Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 4291040, 15 pages
http://dx.doi.org/10.1155/2016/4291040

Research Article

Integration of Data from Vehicular Ad Hoc Networks Using Model-Driven Collaborative Tools

Raquel Lacuesta,[1] Jesús Gallardo,[1] Jaime Lloret,[2] and Guillermo Palacios[3]

[1]*Department of Computing and Systems Engineering, University of Zaragoza, Ciudad Escolar s/n, 44003 Teruel, Spain*
[2]*Instituto de Investigación para la Gestión Integrada de Zonas Costeras, Universidad Politécnica de Valencia, Camino Vera s/n, 46022 Valencia, Spain*
[3]*Department of Electronic Engineering and Communications, University of Zaragoza, Ciudad Escolar s/n, 44003 Teruel, Spain*

Correspondence should be addressed to Jaime Lloret; jlloret@dcom.upv.es

Received 3 December 2015; Accepted 9 February 2016

Academic Editor: Manabu Tsukada

Ubiquitous environments such as Vehicular Ad Hoc Networks need applications that allow them to integrate data and services to build knowledge that can be used to make decisions and to improve standards of living and user safety, among others. We have designed a collaborative virtual environment that covers the needs of integration of knowledge from different vehicles to endow the final user with the necessary information. This environment has been carried out following a model-driven approach that generates a groupware application for improving collaborative work and access to services. The implemented tool facilitates the development and implementation of collaborative frameworks in VANETs, where every vehicle acts as a node.

1. Introduction

The current social context marked by the reduction and wide diffusion of networks and wireless technologies, as well as the reduction in the price of mobile devices, entails that the provision of ubiquitous services that simplify the access to the information in a VANET is a growing need day by day [1].

In the near future, most of the new vehicles will be equipped with short-range radio capable of communicating with other vehicles or with road infrastructure at a distance of at least one kilometer. Radio will allow new applications that are going to revolutionize the driving experience, offering a lot of applications, from instantaneous and updated location of the traffic to warning signs when the vehicle that goes in front stops sharply [2]. In fact, communications wireless networks among vehicles (VANETs or Vehicular Ad Hoc Networks) are being already used at present as a promising technology to improve the safety in the roads. It is also necessary to mention goods transportation that in certain areas is fundamental for supplying consumer goods, being critical in areas where there is neither railway line nor airports. Thus, road networks and highways are an important public service but it entails some problems at the same time. On one hand, there are problems related to the safety, because accidents are in the present day agenda, despite having better safety measures that are reducing the figures. On the other hand, the management of these communication roads becoming more and more efficient faces the increasing number of vehicles that in occasions is excessive relating to the network, resulting in traffic jams and slow traffic.

In a VANET or Vehicular Ad Hoc Network the nodes are vehicles (cars, trucks, buses, etc.) which constitute a network in a continuous movement. Nodes move in arbitrary ways and they can communicate among them (vehicle-vehicle) or can establish a communication with some type of infrastructures. Some main vehicular networks characteristics are as follows:

(i) Autonomy: every terminal is an autonomous node with capacity to process and route the information coming from other nodes of the same network.

(ii) Network distributed control: the control is done in every node since there is no infrastructure performing that.

(iii) Routing: it is necessary that every node provides a dynamic mechanism of routing, separately, and all as a whole. Classical routing protocols are not applicable to this type of networks since they are not prepared for topology variations that VANETs exhibit. Nowadays, routing algorithms are being developed to face this problem.

(iv) Variable network topology: in vehicular networks, nodes or vehicles can move in an arbitrary way, though sometimes they follow some mobility patterns. Due to this, networks can be subdivided, and, consequently, they can experience packet loss. To minimize these effects, mechanisms that detect these circumstances must be developed.

VANETs allow communications among drivers and passengers without any implemented additional infrastructure. Nowadays, ubiquitous computing environments are considered a priority researching line, as they are supposed to be future components of mobile communications networks. Ubiquitous computing in Vehicular Ad Hoc Networks capacity to offer personalized and content-dependent services will create new services and added value applications to directly benefit users [3]. The applications go from a simple exchange of information to the integration of infrastructure with high complexity. The general applications framework takes into account the following items: spreading of warning messages for those vehicles that could find accidents or dangerous situations, to obtain useful information for drivers, such us restaurants, hotels, and service stations, as well as entertainment: Internet, multimedia downloading, or chat among vehicles.

The initial aims of a vehicular network deal with the improvement of safety conditions on the road and the efficiency in daily transportation. The development of intelligent transport systems (ITS) tries to provide every single driver with information relating to the vehicles located in his/her zone of influence [4].

In these systems, the speed of displacement is high, the formation of the networks is very dynamic, and the environment of spreading is extremely variable and could be very large [5]. The presence of a high number of sources of information is another remarkable factor. This, coupled with the fact that transmissions must be trustworthy to give robustness to safety applications, makes the process of definition of a suitable communications system a complex task [6]. The main lines of research promote the use of this technology to offer intelligent transport systems (ITS), whose priority aim deals with the formation of communication networks between vehicles (V2V communication), as well as among vehicles and the supporting infrastructure (V2I communications). They claim providing drivers information relating to vehicles located in their zone of influence, especially of those vehicles which are not within the field of vision. Data can be requested or gathered from the vehicles [7]. Moreover, the information must be routed to the destination using a routing algorithm [8].

Although VANET technologies have prevailed in V2V communications, it is necessary to study if software applications work properly. The design of a collaborative application will allow having a better control of users, optimize the process, and make the way they contribute to the system flexible.

The definition and creation of services in a flexible way making it easy to adapt dynamically the situation to the condition and situation is a necessity in VANETs, as connectivity with any device or infrastructure will be needed in order to warrant safety and performance. VANETs should be able to introduce a set of new services in a robust and cost-efficient way. Nowadays, ubiquitous computation environments are considered as a priority line of research, since they are future components of mobile communications networks. The capacity of ubiquitous computation to offer personalized and context-dependent services allows offering new services and added value applications to directly benefit users and therefore VANETs users. Ubiquitous computing is an idea that is being widely developed [9, 10] and that has concreted in issues such as context-aware computing [11] and ambient intelligence [12]. The development of wireless sensors has led to the concept of ubiquitous sensing, with some interesting applications such as [13], and to the concept of wireless sensor networks, which leads to development such as [14]. From some time now, the usual software has been demonstrated insufficient to cover ubiquitous needs so it became necessary to use other types of software [15, 16], designed for the development of activities in group, normally known as groupware. Also, the use of component-based technologies [17, 18] to define and work services can contribute to improving groupware software as it contributes to solving distribution-related problems. Such use of components enhances the degree of tailorability provided by a system as it allows configuring and replacing components on the fly.

Currently, to develop ubiquitous networks it is necessary to have generic applications that allow them to integrate data and services aimed at improving standards of living and the user safety, among others. We can solve this problem to work on VANETs. It allows covering the needs of communication and group activity in the different tasks carried out at work, as well as in learning activities in leisure moments and even in collaboration tasks to increase security and performance of the networks. Some advantages of the use of collaborative software are that information can be gathered anywhere and instantly transmitted through the network and that the same information can be available to anyone through a device belonging to the network or any other device with access to the Internet.

Groupware development, included in the field of Software Engineering, has gained relevancy in past decades. There are aspects such as synchronization and coordination, awareness, or shared workspaces that did not exist in single-user systems but they appear in ubiquitous environments [19]. Groupware has turned out to be more difficult to design, to implement, and to evaluate than noncollaborative applications, since it is necessary to bear in mind aspects such as social protocols or group activities. To facilitate the task of construction of this type of tools, the use of a complete systematic method is proposed. We try to simulate collaborative team groups

in companies where people work from remote locations to solve a common problem. In this case each node obtains data from its different sensor. With these data it generates its own knowledge that it shares with the rest of the nodes. The adaptation to new network scenarios (e.g., when nodes enter and leave networks and where needed services could change quickly) will be also easier.

In this paper we develop a collaborative environment; it allows an optimal operation of applications. The platform will let nodes collaborate to define new services, to access them, and to obtain a shared knowledge. Each node will work as an employee in a company, working together with others to generate valued knowledge while maximizing the efficiency of collaboration. This approach allows performing joint tasks by means of nodes placed in different geographical locations, in such a way that the resulting delay is almost negligible and the obtained information is very helpful for nodes.

This paper is structured as follows. In Section 2 we review some current research that works with groupware and model-driven collaborative tools. Section 3 presents our system proposal and its application to the definition and creation of software for VANETs. The verification is presented in Section 4. Finally, some conclusions are derived in Section 5.

2. Related Work

There are several groupware tools that allow the construction of models or artifacts in a specific application domain. For example, DomoSim-TPC, which works on the domain of home automation, Co-Lab, which deals with systems dynamics, COLLECT-UML, which is a tool for the UML teaching, and COLER, designed for the teaching of entity-relationship diagrams. In this specific situation, our application domain will be the one of vehicle networks, as it has been previously explained. These systems are introduced and compared in [20].

An important concept that is usually present in groupware tools deals with awareness. This concept refers to the knowledge and perception of the group and its activity [21, 22]. In some situations, awareness can be used for enhancing collaborative opportunities reducing the metacommunicative efforts needed to collaborate across physical distances and in computer-mediated environments. The main idea behind awareness is to keep users working in a collaborative way informed about the actions of other users and even about the next expected action to be carried out by them. In this case, nodes in the vehicle network will broadcast not only the information obtained by every one of them but also the knowledge obtained from the analysis of its information and the information received from the other nodes. This information will refer to the present state of the environment as well as to past states and possible predictions of future states.

As it has been stated before, groupware development is not a trivial activity at all. When developing groupware, different aspects must be taken into account, and even some of them do not belong to the software development field but are typical of other disciplines. With the goal of making this task of development easier, several methodological approaches have been developed. One of them is AMENITIES [23], which is a model-based methodology that uses behavior and task models for the analysis, design, and development of collaborative systems. This approach is based on an ontological definition for the specification of such systems. It is also worth talking about the CIAM (Collaborative Interactive Applications Methodology) proposal [24], which is a proposal for the design and specification of the user interface in collaborative interactive applications. The CIAM includes the CIAN (Collaborative Interactive Applications Notation), which allows the representation in a graphical way of the modeling of each stage in the methodology. Finally, the SCOPE is also an interesting approach. This environment is a system that gives support to the definition of workflows in which the activities are collaborative tasks. It also allows the specification of each task, including the configuration of the session and the input and output artifacts. In this approach, the mechanism for the definition of collaboration protocols is especially important. Other authors center their research on finding the best way to provide the information to the user. In [25], authors propose a method that allows graphical interface designers to incorporate awareness mechanisms in driving simulation environments. They work with collaborative systems and try to design usable interfaces to simulate the reality more adequately.

During the last years, building software, as well as particularly groupware, has become an increasingly complex task. In order to make this task easier, the paradigm Model-Driven Engineering (MDE) [26, 27], also known in the scope of computer science as Model-Driven Software Development (MDSD), is gaining importance. This approach for software development is also being used in the specific field of groupware applications. One of the main goals of MDE is to bridge the conceptual gap between the problem to be solved and its software implementation and to facilitate migration between platforms with the reusability of developed code and to keep developers separate from the complexities of the implementation platform [23].

The MDE paradigm is based on the use of models for the construction of software. A model is a computable representation in which each element of the representation corresponds to an element in the domain. In order to work with models, MDE [28] defines two concepts: domain-specific languages and transformation and generation engines. Domain-specific languages (DSL) are languages designed to represent aspects of a given domain. The concept of DSL is not specific of metamodeling, but it is present in different fields of computer science. In MDSD, these languages are defined by means of metamodels, which are models that define the structure of other models. The concept of meta-metamodel also exists, which is a model that defines the structure of metamodels. As far as transformation and generation engines are concerned, these are any piece of software that, from one or more models, generates other models or source code. Usually, two kinds of transformations are defined: model-to-model (M2M) and model-to-text (M2T) transformations. The former are used to generate models from other models, while the latter are usually designed to generate source code or another kind of textual representation. The MDSD

paradigm is being used by researchers in diverse fields and domains, such as Feature-Oriented Programming [29], GUI (Graphical User Interface) development [30], or analysis of users' activities in collaborative systems [31].

An interesting approach to MDSD is that defined by the Eclipse Foundation in the Eclipse Modeling Project. The project consists of a series of plug-ins that provide support for the different steps in the metamodeling process. The main one is the Eclipse Modeling Framework (EMF), which features technologies for model creation and transformation. This plug-in works with a meta-metamodel called Ecore and it offers the possibility of working with several transformation languages. The plug-in also includes a tree editor which makes it easier to create and edit Ecore-based models, which are stored in the XMI (XML Metadata Interchange) file format. In the specific field of tools for developing models or diagrams, EMF is usually integrated with another plug-in in the Eclipse Modeling Project called Graphical Modeling Framework (GMF). GMF allows the creation of graphical modeling tools based on Ecore metamodels. To develop a graphical editor with GMF, four models are needed: the Ecore model, which is the metamodel defining the application domain; the model with the graphical elements; the model that defines the toolbar; and a fourth model, containing the mapping of the others. Once these four models are created, the GMF runtime environment is able to create the source code needed to implement a tool for the graphical creation of models. This makes GMF not very easy to use for nonexperienced designers. Of course, graphical editors developed by GMF are single-user, so there is no way of doing synchronous collaborative editing with them.

Collaborative systems in VANET have been proposed in order to avoid collisions [32] and intrusion detection [33] or even to carry out collaborative positioning in advanced driver assistance system [34] and to download data in a collaborative way [35]. However, not much has been done regarding use of model-driven collaborative tools. A study center in model-driven collaborative tools is the one carried out in [36] where authors carry out a model-driven approach to design a secure routing protocol for UAV Ad hoc networks.

Our proposal introduces model-driven elements in systems for VANETs in order to develop a collaborative system that adapts itself and allows having the necessary strategies to provide the available services. Therefore, our method to develop collaborative systems has been instantiated for generating synchronous distributed collaborative modeling tools in vehicular networks. The interest in using our method and not another one resides, on the one hand, in the integration of information with the related awareness support, so that users are always provided with all the information of interest. And, on the other hand, the model-driven approach itself is always a strong point of the proposal, as we are allowing nonexperienced users to work with it.

3. System Proposal

In this section, we introduce the system model proposed for vehicular environments. We explain the communication technology used and the model-driven approach used in the collaborative system development.

3.1. Vehicular Environments. When the amalgam of services (both in the area of the safety and in the field of generic provision of information) increases, it is necessary to investigate in technological solutions that replace the requirements of a sufficiently generic and flexible infrastructure of services.

The fulfilled research has a strong crosscutting character. Problems in the different levels of development of collaborative applications for VANET environments are taken into account, focusing on the integration of data and on the provision of ubiquitous services. The final goal deals with the integration of different solutions in order to define a prototype over a few scenarios of interest as well as a possible application to VANET environments. In our system numerous devices in vehicles interact among them and with user in a transparent way as a "ubiquitous computation model."

A collaborative system that will allow analyzing and visualizing the environment has been modeled. Thanks to the flexibility of the model, this system could be developed to detect situations of danger and/or emergency, generating notices to the driver when the situation is critical and interacting in addition with the supporting infrastructure provided that it can give useful security and/or energetic information. Furthermore, the vigilance of the vehicle will allow diagnosing both vehicle consumption and wear. For the design of these systems factors such as the speed of movement are not considered. Our system will not center on a special attention to the maintenance of an interface of continuous communication and without fissures, nor on the changes or handoff that could take place among different network technologies. The received information from different nodes will allow constructing the collaborative system, not being necessary the total reception of the data from a node, since there are duplicities of both information and flow of information. The distribution of information all along the network nodes (vehicles or infrastructure points) will allow the fact that the loss of information due to the speed of these ones and their dynamic should not have repercussions in the quality of the showed information by the collaborative system. The collaborative application will allow the inclusion of new received information, as well as that of its maintenance or elimination in the update process. The possible communications technologies that can be used are explained in the following points.

The application model will allow the fact that the user or driver (and even passengers) benefits with a transparent access to a set of services oriented not only to the driving and safety improvement but to the reception of information about interest points (leisure and others).

To model the system two levels of collaboration are established:

(i) Level 1: vehicle sensors, establishment of collaboration to deduce knowledge.

(ii) Level 2: vehicles and infrastructure systems, exchange of information to obtain knowledge.

The tests are being carried out using simulation using Wi-Fi connections.

Table 1: Comparison between the method used and two different approaches for the modeling of collaborative systems.

	Method used	CIAM	AMENITIES
Kind of systems it supports	Collaborative modeling systems	Collaborative systems	Collaborative systems
Notations used	EMF, a DSL developed with GMF	CIAN notation, which integrates CTT (ConcurTaskTrees) and others	Ontologies, COMO-UML
Support tools used for modeling	Eclipse with certain plug-ins	Partial	COMO-TOOL
Support for the definition of requirements	No	No	Yes
Support for the definition of the application domain	Yes	No	Partial
Support for the definition of users, sessions, and so forth	Work in progress	Yes	Yes
Support for the definition of tasks	Work in progress	Yes	Yes
Support for the configuration of workspaces	Yes	No	No
Support for the production of the groupware system	Yes	Partial	No

3.2. Communication. Although many different communications technologies have been used in vehicular environments [37, 38] some of the most relevant ones in VANETs are going to be presented. Any of the selected technologies will allow the development of services with our collaborative platform among the different vehicles. Several wireless access methods in vehicular environments are overviewed, namely, DSRC/WAVE, Cellular method, Wi-Fi, WiMAX, and so forth. As it has been previously introduced, to carry out the simulations a Wi-Fi connection has been used. Wi-Fi or WLAN can also support broadband wireless services. Among the group of standards established by IEEE 802.11 we use the IEEE 802.11b. IEEE 802.11b standard does not arise specifically for standardization of vehicular networks, but it is used for researching purposes. Some of its characteristics are link level protocol, CSMA/CA as a medium access control, and Direct Sequence Spread Spectrum (DSSS) modulation and it works in the 2.4 GHz band.

3.3. Model-Driven Development of Collaborative Systems. The software support of the platform introduced in this work has been developed as a collaborative modeling system that has followed a model-driven approach in its implementation. The model-driven method has been used to develop that collaborative system.

3.3.1. Collaborative Systems for Synchronous Modeling. In order to carry out the software support of our approach, a series of strategies for the modeling of collaborative networks have been considered, focusing our efforts on the achievement of an integrated environment. Specifically, we have focused on the use of collaborative modeling synchronous tools. This kind of tools is usually used by several designers that build a model or diagram in a collaborative and synchronous way. That is to say, they work at the same time over a specific model or diagram. In our case, instead of designers, the nodes may be mobile phones, sensors, vehicles, infrastructure nodes, or others. But these nodes, which will make up our ad hoc networks, will also work exchanging messages in a collaborative way in order to have a representation of the whole node network in each moment, so that the representation is useful for the users in the network.

3.3.2. Application of a Model-Driven Development Method for Collaborative Modeling Tools. In the related works, some methodological approaches for the development of collaborative systems were analyzed. However, these approaches lack some interesting features for the development of collaborative systems. This fact has led us to the use of a different approach for the development of the collaborative software in our proposal. This method [20] was conceived as a way of generating synchronous distributed collaborative modeling tools that may be adapted to any kind of application domain. This is a model-driven method based on the Eclipse framework and the aforementioned Eclipse modeling plug-ins. Every element related to the configuration of the generated tools is defined by means of models according to a series of previously defined metamodels. As previously mentioned, tools generated by the GMF plug-in are single-user tools, and the definition of application domains is difficult for less experienced users since it involves the creation of several models. Thus, one of the main advantages of this development method lies in its offer of integration of collaborative functionality in GMF-based tools along with a simpler way to define application domains by working with just one application domain model.

To validate the method a comparative study taking into account other approaches of collaborative system modeling has been done. Table 1 depicts a comparative analysis with this method and two different approaches from the ones mentioned in Section 2. Issues relating to notations and modeling languages have been taken into account, since we are dealing with approaches that do not seek the same goals that the proposed development method does.

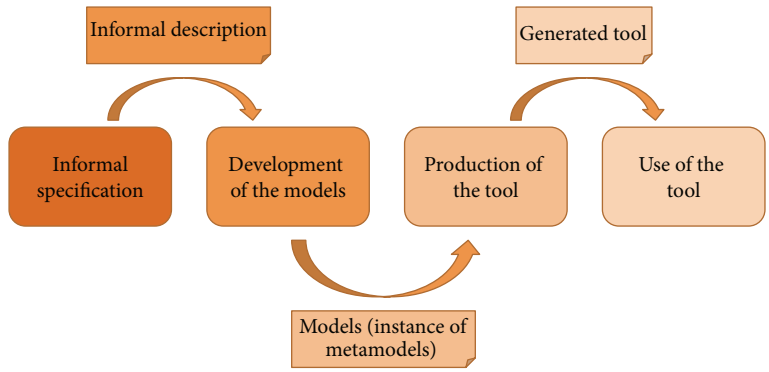

Figure 1: Steps in the methodological framework of the method.

Through the analysis of this comparative study, some interesting conclusions can be drawn. To begin with, our method is the only one that has been conceived for a specific type of collaborative systems, such as the modeling collaborative systems. This makes our method less generic, but it allows dealing with specific problems of these systems, such as difficulties at the moment of developing the systems from the very beginning or the way in which the application domains are defined. Regarding notations and tools that every methodological approach uses, our method has the following advantage: it treats with notations and tools that are widely used in both the scientific area and the industry (Eclipse technology), whereas other approaches either have no notations or associate tools or use a new and not well-known tool by the majority of users.

Our model-driven development method is based on three frameworks. These are (i) the methodological framework, which consists of a series of phases that must be followed by any user who wishes to develop a collaborative modeling tool, together with the roles that are involved in each phase, the used tools, and the obtained products; (ii) the conceptual framework, which is made up of the models that are used in the metamodeling process in order to conceptualize and express the application domain and workspace issues of the tool to be generated; and (iii) the technological framework, which consists of a series of extensions to the Eclipse metamodeling plug-ins platform and the transformation processes used to generate the models needed in GMF. The steps in the methodological framework are informal specification, development of the models, production of the tool, and use of the tool, as shown in Figure 1.

Specifically, the work carried out in each phase is the following:

(i) Informal specification: the first thing to do when applying our development method to a given domain is to identify and isolate the application domain. This step produces an informal description of the domain that can be made up by text in natural language, drawings, sketches, and so forth. Any of the usual techniques for requirement analysis in Software Engineering can be used in this phase.

(ii) Development of the models: the metamodels that are needed for the subsequent generation of the tool have to be developed. On one hand, the application domain has to be modeled, and, on the other hand, the structure of the workspaces should be defined as well. The models generated are instances of the metamodels that make up the conceptual framework of the development method. To generate these models, authoring tools included in the technological framework of the method can be used.

(iii) Production of the collaborative modeling tool: once the application domain and the workspaces definition have been formalized, some automatic steps take place resulting in the generation of the collaborative modeling tool. This automatic steps include mainly some M2T (model-to-text) transformations that make it possible to obtain source code from the models developed in the second step. This is the phase that actually implements the model-driven features of the development method.

(iv) Use of the collaborative modeling tool: once the modeling tool has been generated, designers can build models belonging to the application domain, working collaboratively in organized groups. The infrastructure included in the technological framework of the method allows the collaborative modeling tools obtained in the previous phase to include the code that is necessary for the creation of sessions, the management of sessions and users and all the connection, coordination, and communication issues.

The UML sequence diagram in Figure 2 depicts how data is read and sent over the network when a change is detected by any sensor in a node in the network. The process follows a sort of Model-View-Controller pattern. The process starts when the sensor sends a piece of data to its associated listener. If the change is significant enough, the listener proceeds to begin with the update of the model and the tool. Thus, the model itself is updated, and the listener sends a message to the graphical tool so that it reads the new version of the model. This is done, and the graphical tool updates its view.

```
protected void createChannel() throws ECFException {
  IChannelContainerAdapter channelContainer = (IChannelContainerAdapter)
  container.getAdapter(IChannelContainerAdapter.class);
  final ID channelID = IDFactory.getDefault().createID
  (channelContainer.getChannelNamespace(),nombreCanalLocal);
  final IChannelListener channelListener = new IChannelListener() {
    @Override
    public void handleChannelEvent(IChannelEvent event) {
      if (event instanceof IChannelMessageEvent) {
            IChannelMessageEvent msg = (IChannelMessageEvent) event;
            aplicObservador.recibirDatos(msg.getData());
      }
    }
  };
  channel = channelContainer.createChannel(channelID,channelListener, new HashMap());
}
```

Algorithm 1: Creation of a channel.

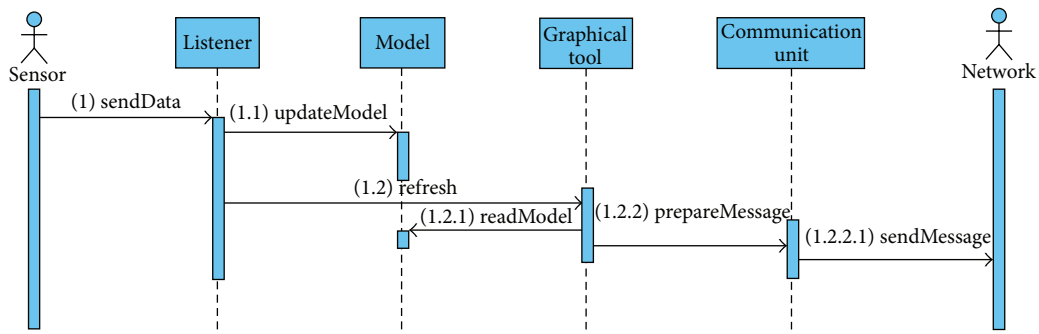

Figure 2: UML diagram: data read and sent.

The last step is to send the message that will replicate these changes over the network. When the remaining nodes in the scope of the network in that moment receive the message, a similar update will be done so the relevant data is available in such nodes.

Algorithm 1 includes an example of source code from the tool that supports our system. Specifically, this method is the one in which a channel is created in the space-clipse.ecf.SpaceClient class. The channel is used to support the exchange of information inside a given network. When the channel is created, a method that handles the information received is created. When this information reaches this method, the adequate processing takes place. The specific implementation details of the method are due to the election of ECF (Eclipse Collaboration Framework) as a framework for the communication among the nodes in the network. Thus, this implies the use of concepts such as ChannelContainer or ChannelID that are specific of ECF.

As it has been already introduced, the tool that supports the system has been developed following a model-driven development method for collaborative tools. The development of such method started with an ontological definition of the concepts that are present in the kind of tools that were going to be developed by following the method [39]. In this sense, a special attention was given to awareness, which is a concept that is usually present in collaborative systems and tools.

The XML code in Algorithm 2 is an excerpt of the domain metamodel included in the conceptual framework of the model-driven development method in which the system is based. This metamodel is Ecore model that defines the structure of the data to be handled by the tool that supports our method. In the algorithm, some elements that have to be later instantiated for each use of the method for a specific case are included: for example, operators (entities) and relationships. Later, models developed following this metamodel are able to undergo some M2M transformations so that all the models needed by the GMF M2T transformation engine are generated in an automatic way. This differs in an important way from the traditional way of using GMF, in which several complex models had to be developed from scratch. The model developed is stored in XML-based file. Thus, this model and the others that have been developed suppose the semantic definition of the tool, in the form of some semantic models.

In another excerpt of XML-based file, showed in Algorithm 3, some elements of the mapping model generated

```
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="Space"
    nsURI="Space" nsPrefix="Space">
  <eClassifiers xsi:type="ecore:EClass" name="Root">
    <eStructuralFeatures xsi:type="ecore:EReference" name="operators" lowerBound="1"
        eType="#//Operators" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="relationships" lowerBound="1"
        eType="#//Relationships" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="graphics" lowerBound="1"
        eType="#//Graphics" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
        eType="ecore:EdataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="operator_areas" upperBound="-1"
        eType="#//Area" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="operator_types" upperBound="-1"
        eType="#//Type" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Operators">
    <eStructuralFeatures xsi:type="ecore:EReference" name="operator" upperBound="-1"
        eType="#//Operator" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Relationships">
    <eStructuralFeatures xsi:type="ecore:EReference" name="relationship" upperBound="-1"
        eType="#//Relationship" containment="true"/>
  </eClassifiers>
```

Algorithm 2: Excerpt of the application domain metamodel.

```
<gmfmap:Mapping xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:gmfmap="http://www.eclipse.org/gmf/2008/mappings"
    xmlns:gmftool="http://www.eclipse.org/gmf/2005/ToolDefinition">
  <nodes>
      <containmentFeature
          href="spacemm.ecore#//Root/Vehicle"/>
      <ownedChild>
        <domainMetaElement
            href="spacemm.ecore#//Vehicle"/>
        <labelMappings
            xsi:type="gmfmap:FeatureLabelMapping">
          <diagramLabel
              href="spacemm.gmfgraph#VehicleName"/>
          <features
              href="spacemm.ecore#//Vehicle/label"/>
        </labelMappings>
        <tool
            xsi:type="gmftool:CreationTool"
            href="spacemm.gmftool#//@palette/@tools.0/@tools.0"/>
        <diagramNode
            href="spacemm.gmfgraph#Vehicle"/>
      </ownedChild>
  </nodes>
```

Algorithm 3: Excerpt of a mapping model.

in an automatic way from the domain model that instantiated the meta-model in Algorithm 2 are included. Starting from that instantiated model, four different models are generated: the Ecore metamodel, the graphical elements model, the toolbar model, and this one, the mapping model. Thus, the algorithm shows how the method has been instantiated for the specific case of vehicular networks, and each node in the diagrams is a vehicle. This model is the mapping model, so it is responsible for relating the graphical and semantic properties of each element from other models. Thus, elements from the Ecore metamodel and graphical definition and tool definition models are related in this specific node, as it can be seen in the algorithm.

The aforementioned development method has been used in order to develop our collaborative software. In further sections of the paper how it has been applied to our specific situation and how a collaborative tool that could be used to monitor our vehicle networks has been modeled will be detailed. The main parameters to be taken into account for using the model have been introduced. A first analysis of vehicular environment has been also developed.

3.4. Application of Our Model to VANETs. For context modeling, the technologies explained previously are used. These technologies will let us adapt and abstract the network user, allowing having the necessary strategies to provide available services with ubiquity not only for drivers but for passengers [40]. The model includes sensor and communication variables shown in Section 3. The vehicular environment and the information that the vehicle receives need a more complete model (both of the driving place and the user), aimed at offering context-based services. The strategy of subscription has a special interest in a generic system of services provision. Although typical Internet services such as web browsing, management of transactions, and reservations use a client/server model, where the server replies to a previous client request, the requirements of operation of vehicular services answer to another communication strategy much more suitable (well-known as publish/subscribe system). By using this methodology, the user would not have to do an explicit request to the system he/she wants to receive information from, nor would he be continuously receiving information he/she is not interested in, as it happens in RDS, for example. In a publish/subscribe system, the user shows his/her interest in receiving certain information, and the system is in charge of sending the notifications of interest [41, 42]. A publish/subscribe system allows perfectly working not only in the contextualization of a vehicular environment, but in the modeling of user requirements.

The developed platform analyzes the data coming from different sensors, generates knowledge, and sends this knowledge to the rest of the nodes of the network. The visual interface is updated using the incoming knowledge from the network.

The design has been carried out using collaborative tools. The generation of modeling tools is done by means of messages generated in every node of the network. The graphical interface is updated by the knowledge coming from different nodes. That is, the updating is dictated by messages generated by the tool with the information obtained from sensors and that will be processed before sending. Knowledge will be automatically sent to other nodes taking part in the network in that moment. If any node is providing Internet connection, the data could be also accessed by Internet.

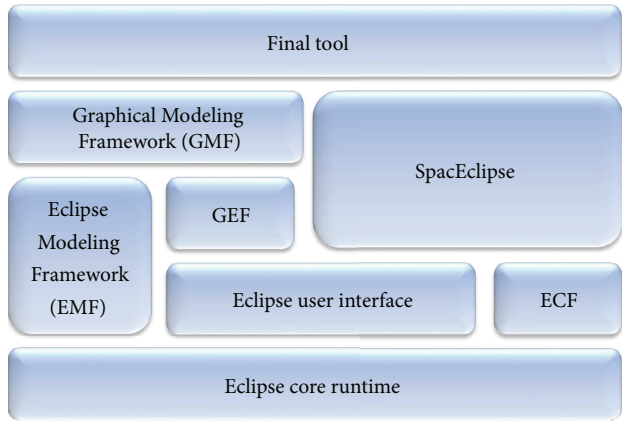

Figure 3: Diagram showing the architecture of the generated tool.

The architecture of the generated tool appears in the block diagram of Figure 3. Due to the fact that the tool is based on Eclipse, all its components will depend ultimately on the Eclipse core runtime. The main components used by the tool are GMF and SpacEclipse. GMF is the plug-in of Eclipse graphic metamodel and depends on both EMF for the aspects of metamodel and GEF for the implementation of the graphical tool. SpacEclipse is the package of communications generated for the method of development based on Gallardo et al. [20] and it uses ECF (Eclipse Communications Framework) to implement the aspects of communication among the different nodes that work with the application.

Figure 4 depicts the metamodel that has been developed as a conceptualization of the elements that are going to take part in our vehicular networks and that has served as a good starting point for the development of the application of monitoring based on models (the core of our proposal). That is to say, the developed metamodel has a double purpose: on the one hand, to organize the knowledge and the concepts that are handled in our proposal and, on the other hand, to use it in the automatic development of the software tool. Figure 4 reflects the metamodel with Ecore notation, developed using the support provided by the metamodel plug-ins of the Eclipse environment.

In the VANET environment each vehicle will act as a node. Firstly, the obtained data are processed; secondly the obtained knowledge is transmitted. According to this data the car will appear on the visual environment with a specific color for its situation (blue: the car is stopped, red: the car does not work properly, green: the car works correctly, and grey: the car has some problems that however let driver drive it). The model has been designed following the way of VANET services development for general purpose computers [43–45]. In our case the data will work collaboratively using the development application.

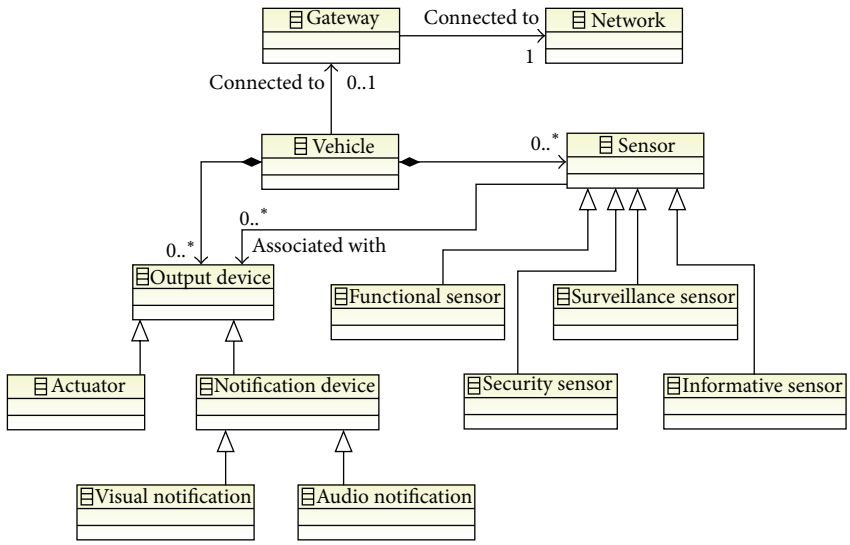

Figure 4: Metamodel for the conceptualization of the elements in vehicular networks.

The concepts taking part in the metamodel are those that represent elements that, in one way or another, are involved in the system or must be represented by it. Specifically, they are the following:

(i) Vehicle: It is the representation of all vehicles, regardless of their type, included in the network. Every vehicle includes a set of sensors and other different output devices.

(ii) Sensors: They are used to gather and monitor different information of usefulness. Four categories are distinguished: functional sensors, security sensors, surveillance sensors, and informative sensors. Every output will be associated with a specific output device, since the collected information may be used for different purposes. For example, the information provided by a GPS would be used at an informative level (device of notification), but the proximity sensor used upon parking might have a direct influence on the braking system (actuator).

(iii) Output device: The vehicle can integrate different output devices, which in general would be classified into two types: actuators and notification devices. Actuators are devices that have the possibility of acting directly on some of the vehicle systems, in the same way that the proximity sensor is sending a signal so as an actuator could activate brakes in case of a very close approaching upon parking. On the other hand, warning devices would be those elements whose sole usefulness deals with reporting the results obtained by sensors, without doing any action on the vehicle mechanisms. The created application, by means of the model-oriented development put in practice, is itself a way of warning device. The metamodel also provides the presence of notification devices, both visual (screens) or acoustic (beeps).

(iv) Gateway: In the proposed architecture, vehicles communicate to each other by means of gateways, located in such a way that they provide services to the whole physical zone in which the network and system will be implemented. These gateways would also serve for the output communication towards the network.

(v) Network: This concept represents the communications network. This network allows the interconnection of different users from different gateways giving support to the whole vehicular network.

The XML code in Algorithm 4 is an excerpt of the Ecore metamodel that defines the structure of the data to be handled by the tool that supports our method, instantiated for the specific case of vehicular network. The Ecore metamodel is stored in an XML-based file. Thus, this model and the others that we have developed suppose the semantic definition of the tool, in the form of some semantic models. In this metamodel, the definitions of some EClasses are included. EClasses are the main concept in Ecore metamodels, so every element in our system may be modeled as one of these EClasses. Subsequently, these EClasses can be related among them and also hierarchical relationships may be established. The automatic generation code process that GMF implements includes the generation of source code from these kinds of models, so that the application is generated in a more or less easy way. One of the main advantages of such approach is the easy way to instantiate the system for a different situation. For instance, the example we are dealing with is about vehicular networks, but a different kind of network could be managed just by developing a series of models that later take part in a process of automatic source code generation.

The tool generated according to our proposal has the appearance of the prototype shown in Figure 5. A portion of a road map can be observed. In it, a vehicle depicts the vehicle in which we are, together with a small panel showing

```
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="VehicularNetwork"
    nsURI="VehicularNetwork" nsPrefix="VehicularNetwork">
  <eClassifiers xsi:type="ecore:EClass" name="Vehicles"/>
  <eClassifiers xsi:type="ecore:EClass" name="Variables"/>
  <eClassifiers xsi:type="ecore:EClass" name="Vehicle" eSuperTypes="//Vehicles">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" lowerBound="1">
        <eType xsi:type="ecore:EDataType" href="http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eStructuralFeatures>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Speed" eSuperTypes="//Variables">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" lowerBound="1">
        <eType xsi:type="ecore:EDataType" href="http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eStructuralFeatures>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Rain" eSuperTypes="//Variables">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" lowerBound="1">
        <eType xsi:type="ecore:EDataType" href="http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    </eStructuralFeatures>
  </eClassifiers>
```

Algorithm 4: XML diagram: data structure.

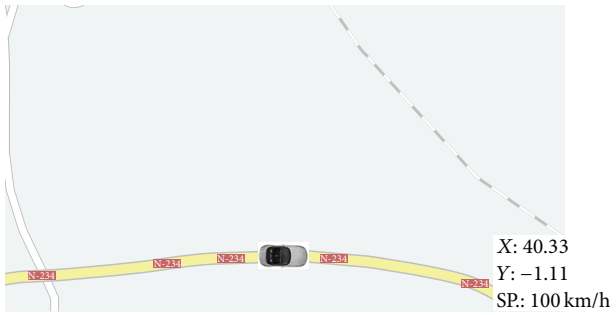

Figure 5: A possible scenario in a vehicular network.

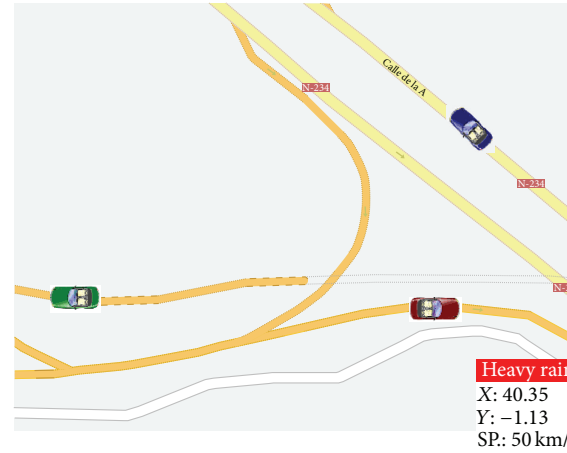

Figure 6: Another possible scenario in a vehicular network.

the current values for X and Y coordinates and the speed relating to our vehicle.

In this scenario it can be seen that the car is running but has some problems of working. All along the route covered for the vehicle, values of variables will be updated according to the data gathered by the vehicle sensors and the information that comes from other nodes belonging to the established network. As well as the road map and the situation of the vehicles in it, gathered values by the system will be used as inputs to a few calls to the Google Maps static API, which will generate the corresponding output.

Figure 6 shows a situation in which several vehicles appear, in such a way that the information about the location and the direction of each one of them will have been received by means of the exchanged messages in the network. There can be seen one car stopped (with the danger it implies), one car that is failing (other cars should take care of it), and another that works properly.

The collaborative application allows designing an integrated system provided with enough capacity to analyze the car state. Also environment and situations of danger and/or emergency quickly can be analyzed. The system could also generate warnings for the driver when the situation is critical, while interacting with the supporting infrastructure provided that it can give useful information.

4. System Validation

In order to validate our system, we have performed several tests. These tests have been carried out in an Intel Core i5, 2.67 GHz, with 4 GB of RAM and 64-bit Windows 7. One client was executed using another Intel Core i5, 2.67 GHz,

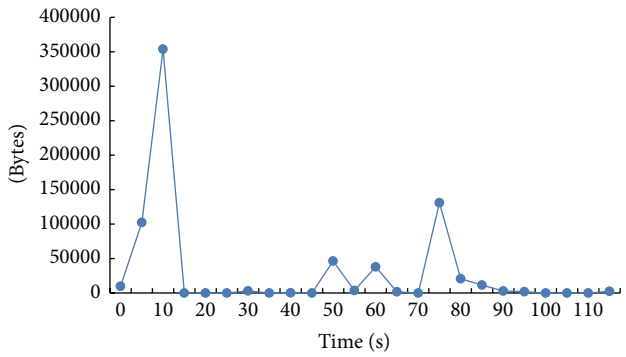

Figure 7: Bytes received when the software is running in Case 1.

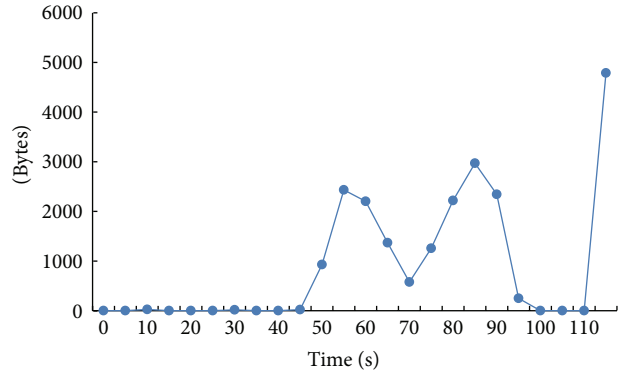

Figure 8: Output bytes written when the software is running in Case 1.

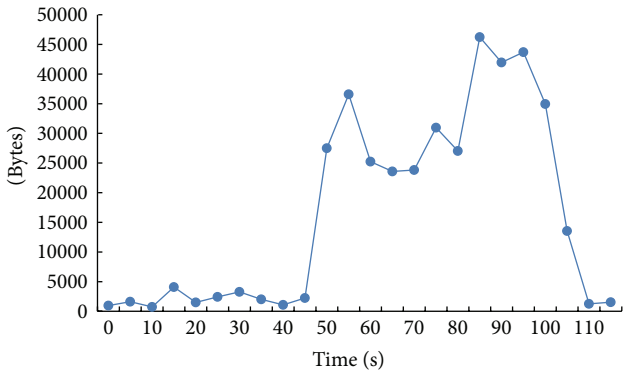

Figure 9: Input bytes read when the software is running in Case 1.

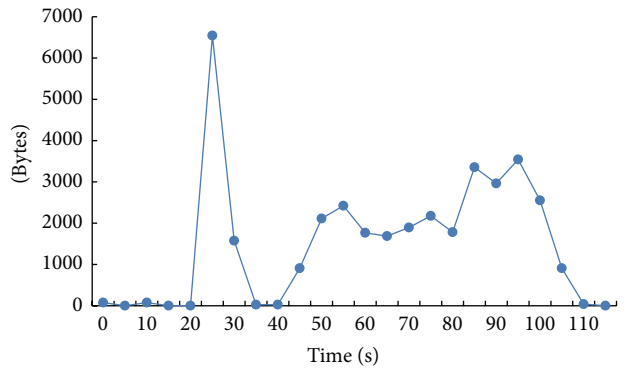

Figure 10: Bytes sent when the software is running in Case 1.

with 4 GB of RAM and 64-bit Windows 7 and the other clients were launched on an Intel Core i3 3.10 GHz with 2 GB of RAM and Windows 7.

We test the performance of our system in two cases. In Case 1, two clients accessed the system. We can observe in Figures 7, 8, 9, and 10 how, in the first moments, initial information is reading and the associated data is sent to the clients. Later, clients start sending information, and that means an increase in network flow and writing to disk. The metrics we have measured are bytes received by the system, output bytes written by the system, and input bytes read by the system.

In Figure 7 we can observe that the maximum value of data received along all the process is 353.8 Kbytes. During some time, no data are received. This is due to the fact that the client is generating the information and does not receive any data, but it generates data and sends it to the other nodes in the network, as it will be seen in the next figures. The system received an average of 30 Kbytes per second.

In Figure 8 we can observe that the maximum value of output bytes written along all the process is 4.7 Kbytes. During some periods of time, no data are written. This is related to the work carried out, as explained when talking about Figure 7. Later, data is written when the client starts receiving information that should be stored. On average, the system is writing 0.89 Kbytes per second.

Figure 9 depicts the input bytes read along the test. The maximum value of input bytes read along all the process is 46.23 Kbytes, the minimum value is 0.74 Kbytes, and the system input is 16.57 Kbytes per second on average.

In Figure 10 we can observe that the maximum value of bytes sent along all the process is 6.54 Kbytes and no bytes are sent along some periods of time. On average, the system is sending 1.51 Kbytes.

In the second case there are three clients connected, although last connection is carried out in the middle of the test. All processes are run in a similar way. However, clients' notifications arrive earlier than those in the other connection. There is a moment when no interactions occur and finally the last client arrives, so the flow begins again (Figures 11, 12, 13, and 14).

In Figure 11 we can observe that the maximum value of data received along all the process is 352.3 Kbytes and no data are received along some periods of time. On average, the system is supporting a reception of 24.76 Kbytes.

In Figure 12 we can observe that the maximum value of output bytes written along all the process is 4.2 Kbytes and no data are written along some periods of time. On average, the system is writing 1.21 Kbytes.

Figure 13 depicts the input bytes read along the test. The maximum value of input bytes read along all the process is 80.73 Kbytes, the minimum value is 0.63 Kbytes, and the system input is 22.65 Kbytes on average.

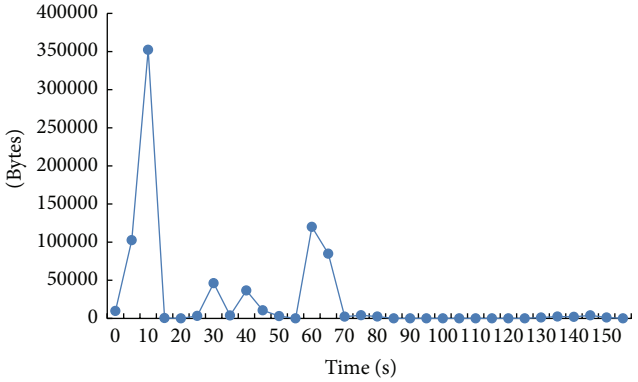

Figure 11: Bytes received when the software is running in Case 2.

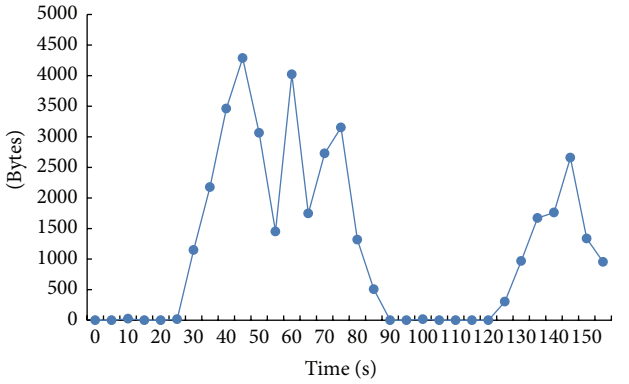

Figure 12: Output bytes written when the software is running in Case 2.

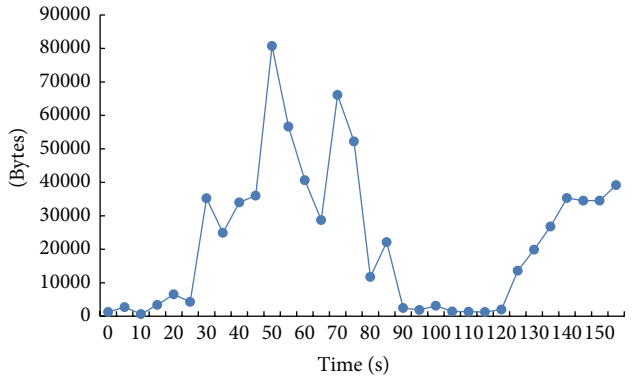

Figure 13: Input bytes read when the software is running in Case 2.

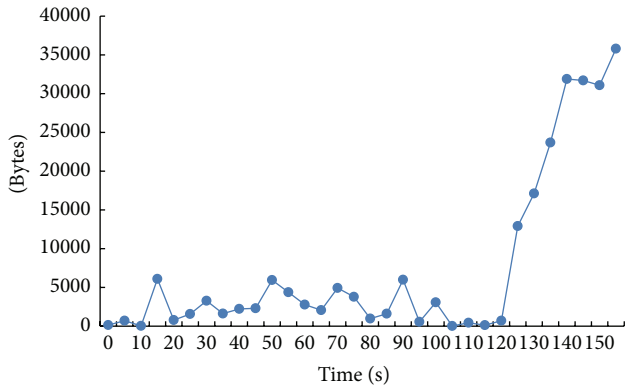

Figure 14: Sent bytes when the software is running in Case 2.

In Figure 14 we can observe that the maximum value of sent bytes along all the process is 35.79 Kbytes, the minimum value is 0.02 Kbytes, and the average of data sent is 0.75 Kbytes.

Taking all these results into account, we understand that the values obtained are acceptable, so the system and the approach that has generated it can be seen as validated.

5. Conclusion and Future Work

The contributions of this development suppose an advance in the state of the art of information and communication technologies (ICTs) in Vehicular Ad Hoc Networks, allowing the development and implementation of software for these scenarios in collaborative contexts.

A new approach for the integration of vehicular data provided by several nodes in the Vehicular Ad Hoc Network and the collaboration among them has been presented. Nodes in these networks have significantly different characteristics and demands from those belonging to traditional wireless ad hoc networks deployed in environments without any infrastructure. The application that implements collaboration and displays the results has been developed following a model-driven approach that optimizes the process and makes the system more flexible. We have generated a groupware tool for vehicular collaborative work. The application of the approach to vehicular environments services allows us to observe an example of application with satisfactory results. We have shown the performance of the developed tool in terms of bytes received by the system, output bytes written by the system, and input bytes read by the system.

As future work, we plan to carry out further studies with the application implying the connection of more nodes in real scenarios. Also, we are planning to improve the application and its associated process by enriching the models in the conceptual framework and by adding more elements for awareness support. Moreover, we are planning to add an artificial neuronal network to predict movements and events.

Competing Interests

Authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] J. Wan, D. Zhang, S. Zhao, L. Yang, and J. Lloret, "Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 106–113, 2014.

[2] T. Yan and G. Wang, "Collecting vehicle trajectory through message dissemination," *Ad Hoc and Sensor Wireless Networks*, vol. 29, no. 1–4, pp. 153–176, 2015.

[3] L. Mansour and S. Moussaoui, "A new framework for request-driven data harvesting in vehicular sensor networks," *Network Protocols and Algorithms*, vol. 5, no. 4, pp. 1–18, 2013.

[4] K. Z. Ghafoor, J. Lloret, K. Abu Bakar, A. S. Sadiq, and S. A. Ben Mussa, "Beaconing approaches in vehicular ad hoc networks: a survey," *Wireless Personal Communications*, vol. 73, no. 3, pp. 885–912, 2013.

[5] K. Z. Ghafoor, M. A. Mohammed, J. Lloret, K. Abu Bakar, and Z. M. Zainuddin, "Routing protocols in vehicular ad hoc networks: survey and research challenges," *Network Protocols and Algorithms*, vol. 5, no. 4, pp. 39–83, 2013.

[6] Y. M. Khamayseh, M. B. Yassein, M. A. Alghani, and C. X. Mavromoustakis, "Network size estimation in VANETs," *Network Protocols and Algorithms*, vol. 5, no. 3, pp. 136–152, 2013.

[7] H. Yang, L. Huang, and H. Xu, "Distributed compressed sensing in vehicular ad-hoc network," *Ad Hoc and Sensor Wireless Networks*, vol. 25, no. 1-2, pp. 121–145, 2015.

[8] K. Z. Ghafoor, K. Abu Bakar, J. Lloret, R. H. Khokhar, and K. C. Lee, "Intelligent beaconless geographical forwarding for urban vehicular environments," *Wireless Networks*, vol. 19, no. 3, pp. 345–362, 2013.

[9] F. Salim and U. Haque, "Urban computing in the wild: A survey on large scale participation and citizen engagement with ubiquitous computing, cyber physical systems, and Internet of Things," *International Journal of Human-Computer Studies*, vol. 81, pp. 31–48, 2015.

[10] M. R. Martínez-Torres, M. C. Díaz-Fernández, S. L. Toral, and F. Barrero, "The moderating role of prior experience in technological acceptance models for ubiquitous computing services in urban environments," *Technological Forecasting and Social Change*, vol. 91, pp. 146–160, 2015.

[11] R. Sriram, S. Geetha, J. Madhusudanan, P. Iyappan, V. P. Venkatesan, and M. Ganesan, "A study on context-aware computing framework in pervasive healthcare," in *Proceedings of the International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET '15)*, Unnao, India, March 2015.

[12] C. Roda, A. Rodríguez, V. López-Jaquero, P. González, and E. Navarro, "A multi-agent system in ambient intelligence for the physical rehabilitation of older people," in *Trends in Practical Applications of Agents, Multi-Agent Systems and Sustainability*, vol. 372 of *Advances in Intelligent Systems and Computing*, pp. 113–123, 2015.

[13] R. V. Aroca, A. F. Burlamaqui, and L. M. G. Gonçalves, "Method for reading sensors and controlling actuators using audio interfaces of mobile devices," *Sensors*, vol. 12, no. 2, pp. 1572–1593, 2012.

[14] A. N. Campos, E. L. Souza, F. G. Nakamura, E. F. Nakamura, and J. J. P. C. Rodrigues, "On the impact of localization and density control algorithms in target tracking applications for wireless sensor networks," *Sensors*, vol. 12, no. 6, pp. 6930–6952, 2012.

[15] E. de Jesus Alcocer Polo, "Groupware para el diseño de diagramas UML en tiempo real dentro de un ambiente WEB," *Tecnología Investigación y Academia*, vol. 2, no. 2, pp. 18–32, 2014.

[16] M. Sosa, I. Velázquez, C. Silva, I. Maldonado, and F. Rosenzvaig, "Interfaz de usuario para Groupware educativos," in *XVI Workshop de Investigadores en Ciencias de la Computación (WICC '14)*, Ushuaia, Argentina, May 2014.

[17] P. M. Vera, "Component based model driven development: an approach for creating mobile web applications from design models," *International Journal of Information Technologies and Systems Approach*, vol. 8, no. 2, pp. 80–100, 2015.

[18] C. Pons, G. Pérez, C. Neil, R. S. Giandini, and M. De Vincenzi, "Ingeniería de software dirigida por modelos aplicada a sistemas robóticos usando los estándares de la OMG," in *Proceedings of the 17th Workshop de Investigadores en Ciencias de la Computación (WICC '15)*, Salta, Argentina, April 2015.

[19] F. V. Cipolla-Ficarra, *Advanced Research and Trends in New Technologies, Software, Human-Computer Interaction, and Communicability*, IGI Global, 2014.

[20] J. Gallardo, C. Bravo, and M. A. Redondo, "A model-driven development method for collaborative modeling tools," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1086–1105, 2012.

[21] T. Y. Tang, P. Winoto, and H. Leung, "A usability study of an educational groupware system: supporting awareness for collaboration," *Journal of Educational Computing Research*, vol. 50, no. 3, pp. 379–402, 2014.

[22] J. Janssen and D. Bodemer, "Coordinated computer-supported collaborative learning: awareness and awareness tools," *Educational Psychologist*, vol. 48, no. 1, pp. 40–55, 2013.

[23] A. I. Molina, W. J. Giraldo, M. Ortega, M. A. Redondo, and C. A. Collazos, "Model-driven development of interactive groupware systems: integration into the software development process," *Science of Computer Programming*, vol. 89, pp. 320–349, 2014.

[24] M. A. Redondo, A. I. Molina, and C. X. Navarro, "Extending CIAM methodology to support mobile application design and evaluation: a case study in m-learning," in *Cooperative Design, Visualization, and Engineering*, vol. 9320 of *Lecture Notes on Computer Science*, pp. 11–18, 2015.

[25] H. Alcazar, J. Martinez, L. Pantoja, C. Collazos, and A. Paz, "Method for incorporating awareness mechanisms in driving simulation environments," *IEEE Latin America Transactions*, vol. 12, no. 1, pp. 36–41, 2014.

[26] J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-driven engineering practices in industry: social, organizational and managerial factors that lead to success or failure," *Science of Computer Programming*, vol. 89, pp. 144–161, 2014.

[27] M. Völter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, *Model-Driven Software Development: Technology, Engineering, Management*, John Wiley & Sons, 2013.

[28] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers, Davis, Calif, USA, 2012.

[29] M. Banerjee, *Efficient model driven development techniques using aspect oriented and feature oriented programming [M.S. thesis]*, Department of Computer Science and Engineering, Indian School of Mines, 2015.

[30] P. A. Akiki, A. K. Bandara, and Y. Yu, "Adaptive model-driven user interface development systems," *ACM Computing Surveys*, vol. 47, no. 1, article a9, 2014.

[31] R. Duque, C. Bravo, and M. Ortega, "A model-based framework to automate the analysis of users' activity in collaborative systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1200–1209, 2011.

[32] T. Taleb, A. Benslimane, and K. B. Letaief, "Toward an effective risk-conscious and collaborative vehicular collision avoidance system," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1474–1486, 2010.

[33] N. Kumar and N. Chilamkurti, "Collaborative trust aware intelligent intrusion detection in VANETs," *Computers & Electrical Engineering*, vol. 40, no. 6, pp. 1981–1996, 2014.

[34] N. Salameh, G. Challita, S. Mousset, A. Bensrhair, and S. Ramaswamy, "Collaborative positioning and embedded multisensors fusion cooperation in advanced driver assistance system," *Transportation Research Part C: Emerging Technologies*, vol. 29, pp. 197–213, 2013.

[35] M. H. Firooz and S. Roy, "Collaborative downloading in VANET using network coding," in *proceedings of the IEEE International Conference on Communications (ICC '12)*, pp. 4584–4588, Ottawa, Canada, June 2012.

[36] M. Jean-Aimé, "Model-driven approach to design a secure routing protocol for UAV Ad hoc networks," in *Proceedings of the 15ème Congrès des Doctorants (EDSYS '15)*, Toulouse, France, May 2015.

[37] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular Ad Hoc network," *Journal of Network and Computer Applications*, vol. 37, no. 1, pp. 380–392, 2014.

[38] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli, "Progress and challenges in intelligent vehicle area networks," *Communications of the ACM*, vol. 55, no. 2, pp. 90–100, 2012.

[39] J. Gallardo, A. I. Molina, C. Bravo, M. A. Redondo, and C. A. Collazos, "An ontological conceptualization approach for awareness in domain-independent collaborative modeling systems: application to a model-driven development method," *Expert Systems with Applications*, vol. 38, no. 2, pp. 1099–1118, 2011.

[40] S.-P. Lin and N. F. Maxemchuk, "An architecture for collaborative driving systems," in *Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP '12)*, pp. 1–2, Austin, Tex, USA, November 2012.

[41] A. King and I. Lee, "Methods, systems, and computer readable media for enabling real-time guarantees in publish-subscribe middleware using dynamically reconfigurable networks," U.S. Patent Application 14/452,155, 5 Ago, 2014.

[42] M. A. Tariq, B. Koldehofe, G. G. Koch, I. Khan, and K. Rothermel, "Meeting subscriber-defined QoS constraints in publish/subscribe systems," *Concurrency Computation: Practice and Experience*, vol. 23, no. 17, pp. 2140–2153, 2011.

[43] W. He, G. Yan, and L. D. Xu, "Developing vehicular data cloud services in the IoT environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1587–1595, 2014.

[44] C. Simonds, "Software for the next-generation automobile," *IT Professional*, vol. 5, no. 6, pp. 7–11, 2003.

[45] L. Juan, W. Feng, and A. F. A. Abdo, "Vehicle service middleware based on OSGi," in *Proceedings of the 3rd International Conference on Computer Science and Service System*, pp. 714–717, Atlantis Press, Bangkok, Thailand, June 2014.