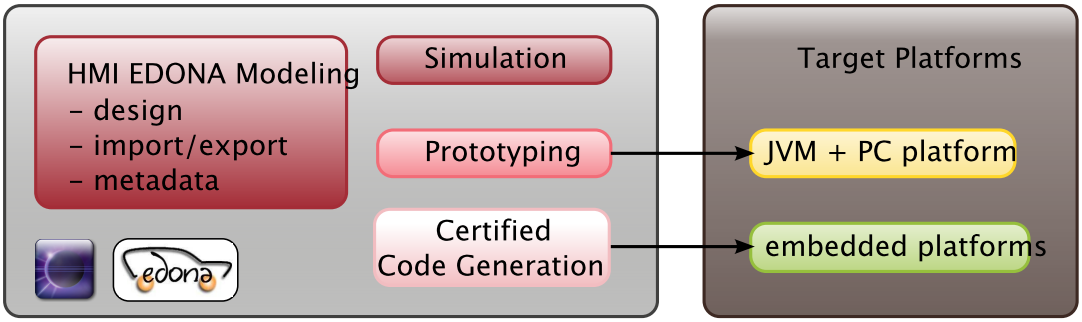
HMI software systems designed for road vehicles are used in wide range of application contexts and may accordingly be embedded in a large panel of targets. Beyond production-ready HMI aimed at mass-market vehicles, there is a continuum of situations: HMI design for high-end vehicles, proofs of concept and prototype vehicles, research demonstrators in the context of intelligent transportation systems, etc. The requirements for the HMI platform vary greatly according to the application context. For cost reasons, simple HMI components that are usually used for audio systems control (Radio/Tuner AM/FM, CD, MP3, etc.) or climate control (EATC) may rely on simple multi-functional display, low-end processors and restricted graphics library. On the other side of the range, prototypes and academic

demonstration platform commonly use embedded PCs, large color displays and touch screens, a combination that ensures the best user experience and enables rapid application development as well. A complete tool chain solution for HMI design in automotive has to provide adequate support for this large panel of use-cases.

EDONA HMI Design Environment

The EDONA HMI project aims to deliver an integrated design environment that supports the full embedded HMI design cycle, from the early stages of system specification to the generation, configuration and integration of executable software components on embedded targets. This tool chain is structured around modeling services and two execution infrastructures.

Figure 3. EDONA HMI Tool Chain



HMI model design

This module gathers all services related to the creation, transformation and management of HMI models. It includes a model editor that allow design of new components and assembly of existing components, import/export services for external HMI models, and validation services. Management of HMI models include support for documentation, internationalization, quality standards and requirements.

HMI simulation and prototyping

The first execution infrastructure for HMI components focuses on integration with the development environment, rapid application development, and design of prototypes for advanced applications.

Its target platform is desktop or embedded PCs equipped with the Java runtime. Using a simulation layer, it is tightly integrated into the EDONA HMI development environment, an Eclipse platform. It is then used to test, improve and validate HMI models in a fast design-simulation iteration loop. For the design of advanced automotive prototypes and demonstrators, especially in the context of intelligent transport systems, that are not subject to high safety and cost related constraints, the same execution architecture is also adequate.

HMI code generation for embedded platforms

The second execution infrastructure for HMI components targets embedded platforms, and takes into account hardware and software constraints as well as safety and certification requirements.

This module uses HMI models to generate automatically C code of the HMI software component, avoiding manual coding and the associated risks of specification interpretation and coding errors. This is clearly a central feature as in safety-related development, a major part of the software component development life-cycle is consumed by verification tasks: verification cost may account for as much as 80% of the budget. The model-based approach reduces the need for code testing and allows most

verification tasks to be performed at the model-level. Automatic code generation also allows the production of simple, deterministic and efficient code. Code retargetability is an additional benefit: generators may be configured to produce a code adapted to given target platforms, and for example use fixed-point representation for numbers if floating-points are not supported.

HMI Model Contents

A survey of common use cases, tools and standards applicable in automotive HMI design was performed as one of the first steps of the EDONA HMI project. Its conclusions deliver a set of requirements for the contents of a suitable HMI model, a prerequisite for the specification of the format. First of all, the scope of the modeling was considered and if a wide range of HMIs are included in the scope of our meta-model, some specialized subsystems, that require 3D models or video displays were excluded for being too different in nature from our core target. Most of the HMI display content, from multi-functional devices to reconfigurable clusters, may be adequately described with 2D graphics only: this graphic layer is addressed in the first sub-section. For the high-end displays that display other kinds of graphic content, we focus on the ability to integrate with these subsystems rather than including them in the model. For this reason among others, HMIs require a consistent component interface model, described in the second subsection. Micro-functional constructs, compatible with this interface model, allow the components to present a high-level and safe interface. They are discussed in the third subsection. Finally, the constraints attached to additional HMI data such as documentation or internationalization content are considered.

Graphic Layer

The HMI graphic layer is the part of its model that describes all possible appearances of the display at runtime. It therefore gathers all the elements of the HMI specifications related to graphic layout, dimensions, shapes, colors, ans so on. Two basic schemes are behind any change in the HMI display. The simplest one is the switch from a graphic layer to another, among a fixed set, for example as it may happen to a multi-functional display when a user switches from the audio control mode the climate control mode. A more complex source of animations in graphics relies on parametric change in the current graphics content. The parameters may control shapes, geometric transformations, styles or text content. For example, a digital gauge needle rotation may be controlled to display the current RPM rate.

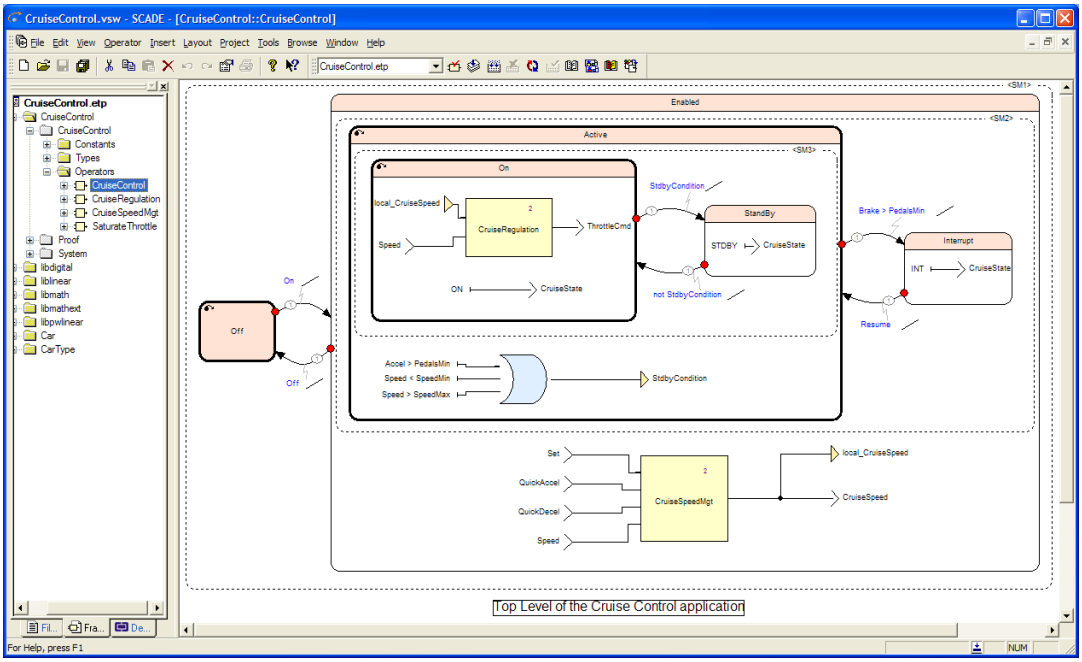
The ability to create totally new HMI appearances also requires a low-level description of its graphics elements. Support for component-level assembly of HMI graphic models is also considered for its ability to favor rapid design cycles. Beyond these general constraints, a detailed list features that were required by common HMI models was established. All these requirements clearly plead in favor of a structured vectorial model for the graphics content.

Figure 4. Example Radio/Tuner/CD Display, using transparency for the volume display

Component Interface

The graphic content of an HMI component is controlled by its external environment. Such component may also export information about its current graphic state. The specification of this interface layer has to be compatible with the models of computation that are commonly used to model HMI logic. Our survey demonstrates that HMI design tools are dominated by the graphical versions of synchronous languages that are popular in the context of embedded software design. These languages may leverage powerful verification methods and allow efficient code generation. They are either based on data-flow diagrams or state charts families. Some solutions, like SCADE, Esterel Technologies safety-critical application development environment, even support constructs of both types.

Figure 5. Scade cruise control functional model



A simple interface model that can be integrated with this family of functionals languages consists in a fixed list of typed input and output signals that are always synchronously updated. The HMI component is repeatedly activated: new values are provided for the input signals and in return, the HMI component shall provide a new value for every output.

Micro-Functional Constructs

The description of full HMI logic or functional layer is beyond the scope of the EDONA HMI model. However, the inclusion of some simple functional constructs – intermediate components between component interface signals and graphic data – is necessary to provide an interface that is high-level enough. Without any such construct, a gauge needle would be controllable only by its rotation angle and not with the functional value that it displays (km/h, RPM, etc.), and the interface of a 7-segment display would consist in a sequence of inputs that control each segment visibility instead of a signal input with the digit that the display shall render.

Therefore, a minimal data-flow model of computation – referred to as micro-functional layer in the sequel – was selected for this purpose. It extends the expressivity of the HMI component model but does not alter the nature of the component interface.

Additional Data

An HMI specification contains some context information that complements its graphic, interface, and micro-functional content. Documentation, requirement traceability, internationalization support are among the kind of extra information that shall be embeddable in the HMI models.

SVG standards for automotive HMI modeling

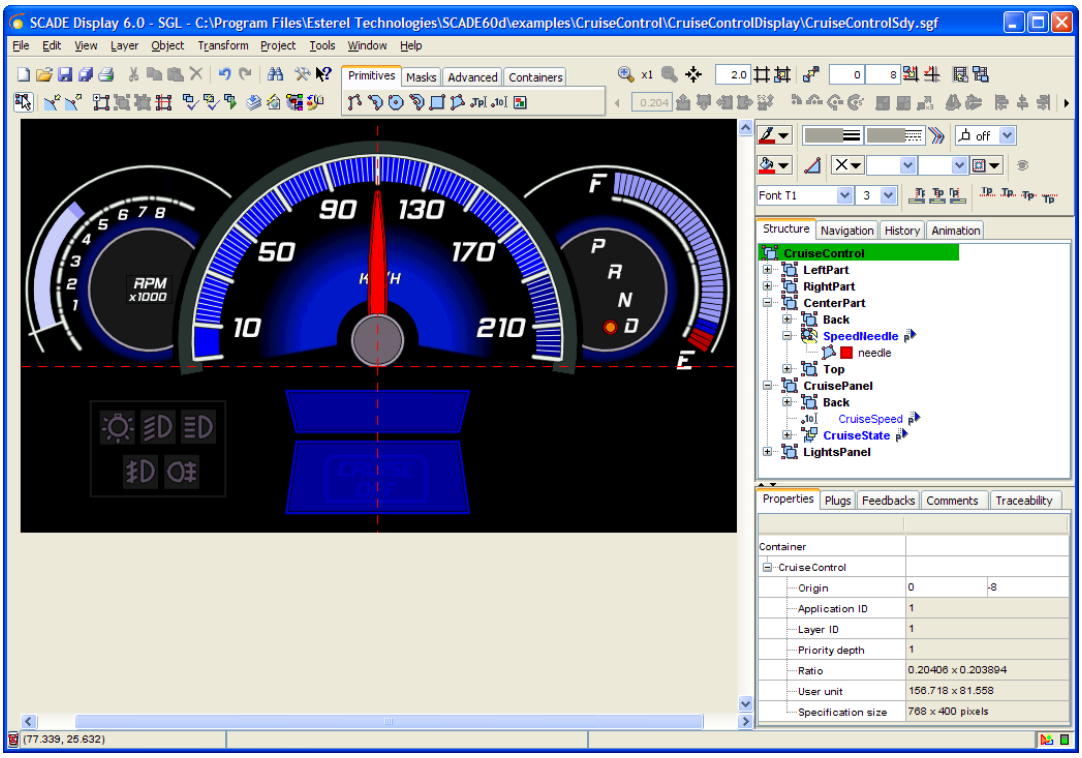
SVG 1.1 and its mobile profiles are mature and popular standards created by the W3C and their usage scenarios include the description of graphical user interfaces. They were therefore considered early as candidate formats for being the core graphic layer that would be extended to describe complete automotive HMI models.

In this regard, the SVG 1.1 set of standards provides consistent restriction and extension policies. SVG 1.1 is described as a set of modules that simplifies the definition of profiles, XML dialects that are subsets of the original specification. The W3C recommends two such subsets, SVG Tiny 1.1 and SVG Basic 1.1. To these standard profiles should be added industrial initiatives such as SVG Tiny 1.1 Plus that originates from the mobile phone ecosystem. These profiles are specifically designed for mobile devices, platforms that are primarily characterized by specific constraints in terms of CPU speed, memory size, color support, etc. The family of mobile devices obviously contains mobile phones and personal digital assistants but the target platforms for embedded HMI in road vehicles share similar constraints.

SVG 1.1 also comes with a standard extensibility policy: foreign namespaces may used to complement SVG data with application-specific content, this extra information being simply ignored by SVG applications. This policy is used effectively in SVG authoring tools such as Inkscape, but also in SVG models exported by automotive interface designers – such as 'ALTIA design' – to complement the graphic model with application-specific content.

The existence of several SVG authoring tools makes the design of new HMI graphic content a simpler task. SVG being an authoritative description for vector graphics, some HMI design tools used in the automotive industry already have a partial support for it. This interoperability is possible the because such tools have an internal model for the description of graphic content that shares the most important structural features of SVG: for example, SGF, the format used in the 'Scade Display' design tool, has a document with a tree-like structure that supports groups, some nested affine transformations, graphic elements that consist in basic and complex shapes as well as text, customization through style attributes, a classic painter rendering model, all features that are present in the SVG model as well. The selection of SVG as a core format therefore has the potential to increases interoperability between generic and application-specific design tools.

Figure 6. Scade Display Editor



Finally, beyond authoring tools, SVG format in this context greatly benefits from dedicated software support, especially from the Batik SVG Toolkit. This library provides a large set of tools for generation, manipulation and transformation of SVG models as well as their extensions, crucial features that can be used as a basis for HMI model designers. It supports dynamic SVG documents – graphic models whose content changes with time – and handles user events, and is therefore also a suitable basis as HMI simulation engine. Being a Java library that can generate AWT/SWING components – and therefore, with the appropriate bridges, SWT components – its integration in the EDONA Eclipse platform is natural.

EDONA HMI format

The EDONA HMI format is made of a subset of SVG constructs – a custom SVG profile – and of a set of application-specific functional extensions.

Profile

The SVG Tiny 1.1 specification, a profile dedicated to highly restricted platforms, is selected as a reference to define our EDONA HMI profile as its basic features cover most of our needs. However, to support some features commonly found in HMI designs, a small number of extra SVG features, not present in SVGT 1.1, are added: they are gradients, opacity and clipping. The extent of this support is the following: gradients and opacity constructs are supported as in the SVG Tiny 1.2 candidate recommendation (or SVG Tiny 1.1 Plus) and clipping as in SVG Basic 1.1. Based on the study of HMI design use cases, this set of constructs is currently a good trade-off between model simplicity (that eases model management, code generation and guarantees a good embedded graphics library support) and the ability to describe a large range of HMI models. However, as the complexity of HMIs and the target platform capabilities evolve quickly, this particular selection of constructs may have to be updated.

We explicitly exclude from the EDONA HMI profile the animation elements of SVGT 1.1, because this animation model not compatible with our HMI component model. To begin with the declarative

constructs the SVG Tiny animation model rely heavily on physical time for animations: to start the animations, control their duration, specify how the animated values change in time, etc. But in the component model outlined above, time measures are generally not available. Moreover, event-based activation of animations in SVG Tiny is based on a fixed set of supported events, while we may update the graphic content as a consequence of an arbitrary external event (for example of an interaction between the user and an audio system turn knob).

The kind of dynamic parameters we describe in the graphic content is related to changes in the graphic transforms, shapes, style or text data, but does not require changes to the structure of the element tree itself. The structure of the document being fixed, a very simple document model may be adopted: SVG data stored in attributes or text content have to be explicitly labeled as writable and/or readable and exposed as inputs and/or outputs in a component interface.

HMI extensions

The lack of support for animations is compensated by the ability to label SVG data, and then expose it as a component input signals or output signals: we explicitly flag the subset of the document data that is eligible for interaction. As the structure of the SVG document cannot be changed, such exposed signals are guaranteed to be always available in the document.

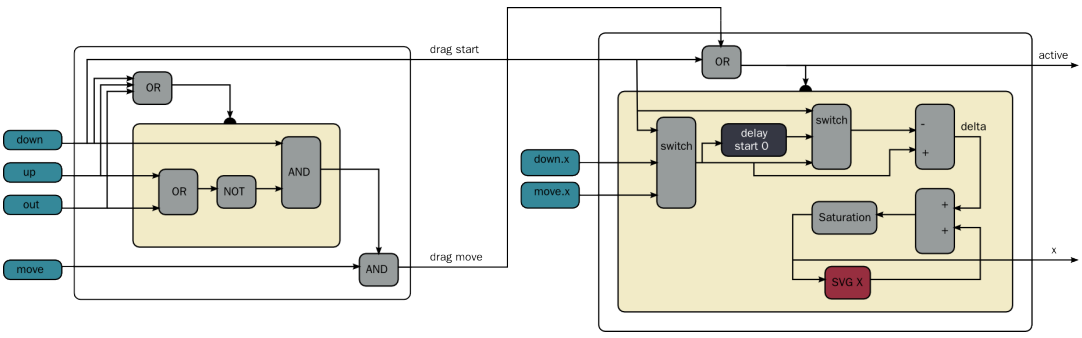
Exposing a labeled data as a component input is done by linking a component interface input to the labeled data. Linking denotes the path network followed by the information flow inside the component.

Group component is the construct that allows to create new components with user-defined functionality and interfaces. Group components are hybrid containers: they may contain graphic elements and micro-functional constructs, including other group components: they therefore provide a hierarchy that is consistent with the graphic structure. Group components encapsulate information: the signals they contain are not available for linking outside of the component unless they have been explicitly exposed into the component interface. Inside components, signals may be linked consistently with these visibility rules and also information flow: information sources to information sinks.

Group components may be triggered: their content won't be activated unless a specific numeric signal – the trigger – differs from zero. The remaining functional constructs are constants – typed values that do not change and are not present in the SVG document) – delays (basic memories that output at a given time the input value at previous instant) and built-in functions – memoryless components that compute their output values based only on input values.

These constructs are simple but sufficient to describe common HMI display and control elements. For example, the simplified micro-functional diagram of a slider control is represented in the figure below. Component inputs, in blue, provide information about the user events that controls the slider activation and movement. These signals are processed by functions and delays (light and dark gray blocks) that perform basic logic and mathematical operations, input selection and data storage. Some signals as used to trigger (black circles) the activation of subsystems while others are exported as component outputs ('active' the slider status, and 'x', its position). The slider position is synchronized with the SVG document through the SVG signal data (red block) that controls the slider position.

Figure 7. Slider micro-functional model



The additional data that we wish to include in the HMI model leverages the 'desc', 'title' and 'metadata' SVG elements that all share the feature of allowing the embedding of non-svg elements namespaces. The 'desc' and 'title' elements are natural targets for component documentations and requirements traceability. Internationalization support requires several annotations in the document: references to an external translation table, something that takes place in metadata elements, as well as specific text id flags that are applicable to SVG text contents as well as functional text constants for identification and localization purposes.

Conclusion

This paper described the effort undertaken by the EDONA HMI project for the modeling of HMI displays in car cockpits. We demonstrated the benefits of using the SVG format in the modeling of the HMI graphics layer and the nature of domain-specific extensions that together are the core model of the EDONA HMI design environment.

EDONA is a 3-year project that has started in september 2007, and although most of the development effort for the HMI design environment is still ahead, the structure and content of the HMI models, a central point for a model-based development environment, has been established. By the end of the project, it it planned to demonstrate the complete HMI design environment with the development of two platforms: the HMI of an advanced prototype of intelligent transportation system on one hand, and a production-ready, certified HMI application on the other hand.

Acknowledgments

The authors acknowledge the contributions of EDONA HMI work group leader Visteon and partners CAOR Mines Paris-Tech, Intempora, Esterel Technologies and Geensys.

This work has been performed in the context of the EDONA project of the System@tic Paris Région Cluster. It is sponsored by the "Direction Générale des Entreprises of the french administration", the "Conseil Régional d' Île de France", the "Conseil Général des Yvelines", the "Conseil Général de l'Essonne and the "Conseil Général des Hauts de Seine".

References

EDONA project: http://www.edona.fr .

SYSTEM@TIC PARIS-REGION cluster, Automotive and Transport working group. http://www.systematic-paris-region.org/fr/automobile .

NUM@TECH AUTOMOTIVE initiative. http://www.numatech-automotive.com .

AUTOSAR: AUTomotive Open System Architecture. http://www.autosar.com .

CAOR, Mines ParisTech. http://caor.ensmp.fr .

Visteon. http://www.visteon.com. See also http://visteon.com/innovate/home.html for the Visteon/3M X-Wave platform.

Intempora, RT-Maps. http://www.intempora.fr/maps .

Esterel Technologies, Scade Display and Scade Suite. http://www-esterel-technologies.com .

Geensys. http://www.geensys.com .

SVG recommandation and mobile profiles. http://www.w3.org/Graphics/SVG/ .

Batik SVG Toolkit. http://xmlgraphics.apache.org/batik .

Altia design. http://www.altia.com/products_design.php .

Inkscape Vector Graphic Editor. http://www.inkscape.org .