An Extensible Software Framework for Building Vehicle to Vehicle Applications

Hazem M. Hajj Wassim El-Hajj Mohamad El Dana Marwan Dakroub Faysal Fawaz
Electrical and Computer Engineering Department, American University of Beirut,
Beirut, Lebanon 1107-2020
hh63@aub.edu.lb we07@aub.edu.lb mae40@aub.edu.lb mhd11@aub.edu.lb fjf02@aub.edu.lb

ABSTRACT

Artificial intelligence and wireless technologies have progressed rapidly in the last decade driven by technology evolution and new emerging applications. The automotive industry has the opportunity to benefit from progress in these fields. In this paper, we propose an extensible software framework that can be implemented on all vehicles. It comprehends the capabilities supporting wireless communication and analytics for collaborative intelligence among vehicles and data mining solutions. The architecture has four key modules for: interface to on-board controls, external wireless communication, artificial intelligence, and an internal virtual bus. The design is implemented as a framework so that new applications can be developed with minimal additions to achieve high end solutions. We demonstrate the effectiveness of the architecture with vehicle safety applications, and illustrate the use of collaborative intelligence. A potential implementation is proposed with multi-threads lending itself to parallel programming and high performance computing for real-time response.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *network communications, wireless communication*. I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *industrial automation.* B.4.3 [**Input/output and Data Communication**]: Interconnections – *interfaces, topology*.

General Terms

Design, Performance

Keywords

Framework, extensible software, vehicle to vehicle, artificial intelligence.

1. INTRODUCTION

Not long ago science fiction movies were showing autonomous vehicles transporting passengers around futuristic cities. These vehicles would interact with their passengers, entertain them, and provide them with information on any desired subject, while safely carrying them to their destination. These vehicles no longer belong in science fiction. This paper proposes an efficient modular architecture to make those features ubiquitous in today's vehicles. Today's vehicle manufacturers are actively pursuing research and development to equip their vehicles with capabilities which can be enabled with our framework.

Among intelligent vehicle applications, passenger safety continues to be top priority as mortality from accidents [1] remains among the leading causes of deaths. In the recent years, there has been a high interest in leveraging vehicle to vehicle (v2v) technology for proposing new vehicle applications such as parking spots identification [2], traffic jams [3], driver assistance [4], and fleet management [5]. Automotive software development dates back to the late 1980s where it was mostly restricted to vehicle manufacturers. The initial focus was on software engineering practices [6-9]. It then shifted to advanced models [11] and applications [10] where Schaefer proposed technologies for autonomous spacecraft with focus on uninhabited air vehicles. More recent efforts on vehicle software have shifted towards standardizing development under the AUTOSAR [12] open source framework standard for global cooperation in automotive software development with the goal of full proliferation to all new development through 2011. In summary, the automotive industry has seen development of methods and architectures to support vehicle software but still these techniques have not yet expanded to support developers outside of the automotive industry, and have not comprehended newer technologies such as v2v communication and potential applications such as electronic gaming, electronic banking, or other potential artificial intelligence applications.

On the vehicle communication and networking front, there has been an increased interest in this area which has led to the development of the 802.11p standard [13] for dedicated short range communication (DSRC). There have also been several proposals for communication rules and networks such as the scheduling process suggested for blind intersection [14], the modeling of vehicles as a mobile ad-hoc network of clusters [15], or the use of dedicated infrastructure to aid the v2v communication [16]. It was observed that vehicle networks have several challenges including delays [17] and losses associated with unpredictable mobility [18].

In this paper, we propose a new extensible and modular solution that comprehends new technologies such as v2v communication and GPS, and allow developers outside of the automotive industry to develop applications for users' comfort, entertainment, luxury or safety using their vehicles. The extensible software framework requires elements to support communication and networking with other vehicles and fixed infrastructure. The proposed architecture can integrate with AUTOSAR framework and is consistent with the use of the 802.11p DSRC standard. We expect that once the communication infrastructure is in place for vehicles, new applications will emerge frequently. Our proposed extensible software architecture will be critical to quick implementation, providing a competitive advantage for car manufacturers. It should support ease of development to allow entertainment applications and connection with the world-wide-web. The extensible software architecture could be implemented on any vehicle. It can be enriched with new applications which would make the area of intelligent vehicles the new technological revolution with impact on society comparable to the internet introduction. In the rest of the paper, we will show how our architecture achieves the following objectives: functional requirements for vehicle applications, modularity for ease of improvements, extensibility for quick application development, and accessibility for developers beyond the automotive industry. The paper is organized as follows. Section 2 covers the framework functional requirements. Section 3 includes a description of the architecture modularity and the corresponding modules. Section 4 proposes an implementation that can be used for simulation and framework evaluation. Section 5 demonstrates the extensibility of the framework with several examples and section 6 concludes the paper.

2. Framework Requirements

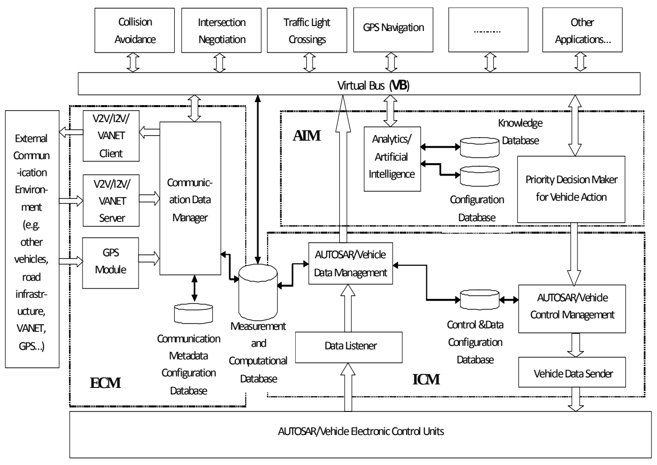
The goal of our software solution is to easily enable the development of new applications for intelligent vehicles. The purpose of the framework is to simplify the development of the application, and provide building blocks so only few additional components would be needed [19]. We propose a callable [20] grey-box framework [21], where some components could be readily available as black-box elements based on existing standards such as interfaces to IEEE 802.11p and AUTOSAR, while others would be formed through continuous evolution and can be open for applications use. The goal of our proposal is to provide a framework where different developers from different companies or industries can build additional applications for intelligent vehicles. Ultimately, we would like individual car drivers or passengers to be able to develop their favorite applications on the vehicle software platform, and be able to enjoy the latest advances in wireless technologies where all vehicles are connected together, to the internet, and to drivers' personal mobile devices. The application possibilities are endless.

The system needs to support on-board and collaborative intelligence. As a result, the framework needs to provide the building blocks needed for all vehicles. The components need to support analytics for intelligent applications, interfaces external to the vehicle, and interfaces internal to the vehicle. External interfaces include: GPS satellites, other vehicles enabled with wireless communication, fixed infrastructure communication (i2v), and sensor to sensor communication. Internal interfaces include integration with on-board hardware and vehicle intelligence software, where the on-board hardware may include sensors, cameras, vehicle control unit, stereo system, and brakes. The analytics can provide artificial intelligence building blocks for characterization of a situation or prediction of future risks. Our framework should support integration with AUTOSAR software which allows it to read vehicle data and send instructions through it.

In the following sections, we propose an extensible software architecture that can be leveraged for plug and play feature additions. We will start by showing the general purpose modular architecture and then demonstrate the ease of use for delivering special-purpose applications such as vehicle to vehicle intersection negotiation, obstacle avoidance, and GPS-based navigation.

3. Modular Architecture

To meet the framework functionality while achieving modularity, we propose five modules for the framework as shown in Figure 1: The Intra-Vehicle Communication Module (ICM), the External Communication Module (ECM), the Artificial Intelligence Module (AIM), the Virtual Bus (VB), and the Databases (DB). We briefly describe each module below.



Collision Avoidance
Intersection Negotiation
Traffic Light Crossings
GPS Navigation
....
Other Applications...
Virtual Bus (VB)
External Commun -ication Environ- ment (e.g. other vehicles, road infrastr- ucture, VANET, GPS...)
V2V/I2V/ VANET Client
V2V/I2V/ VANET Server
GPS Module
Communic- ation Data Manager
Communication Metadata Configuration Database
Measurement and Computational Database
ECM
AIM
Analytics/ Artificial Intelligence
Knowledge Database
Configuration Database
Priority Decision Maker for Vehicle Action
AUTOSAR/Vehicle Data Management
Data Listener
Control &Data Configuration Database
AUTOSAR/Vehicle Control Management
Vehicle Data Sender
ICM
AUTOSAR/Vehicle Electronic Control Units

Figure 1. Proposed framework architecture providing intra-vehicle communication module (ICM) through the AUTOSAR/Vehicle data and control interfaces. External communication module (ECM) is enabled by the communication data manager and the related components. The artificial intelligence module (AIM) of the applications is supported by the analytics and the priority decision maker components. The virtual bus (VB) and the databases are shared among modules. Examples of plug-in applications shown on the top interact with the framework through the VB.

3.1 Intra-Vehicle Communication Module (ICM)

The ICM handles communication within the framework and vehicle. The AUTOSAR software and the vehicle electronic control units (ECU) found at the base of the framework are considered external to the framework but internal to the vehicle. They are used to read data from the vehicle (such as sensor readings) and to send instructions to the vehicle components (such as the braking system). As seen in the ICM, data is passed through the Data Listener, which reads it and forwards it to the AUTOSAR/Vehicle Data Management block. This block ensures that the data is mapped to the format required for correct functioning of the framework, before placing it on the VB. The AUTOSAR/Vehicle Control Management block ensures that the instructions passed to it comply with the requirements of the vehicle and AUTOSAR (to ensure proper execution), and then forwards the instructions to the Vehicle which sends the instructions to the vehicle/AUTOSAR.

3.2 External Communication Module (ECM)

The ECM cluster is responsible for all secure communication external to the vehicle. The Communication Data Manager manages all incoming and outgoing communication. It reads the v2v messages which applications need to send to other vehicles from the Virtual Bus, ensures that they are in the correct format, and forwards them to the V2V/I2V/VANET client which in turn can send the data externally to other vehicles. For incoming data (V2V/I2V/VANET messages and GPS positioning data) the Communication Data Manager (CDM) handles the encryption routines for security, standardizes the data format, and places the data on the Virtual Bus for access by the applications. The V2V/I2V/VANET Client block sends all data it receives wirelessly through the vehicle's wireless interface. The V2V/I2V/VANET Server block is where all communication addressed to the vehicle is received. It then forwards all messages received to the Communication Data Manager. The GPS module contains a GPS receiver and forwards the vehicle's positioning data to the Communication Data Manager for placement on the Virtual Bus and subsequent use by applications such as GPS Navigation. For outgoing messages, the CDM is also responsible for message encryption.

3.3 Artificial Intelligence Module (AIM)

The AIM cluster is responsible for all analytics and intelligence required by the framework and the applications. The Analytics/Artificial Intelligence block performs analytic functions on the data it receives (based on application requests), and it services intelligence used in applications such as collision avoidance or route prediction. It would contain classification and prediction utilities such as Support Vector Machines, Bayes classifier, Classification and Regression Trees, and other fundamental statistical computations. To the far right of the framework diagram, the Priority Decision Maker for Vehicle Action reads all instructions sent by the applications from the Virtual Bus and decides which action or actions must be executed. For example, the collision avoidance system might be telling the vehicle to apply the brakes while at the same time the GPS navigation system would be telling the vehicle to continue going straight. In this case, the Priority Decision Maker would read both instructions, decide that applying the brakes is more urgent than continuing forward, and will only forward the braking instruction to the AUTOSAR/Vehicle Control Management block.

3.4 Virtual Bus (VB)

The VB component consists of the Virtual Bus. It acts as a bus where all communication between the framework blocks and the applications occurs. Applications would read any information they need from this VB and would write their output onto it too. As modeled in the diagram, all communication between blocks of the framework occurs directly, but all communication between the framework blocks and the plug-and-play applications occurs through the Virtual Bus.

3.5 Databases

In addition to the above mentioned framework components, the framework contains 5 databases. The Communication Metadata Configuration database contains information on the data formats required. The Measurement and Computational Database stores data from the Communication Data Manager, the AUTOSAR/Vehicle Data Management block, and from the Virtual Bus (data from the Analytics/Artificial Intelligence block and from applications). The Control and Data Configuration Database contains data on the framework's requirements for the data format and the vehicle's/AUTOSAR's requirements for the instruction format. The configuration database contains the configuration of the Analytics/Artificial Intelligence block, and the Knowledge Database stores the knowledge from that block.

New applications built on our framework require plug-in additions for custom logic. These are illustrated on the top of the framework in figure 1. These contain logic specific to the application. In the following section, we will illustrate how these plug-ins are added and demonstrate how the proposed framework can be easily extended to support special purpose vehicle applications for safety and driver assistance.

4. Simulation

To demonstrate the effectiveness of our proposed architecture, we implemented several components of the framework and we tested several applications using four wireless mobile robots to simulate vehicles. The implementation is based on multi-threaded components, publish-subscribe messaging, and a service-based architecture where components' actions are triggered based on incoming data or events. The primary driver for these choices is the need for fast communication between the components.

4.1 Main Process

The pseudo-code for the implemented software architecture is shown in figure 2. The main program is instantiated when the vehicle is started. First (line 3), it downloads the configuration of the framework and of the plugged-in applications. It then uses multi-threading to initiate the different blocks of the framework starting with the ICM (line 4), the ECM (line 5), the AIM (line 6), and then initiates the application specific plug-ins (line 7). Next, the program goes into a loop (line 8) that ends whenever the vehicle is switched off. In this loop, the main program will read sensory data from the vehicle (Line 10) and external data from the ECM server (line 11), then publishes the collected data to all applications (lines 12 and 13). When plug-in applications act on the data, their responses are sent back through the virtual bus. From there, the data could be received by the Priority Decision Maker or by the ECM communication manager or both. These two components will pass on their actions and data to the internal vehicle controls through the ICM control manager (line 15) or to the external environment (line 16). In the next sections, we will cover the wireless (v2v and i2v) implementation, and the plug-in for vehicles negotiating pass through at intersections using v2v.

```
Pseudocode implementation for the main process of the framework
begin
    \\ While car is turned on
    Load_configuration_for_framework_and_applications;
    Initiate_thread_for_ICM; \\ Initiates vehicle control manager and
    ICM data listener
    Initiate_thread_for_ECM; \\Initiates Communication data
    manager, enables receiving and sending data using V2V
    Initiate_thread_for_AIM; \\Initiates Priority decision maker for
    vehicle action
    Initiate_plug_in_application_threads();\\all plugged-in applications
    such as collision avoidance
    While(Car_is_on)
    begin
        Read_data_from_within_vehicle(); \\ If new data available, use
        ICM data listener
        Read_data_from_external_sources; \\If new external data
        available, use ECM server
        Publish_ICM_data(); \\Make new data available to virtual bus
        using ICM AUOTSAR/Vehicle data management
        Publish_ECM_data(); \\Make new data available to virtual
        bus using ECM Communication data manager
        Read_Priority_decision_maker_output(); \\receive commands
        from AIM module
        Execute_Priority_decision_maker_output_To_vehicle(); \\use
        Vehicle control manager in ICM
        Transmit_plug_in_specific_data_using_V2V; \\to communicate
        with v2v/i2v using ECM client
    end
end
```

Figure 2. Pseudo-code for the main branch of the framework, where all components get initialized along with the application plug-ins. The main branch keeps running operating on incoming data and events until the vehicle is turned off.

4.2 ECM Threads for Wireless V2V, I2V client and server implementations

To support vehicle to vehicle communication and infrastructure to vehicle communication, each vehicle must be able to send and receive data at the same time as shown in the ECM. For this purpose, we designed multi-threaded clients and servers in the application as described below. A server's pseudo-code, as shown in figure 3, will be called once the ECM is initiated in the main program.

```
Pseudocode implementation for ECM Server thread
begin
    Create buffer[ ...];
    struct server (hold server info: IP family, receive from any IP,
    port number);
    Create socket;
    bind(socket to server struct);
    buffer = recvfrom(...);
end
```

Figure 3. Pseudo-code for the implementation of the ECM server responsible for listening to incoming data from external wireless infrastructure and other vehicles in the network.

The server process needs a dedicated socket to wait for incoming connections. To create the socket we first need a structure to save its information. The socket's information may have Address family (PF_INET for Internet family), IP address, and port number. The dedicated socket is created using the function socket which returns a positive integer value if successful. Once created, the socket must be bound to the address structure and if successful can receive incoming messages. We used the User Datagram Protocol (UDP) to eliminate the need for handshake between different vehicles, and achieve faster communication. Another alternative to the use of UDP would have been the Transport Control Protocol (TCP), which is characterized by its reliability. TCP has built-in functionalities that ensure the delivery of transmitted data packets. This feature however will reduce communication speed, since TCP has to perform a handshake procedure before transmitting any data which adds overhead. We tested the TCP implementation which required in addition to the UDP steps described above, to listen for incoming connection and to accept the connection before receiving any messages. In our testing, UDP turned out to be more beneficial for two reasons. First, no handshake was needed to send and receive messages which translate to faster communication. Second, UDP supports message broadcasts; a property needed by many applications. In a similar manner, we implemented a UDP-based client to support broadcasting messages to external wireless infrastructures or other vehicles in the network.

4.3 Modules (ICM, VB, DB) Details

The ICM is implemented in a similar way to the ECM using client and server threads to communicate with on-board vehicle sensors and controls. The virtual bus (VB) is implemented through global data structures that are shared across framework threads and application plug-ins. The databases have schemas for the metadata. We also propose in future work to implement dynamic schemas that would be instantiated through XML definitions. The applications can then decide what type of data storage is needed, and would communicate the specifics through XML.

5. Evaluation: Extensibility and Accessibility

In this section, we demonstrate the use of the extensible framework proposed in the previous section to deliver safe navigation. Accessibility of the software to the general population of developers is enabled by the independence of the framework on custom software. The examples include software for negotiating pass-through at street intersection through direct vehicle to vehicle communication, collision avoidance driver-aid system, and GPS-based navigation assistance. The idea behind the choice of these examples is that they illustrate a class of applications leveraging the different available interfaces in the framework. These examples can be used as models to build other applications. For example finding a remote parking spot through vehicle to vehicle communication can leverage a similar solution to what we will propose for intersection negotiation.

In addition to the qualitative assessment, we propose a simple heuristic model for quantitative assessment of the extensibility. We call a "code set" a set of related code used in the implementation of a framework module or function. We consider the pseudocode proposed in the previous section, and we count for each statement calling a distinct function in the framework as one "code set". From Fig. 2, and subsections IV-B and IV-C, we count 15 units of code sets for the framework. We will use these units to give a quantitative assessment of extensibility in the following case study applications. We assume that all these 15 code sets are reusable for any application.

5.1 Collaborative Intelligence: Vehicles Negotiating Intersection Pass through Using V2V

For collaborative intelligence, we are referring to multiple vehicles applying collective intelligence and deciding together on the best action. For the case of vehicles negotiating pass-through at an intersection, the application would use the V2V/I2V/VANET Client block for transmission, and would rely on the V2V/I2V/VANET Server block for listening. The instructions to stop and go will be passed to the Priority Decision Maker for Vehicle Action which will decide on which action the vehicle will execute and forward the instruction for execution.

```
Pseudocode implementation for Intersection plug_in thread
1  begin
2    While(1)
3    begin
4      if_intersection_detetcted; \\ detecting i2v signal through ECM
5      Communicate_with_other_vehicles; \\ through ECM client
6      Make_decision_based_on_Rules_or_AI;\\ use AIM if needed
7      Communicate_decision_to_other_vehicles; \\ Ensures that there
       is no conflicts in decision, will repeat communication if needed
8      Publish_decision_to_virtual_bus; \\ Which in turn gets passed
       as a decision to the vehicle by main() program
9    end
10 end
```

Figure 4. Pseudo-code for the implementation of an application plug-in for vehicles negotiating at an intersection. Lines 4-6 are the only logic additions to the framework for the plug-in.

Here, we describe the plug-in needed to implement this application. We use the scenario of vehicles arriving at an intersection and negotiating directly for safe passage. The pseudo-code is shown in Figure 4. The intersection is detected using i2v communication by the ECM. The application initiates a thread to detect the presence of an intersection (line 4), and once detected the application ensures the safe crossing of vehicles. The pseudo-code illustrates the efficiency of the framework. Quantitatively, we can measure 5 units of code sets in the above pseudocode, where three of them are specific for the application. The rest is reusable from the framework. As a result, we conclude an assessment of an effective 85 % (17/20) reuse.

5.2 Other examples: Collision Avoidance and GPS

For collision avoidance, data about sensor readings and driver actions are sent from the AUTOSAR/Vehicle Electronic Control Units to the Virtual Bus. The collision avoidance application then forwards the sensor readings to the Analytics/Artificial Intelligence block which will return a collision risk level to the application. When the application receives this assessment (the risk level), it will forward the corresponding required action to the Priority Decision Maker for Vehicle Action, which will take care of executing the command. In this example, we would need a plug-in pseudocode similar to the one shown in Figure 3. Lines 4-6 would need to be altered to support the collision avoidance. For example, line 4 would be replaced by function call to check for sensor detection of obstacles.

For autonomous navigation, the GPS asks the passenger for the destination and preferences (fastest route/avoid highways…), and then using the maps stored in it and the current vehicle position obtained from the GPS Module, it will compute optimal routes leveraging the AIM analytics, and send instructions through the Priority Decision Maker for Vehicle Action described in the AIM block, to ensure that the vehicle successfully follows the calculated route to its destination. A more advanced implementation could include collaborative intelligence with other vehicles to select best routes and dynamically adjust based on other vehicles' predictions. Similar conclusions can be drawn here for the required changes to the pseudocode in Figure 3. Line 4 would be replaced by a function call to interact with the passenger's request. In both of these examples, we can see how little a developer has to add to the framework to complete an application. Quantitatively, these two applications also demonstrate an effective 85% reuse.

It is worth noting that the framework has a disadvantage in that it needs to be maintained and enhanced as technologies evolved. This could be resolved by making the framework implementation as open source. Discussions about scalability and reliability were not discussed above, but these would definitely need to be addressed through scalable and reliable server implementation when needed.

6. Conclusion

The advancement of wireless technologies has enabled communication for v2v, i2v, and GPS, and has opened the door for new exciting applications. Some of the v2v applications already exist, but research continues to look for the optimal infrastructure to support the rapid growth. There is a tremendous opportunity for rich applications leveraging the advances in wireless technology and high performance computing. But most of today's software for vehicle applications is custom built, which makes turn-around on applications somehow slow. As a result, there is a need for an infrastructure that helps with fast development and allows for leverage of new technologies before these could become obsolete. In this paper, we proposed an extensible and modular software framework for quick turn-around on building vehicle applications. The framework supports developing intelligent solutions with capabilities of intra-vehicle and external to vehicle wireless communication. Examples of special-purpose applications for vehicle safety were provided to demonstrate the effectiveness of the framework. The plug-in nature of the software makes it attractive to developers. The solutions are consistent with interfaces to state-of-the-art standards such as AUTOSAR vehicle software and the DSRC wireless communication standard. The proposed solutions are not just stand-alone concepts, but they are rather based on real scenarios that can be easily integrated with currently manufactured vehicles, and at low cost. In future work, we plan to examine special-purpose high performance hardware architecture for the framework integration. We will also consider artificial intelligence applications for safer, more reliable, and more predictable transportation.

7. ACKNOWLEDGMENTS

This work was supported in part by American University of Beirut Research Board (URB).

8. REFERENCES

[1] UNECE (United Nations Economic Commission for Europe) Transport Division, Statistics of Road Traffic Accidents in Europe and North America, 2007 edition.

[2] S. Miura, Z. Yi, T. Kuroda, "Evaluation of parking search using sensor network," in 1st International on Wireless Pervasive Computing, 2006, pp. 6.

[3] K. Ohara, Y. Nojima, H. Ishibuchi, "A study on traffic information sharing through inter-vehicle communication," in IEEE 22nd International Symposium on Intelligent Control, 2007, pp. 670-675.

[4] K. Ch. Fuerstenberg, U. Lages, "New European approach for intersection safety- the EC- project intersafe," in Proceedings of the Intelligent Vehicle Symposium, 2005, pp. 177-180.

[5] R. L. Sabounghi, "Intelligent vehicle highway system- the universal close-range road/vehicle communication system concept-the enhanced AVI and its CVO applications," in Vehicle Navigation and Information Systems Conference, 1991, pp. 957-967.

[6] M. D. Grover, "A uniform object/process model for autonomous vehicle component communication" in Proceedings of the IEEE International Symposium on Intelligent Control, 1988, pp. 113-117.

[7] B. Overton, I. Spalding, M. Thomas, "Issues in the validation and verification of vehicle software," in Eight International Conference on Automotive Electronics, 1991, pp. 98-101.

[8] D. Ward, "Guidelines for development of automotive software," Software Engineering Journal, vol. 11, pp. 76-81, March 1996.

[9] M. L. Nelson, "A Design Pattern for Autonomous Vehicle Software Control Architectures," in Computer Software and Applications Conference, 1999, pp. 172-177.

[10] P. Schaefer, R. D. Colgren, R. J. Abbott, H. Park, A. Fijany, F. Fishjer, M. L. James, S. Chien, R. Mackey, M. Zak, T. L. Johnson, S. F. Bush, "Technologies for reliable autonomous control (TRAC) of UAVs," in Proceedings of Digital Avionics System Conference, 2000, pp. 1E3/1 - 1E3/7

[11] D. Earle, D. Wallis, R. Wenham, "Future architecture and design trends for automotive control systems," in The 29th Annual Conference of the IEEE, 2003, pp. 2847-2852.

[12] Joergen Moessinger, "AUTOSAR: The standard for global cooperation in automotive software development", ATI 2008, Tokyo Japan.

[13] D. Jiang, L. Delgrossi, "IEEE 802.11p: towards and international standard for wireless access in vehicular," in IEEE Vehicular Technology Conference, 2008, pp.2036-2040.

[14] L. Li, W. Fei-Yue, "Cooperative driving at blind crossings using intervehicle communication," IEEE Trans. Vehicular Technology, vol. 55, pp. 1712 - 1724, Nov. 2006.

[15] Y. Lui, U. Ozaguner, "Effect of inter-vehicle communication on rear-end collision avoidance," in Proceedings of the IEEE Intelligent Vehicle Symposium, 2003, pp. 168-173.

[16] K. Tsukamoto, H. Nakata, M. Fujii, M. Itami, K. Itoh, "A study on collision avoidance in the system that integrates IVC and RVC," in Proceeding of the Intelligent Vehicles Symposium, 2003, pp. 1-5.

[17] T. Sadayuki, K. Shit, "Evaluation of incident information transmission on highways over inter-vehicle communications," in Proceedings of the Intelligent Vehicle Symposium, 2003, pp. 12-16.

[18] J. J. Blum, A. Eskandarian, L. J. Hoffman, "Challenges of intervehicle ad hoc networks," IEEE Trans. Intelligent Transportation System, vol. 5, pp. 374-351, Dec. 2004.

[19] ME Fayad, DC Schmidt, RE Johnson, "Building application frameworks: object-oriented foundations of framework design, - 1999", John Wiley & Sons, Inc. New York, NY, USA

[20] Sparks, S. Benner, K. Faris, C., "Managing Object-Oriented Framework Reuse Computer, Los Alamitos 1996, vol. 29, number 9, pp. 52-62.

[21] H. Züllighoven, R. F. Beeger, Object-oriented Construction Handbook: Developing Application-oriented Software with the Tools & Materials Approach, Illustrated, Elsevier, 2004.