Organization and Operation of Electronically Coupled Truck Platoons on German Motorways

Ralph Kunze, Richard Ramakers, Klaus Henning, and Sabina Jeschke

RWTH Aachen University,
Center for Learning and Knowledge Management and
Department of Information Management in Mechanical Engineering (ZLW/IMA),
Dennewartstr. 27, 52068 Aachen, Germany

Abstract. One opportunity to manage the increase of freight transportation and to optimize utilization of motorway capacities is the concept of truck platoons. With the aid of Advanced Driver Assistance Systems, trucks are electronically coupled keeping very short gaps (approx. 10 meters) to form truck platoons on motorways. This contributes to optimisation of traffic flow and reduction of fuel consumption advantaged by slipstream driving. In this paper, a brief introduction into these truck platoons is given as well as a short overview about the elements of the automation-, information- and automotive-technology of the experimental trucks. The paper focuses on the Driver Information System which helps truck drivers to organize and operate these platoons. A generic software architecture for the Driver Information System of the platoon system is presented, which guarantees the development of a modern, flexible, extensible and easily configurable system, especially for Human Machine Interfaces of Advanced Driver Assistance Systems.

Keywords: Truck Platoons, Advanced Driver Assistance System, Driver Information System, Software Architecture, Electronically Coupled, HMI.

1 Introduction

The continuing traffic increase in Europe during the last years poses a huge challenge, especially for transit countries such as Germany. Due to the increase of freight transportation, the maximum road capacity in several countries worldwide is nearly reached.

In some countries in Asia and the Pacific, the road density increased up to 100% between 1990 and 2008. More than half of the Asian and Pacific countries had to face an increase of over 20% in that period of time [1]. In Europe a growth rate of 35% in road freight transport was detected between 1995 and 2004. Furthermore, an increase of over 55% in road transportation is expected between the years 2000 and 2020 [2]. In the year 2003, the European Commission stated that every day 7.500 kilometers of the European road system are being blocked by traffic jams [3].

Additionally, the integration of the new European member countries in combination with German's centricity within Europe provides another challenging component for Germany's national traffic planning. Environmental pollution, safety risks and a

M. Xie et al. (Eds.): ICIRA 2009, LNAI 5928, pp. 135–146, 2009.

loss in efficiency for the economy are only some of the effects that result from these factors. Similar problems are known and discussed worldwide.

1.1 The Approach

One possibility to meet the rising transport volume on roads is the modal shift to other types of transportation (e.g. rail, shipping). Another opportunity lies in the optimisation of the road-side traffic flow by driving assistance systems. Since the 90s, Advanced Driver Assistance Systems (ADAS) for trucks have been on offer, including pre-adjustment of speed and distance to the front vehicle. The combination of an Adaptive Cruise Control (ACC) together with an Automatic Guidance (AG) leads to autonomous driving. Here, a precondition is a computerised engine- and brakes-management in connection with an automated transmission.

The difference between platooning and autonomous driving makes the necessity of a leading vehicle. Following trucks can go far distances without any manual engagement by the driver as long as another ahead-driving vehicle exists. Nevertheless, each truck must be assigned with a truck driver at all times due to legal rules and regulations. Within platoons, smaller distances between the vehicles (up to 10 meters) can be realized. These truck platoons contribute to an optimization of traffic flow of up to 9% and a reduction of fuel consumption of up to 10% due to slipstream driving [4].

1.2 Objective

The development of a generic software architecture for a Driver Information System (DIS) for the organization and operation of truck platoons is the objective of this paper. This paper focuses on the DIS of the platoons system and how a generic software architecture of a DIS as a human machine interface (HMI) for ADAS should be designed. Consequently, the system architecture of the ADAS will be not the subject of this paper. A detailed overview of the system architecture can be found in Henning et al. [5].

2 The Scenario “Driver Organized Truck Platoons”

The Project KONVOI is based on the scenario “Driver Organized Truck Platoons” (Figure 1) which was developed in the Project “Operation-Scenarios for Advanced Driving Assistance Systems in Freight Transportation and Their Validation” (EFAS) [6]. The development and evaluation of the practical use of truck platoons is the objective of the project KONVOI, which was funded by German’s Federal Ministry of Economics and Technology. The Project KONVOI is an interdisciplinary research project with partners of the RWTH Aachen University, industry and public institutions, which ended after a duration of 49 months at 09/05/31. With the assistance of virtual and practical driving tests, by using experimental vehicles and a truck driving simulator, the consequences and effects on the human, the organization and the technology have been analyzed [5].

In the scenario “Driver Organized Truck Platoons”, the platoons can operate on today’s existing motorways without extending the infrastructure and the driver has the

permanent control of the autonomous driving procedures [6]. The creation of a platoon depends on the initiating driver who delivers the necessary data about time and place of meeting, the destination, as well as the required truck telemetric data (loading weight, engine power etc.) with the help of a Driver Information System. Because no schedules have to be generated like they have to be in rail traffic, the high flexibility of truck transportation is not lost. After activating the ADAS, it automatically shows a selection of the best matching platoons, informs the driver and prepares the participation to the selected platoon. The DIS acts as human machine interface of the platoon system and helps the truck driver to plan the route and guides the driver to the meeting point [7].

The driver has to initialize and respectively confirm all of the platoon maneuvers in order to build and to dissolve the platoon. As soon as the final position in the platoon is reached, an automated longitudinal guidance with a target distance of 10 meters between the trucks and a lateral guidance is possible. On one hand, this target distance was chosen because the short distance prevents most drivers from driving between the platoons. On the other hand, the short distance causes slipstream effects, which can lead to a reduced fuel consumption of about 10%.

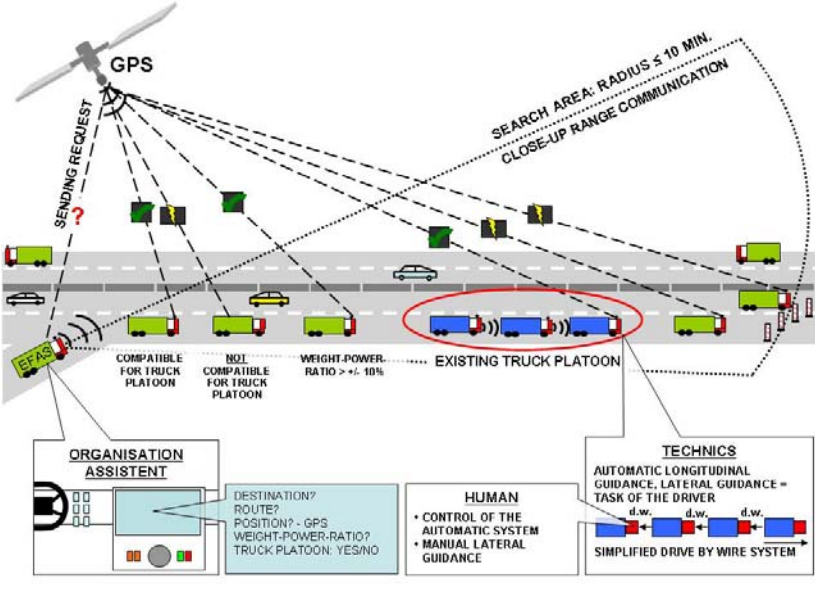

Fig. 1. Scenario 1 – "Driver Organized Truck Platoons" [6]

Since road markings are needed for the lateral guidance, the platoon system is exclusively developed for the use on motorways. Because of a limitation for most trucks at approx. 50 mph, the speed of the trucks on motorways differs only slightly. Therefore, the truck platoons are operated at a speed between 37 and 50 mph. This speed can be managed safely at 10 meters distance by the KONVOI-System.

3 The Platoon System

In order to realize different platoon sizes, four experimental vehicles have been equipped with the required automation-, information- and automotive-technology (Figure 2). The main components for the implementation of the system architecture in the experimental vehicles are the actuators (steering and power train), the sensors (object registration in close-up and far range, recognition of lane), the vehicle-vehicle-communication (WLAN), the automation unit (coordination of the different vehicle states), the control unit (adaptive cruise control and automatic guidance) and the driver information system (human-machine interface, organization assistant, GPS and 3G) [5]. The longitudinal guidance of the ADAS is based on a LIDAR distance sensor, a CMOS-Camera and a RADAR-sensor. The distance sensors are used to determine the distance in longitudinal direction and the lateral offset to the leading vehicle. The vehicle-vehicle-communication transfers necessary vehicle data from all platoon members, which are required for the ACC to realize the target distance of 10 meters. In all trucks, a target acceleration interface is implemented, which automatically calculates the drive-train and the management of the different brakes in the vehicles. The acceleration is either calculated autonomously for each vehicle or deduced from the data which is transferred via the vehicle-to-vehicle-communication.

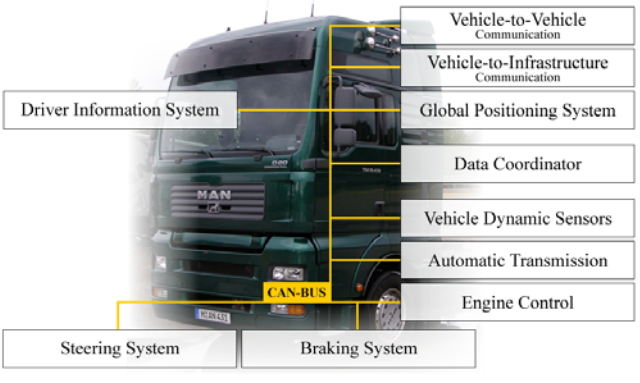

Fig. 2. Automation-, Information- and Automotive-Technology of an Experimental Vehicle

Every experimental vehicle is attached with cameras which are able to identify the traffic lane, thus determining the position of every truck within the traffic lane. An electronically accessible steering system has been integrated additionally. A steering actuator on the base of an electric motor delivers the necessary steering moment for the automated guidance of the trucks [7].

With the help of the Driver Information System, the truck driver plans his route, selects economic platoon participants as well as initialises and respectively confirms the platoon manoeuvres in order to build and to dissolve the platoon.

The platoon organisation is realised on a central server with a data-mining-algorithm under consideration of economic aspects [8]. For this task, the DIS has to send the time schedule, route plan and GPS position of the truck with a vehicle-infrastructure-communication via G3 to the central server (Figure 3).

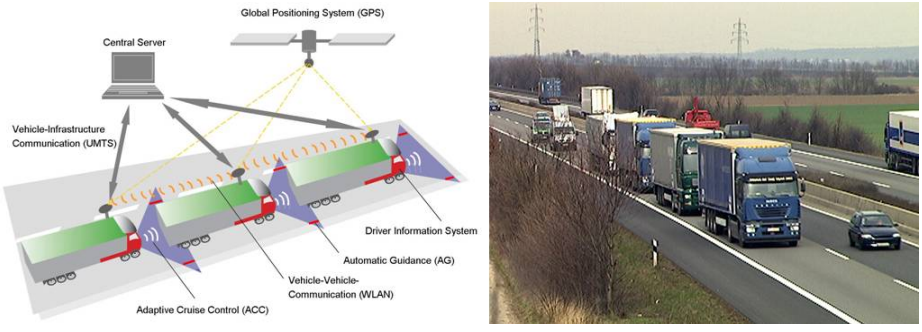

Fig. 3. The Platoon System [9] and Test Run on German Motorways (March 2009)

4 Driver Information System (DIS)

The DIS is not only the HMI of the platoon system, it also acts as information manager between the truck driver and the ADAS as well as the central server. Via touch screen, the user input of the truck driver is processed by the DIS, and in dependency of the user input the data is given to the technical systems. Vice versa, the DIS processes the data of the technical systems and informs the driver in all platoon phases about the current platoon situation (Figure 4).

During the platoon organization, the driver has to enter the DIS settings, planning his route and time schedule and choosing a suitable platoon from an offer list. This data is preprocessed by the DIS and sent to the central server via 3G. The most important data for the central server is the route information of each truck as well as the platoons chosen by the driver. Additionally, the DIS gets a list with suitable platoons sorted by economic criteria from the central server. Furthermore, the central server immediately informs the DIS about any alterations within the planned platoons.

Two further tasks of the DIS are the navigation to the planned meeting points/destinations and the warning of danger areas such as road constructions, bridges, motorway junctions and tunnels. The platoons have to be dissolved manually by the truck driver ahead of these areas.

During the platoon drive, the drivers have to initiate and respectively confirm all platoon maneuvers (connecting, dissolving and lane change). The control signals from the driver to the ADAS are sent through CAN-Bus. The DIS permanently informs the driver about the actual state of the platoon. This applies to the manual as well as to the automated platoon drive.

4.1 Requirements Specification for the Software Architecture

In the following, the requirements of the DIS software architecture are indicated with the letter “R” (for requirement) and a consecutive number.

Modularity, Extensibility, Flexibility and Configurability (R1)
The usual demands made to software architectures for an HMI are modularity, extensibility, flexibility and configurability. In this context, modularity means that certain

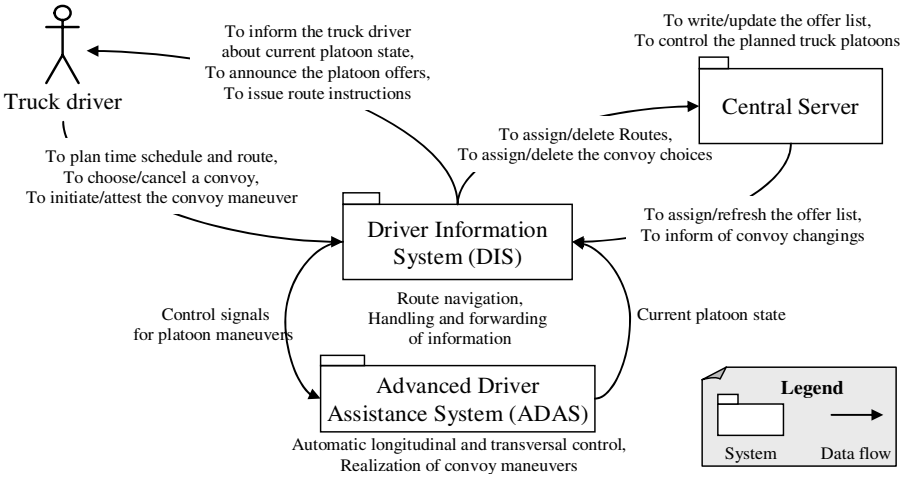

Fig. 4. The DIS as the Information Manager [9]

functionalities are combined in well-defined software components. These components have to be self-explanatory and exchangeable. The extensibility of the software architecture has to be as flexible as possible in order to allow additional functions subsequently and easily. The whole software system – particularly the graphical design of the HMI – is fast to configure and change with a configuration file to avoid alterations of the source code.

Robustness and Reliability (R2)
The automobile sector has especially high demands on technical systems in vehicles concerning robustness and reliability [10]. Therefore, the software architecture has to ensure the robustness and reliability through adequate safety mechanisms and functions.

Information and Data Management (R3)
The DIS as an information manager has to handle and process a large quantity of data as well as the transmitting of processed data to the corresponding technical systems. Therefore, the system architecture has to support an internal communication, in order to support the different software components with the required data. Furthermore, the software architecture has to support different functions to manage the data in the internal memory as well as in data bases.

External Communication with System Environment (R4)
The DIS is capable to communicate with their system environment. Accordingly, an external communication with appropriate communication interfaces has been implemented. Furthermore, the ease extensibility (also for other technologies e. g. WLAN, Flexray) has to be guaranteed by the software architecture (cf. requirement R1).

Interaction with the User (R5)
The DIS is the HMI of the platoon system. The relevant input is made by the driver on the user interface (e.g. touch screens). The system architecture handles the user input through the user interface and gives the effects of the users' manipulation back to the user interface so that the user can assess the system state.

Table 1. Requirements for the Software Architecture

R1	Modularity, Extensibility, Flexibility and Configurability
R2	Robustness and Reliability
R3	Information and Data Management
R4	External Communication with System Environment
R5	Interaction with the User

4.2 Design of the Software Architecture

The software architecture comprises software components, the externally visible properties of those components and the relationships between them [11]. The choice of an architecture pattern, as the base for the software architecture, is a fundamental one and also one of the first decisions to make [12]. The decision for an architecture pattern was made on the basis of the defined requirements (chapter 4.1).

The claimed interaction with the user (R5) classifies the DIS as an Interactive System. The DIS has to process all the incoming data (R3) and has to exchange this data with the system environment, thereby relying on the external communication interfaces (R4).

Compared to the required aspects of modularity, extensibility, flexibility and configurability (R1), the DIS can be regarded as an Adaptive System, which must run robust and reliable in the car (R2).

Consequently, the defined requirements can be summarized as the claim for a modular, extensible, flexible and configurable HMI, whereby especially the aspect of extensibility accounts for the modularity and flexibility of the software architecture. The denotation of the DIS as a HMI makes the importance of a structured human-machine interaction obvious, so finally the Model-View-Controller architecture was chosen for the development of the DIS as an Interactive System [12].

Model-View-Controller (MVC) architecture
The component concept of the DIS is build on the basis of the Model-View-Controller (MVC) pattern introduced before. In some cases, the functionality of the software components is slightly different from the ones in the MVC literature. The architecture – following the MVC architecture pattern – is based on three core components, a Controller-, a View- and a Model-Component [15].

The Model-component serves as a collection of abstract data structures which administrates the data. Unlike the MVC pattern, where the Model-component is responsible for manipulating the dates, the Model-component in the adapted MVC pattern has solely administrative tasks. These tasks are primarily the access control and thread backup. The other components (Controller and View) are unknown to the Model-component.

Along the lines of the general MVC pattern, the View-component displays the information provided by the Model-component. Data manipulation or other alterations by user interaction are communicated to the View-component by the Controller-component. The View-component knows the Model- as well as the Controller-component in order to readout data about user interaction. Simultaneously, it administrates the various GUI-components. These comprise graphical elements, which visualize selected data from the Model-component for the user.

The logic and functionality of the software is located in the Controller-component. This component evaluates user interactions and manipulates data. In addition, the Controller informs the View-component about data manipulation or special requests of the user leading to a new or modified notation of a GUI-component (e.g. change of notation, deactivation of functionality). The Controller-component also administrates Logic- and Communication-components. The Controller extends the program logic on the one hand, and on the other hand enables the possibility of external communication (e.g. CAN, 3G).

4.3 Implementation of the Software-Architecture

Figure 5 shows a simplified UML class diagram of the DIS software architecture. The components Model, View, Controller, Logic, GUI, Comm and ModelData are realized through several classes. To avoid complexity, a well arranged overview is given in the figure, by excluding the attributes, methods and multiplicities of the classes.

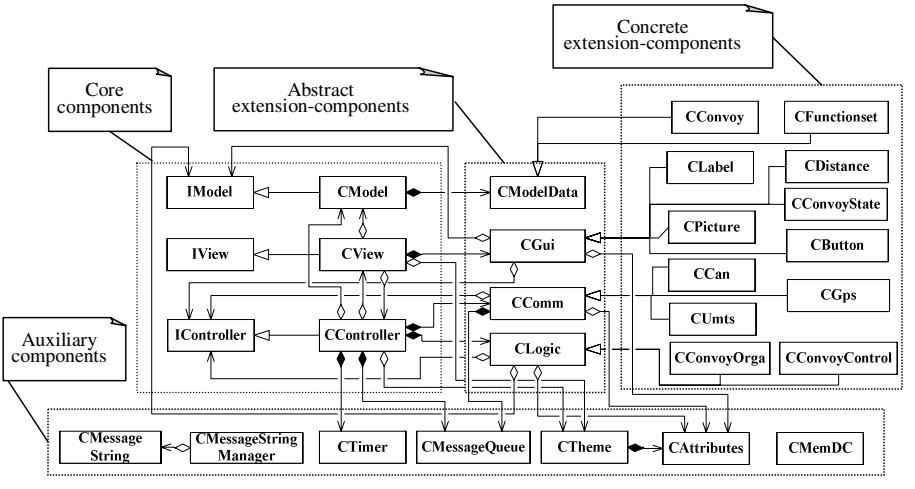

Fig. 5. UML class diagram of the DIS software architecture [9]

Basically, the software architecture is divided into core components, abstract extension-components, concrete extension-components and auxiliary components. This concept enables - next to the chosen software architecture - a high flexibility concerning the extensibility with program logics, data models and design elements. In addition, dynamic libraries are used to uncouple the core source code of the concrete extension-components from the main program of the DIS. Also, a static library for the DIS is used, where all

necessary definitions (e.g. classes, auxiliary classes) for the implementation of the software system are included. In the following, the realization of the different software components will be explained.

Core Components
The core components inherit from interface definitions (IModel, IView and IController), so that some functionalities are concealed to the extension- and auxiliary components and access is only allowed to designated functionalities. The overall functionality (according to the visibility of the methods of the classes) is only known to the core components among themselves and can only be used by them.

The internal communication between the Controller- and the View-components are done with an interprocess communication. Asynchrony, messages are saved into message queues until the recipient retrieves them. Both core components have their own message queue.

The DIS has to handle different threads during the runtime. In such a multithread-application, thread safety is very important. A piece of software code is thread-safe if it functions correctly during simultaneous execution by multiple threads [15].

In the software architecture, the message queues as well as the model-components are thread-safe implemented. For this purpose the synchronization mechanism CSingleLock and CMultiLock from the Microsoft Foundation Classes (MFC) of Microsoft was used. A lock is used to ensure that only one resource respectively one critical section in a software component can be used by a thread. The other threads have to wait – due to the closed locks – until the critical sections are opened for the next thread.

Abstract Extension-Components
An abstract extension-component is an abstract class, i.e. a class not completely implemented concerning the method definitions. In an abstract class some methods are defined, other methods – so called "pure virtual functions" – are not declared. More precisely this means that from an abstract class no object can be derived. In the DIS software architecture the Model-component is extended with ModelData-components, to add, remove or read out data. The View is extended with the abstract class CGui. The Controller is extended with two abstract extension-components: The class CComm for the external communication (e.g. UMTS, CAN) and the class CLogic for the core functionality of the software system. The class CComm also has a message queue, which was described in the previous section. All abstract classes have to be extended with concrete classes.

Concrete Extension-Components
Concrete extension-components are concrete classes which extend the software system through logic functionalities (Logic-components: e.g. the functionality to organize and operate truck platoons), data (ModelData-components: e.g. to manage the convoy state) and views (GUI-components: the different elements for the graphical user interface, for instance pictures, buttons, labels etc.). To reach this goal, the concrete extension-components inherit from the abstract extensions components and complete the non-defined methods of the abstract classes. The abstract extension-components provide a quantity of code for the internal communication and processing within the software

system, so that during the development of a concrete extension component, the relevant part of the component can be focused. Moreover, this procedure secures that every concrete extension component makes the required interfaces available. Most of the concrete extension-components are implemented in Dynamic Linked Libraries.

Auxiliary Components
The DIS software architecture provides a set of auxiliary components in a static library. Static libraries, unlike dynamic ones, are not linked with the program during runtime of the application, but already during compilation. As shown in figure 3, the static library includes the interfaces of the core components (IModel, IView, IController), the abstract extension components (CLogic, CComm, CGui, CModelData) and some auxiliary classes, for instance to support Double-Buffering (CMemDC, CDC), thread safety (CMessageQueue), manipulating text strings (CString) and timer functions (CTimer).

Configurability of the Software System with XML-Files
The design of the software architecture intends the configuration of the software system with a XML configuration file. In this XML-file the configuration of the extension-components, the behavior of the software systems by user interaction, the functionality of the core components and the design of the graphical user interface is specified.

4.4 Trial Implementation

The V-Model has been subject of the development process of the KONVOI-System. The KONVOI-System had been developed according to final specification. Internal and external reviews have been implemented during the development to evaluate parts of the KONVOI-System, such as the software-architecture of the DIS. In the very beginning of the development process a preliminary hazard analysis had been taken place. A variety of systems test were carried out during the project to prove the robustness and reliability of the system. Additionally thread safety had been implemented. Finally, the KONVOI-system had to meet the IEC 61508 standard. The IEC 61508 “Functional safety of electrical/electronic/programmable electronic safety-related systems” is an international standard, which include a Failure Mode and Effects Analysis (FMEA) and a Fault Tree Analysis (FTA).

The development of the KONVOI-System is accompanied by a trial implementation in four steps. First, the whole platoon system and all of its components are completely implemented into a truck driving simulator at the RWTH Aachen University. An integrative part of the simulator is a software- (sil) and hardware-in-the-loop (hil) development environment. All test procedures (unit, integration, system and acceptance tests) as well as the optimization of the different system components were carried out in the simulator. This helps to increase the speed and quality assurance during the development. In addition to the technical development and testing the second step includes another work package: the examination of both the acceptance of the ADAS and the arising stress of the truck drivers. In a third step, the trial implementation on test tracks has taken place and the trial implementation on motorways has taken place in the second half of the project. After this, an evaluation phase on motorways in real

traffic with all four experimental trucks followed to confirm the effects determined by simulation and test drives.

For the system tests the DIS was first implemented into a truck driving simulator which was used as a test environment for the module, integration and system tests. The tests with 30 truck drivers from the freight forwarding companies of the project consortium showed a consistently positive assessment and proved the functionality as well as the reliability of the software system. After a sufficient testing of the ADAS in the driving simulator, the platoon system was tested with experimental vehicles on test tracks. Already thousands of miles were driven with the platoon system on test tracks during the trial runs. The development team worked for more than 60 days on test tracks and proved successfully that the platoon system runs absolutely safe and free from errors.

After the trial runs on test tracks were successfully finished in December 2008 the test runs on German motorways started in March 2009 (Figure 3). The test runs were completely realized on public motorways to measure the effects of the platoon system on the traffic flow, the economic efficiency and the acceptance of the truck drivers.

5 Conclusion

In this paper, an introduction into electronically coupled truck platoons with Advanced Driver Assistance Systems was given. Furthermore, the requirements for the software architecture of the Driver Information System as the HMI and information manager of the platoon system were derived and transferred into a software design. Finally, the implementation of the architecture was described in detail.

The presented software architecture fulfills all the fundamental demands for the development of interactive software systems in the automotive sector. Hereby, the architecture ensures the user interaction between the driver and the technical systems as well as the data processing between the different system components in the vehicle. This architecture guarantees a modern, flexible, extensible and easily configurable system, especially for HMI of driver information and assistance systems. Due to its interactive and adaptive characteristics, the presented architecture could be moreover seen as a generic software architecture framework for Driver Information System of platoon systems.

References

1. United Nations, Economic and Social Commission for Asia and the Pacific: Statistical Yearbook for Asia and the Pacific. United Nations Publication, Bangkok (2008)
2. Commission of the European Communities: Keep Europe moving – Sustainable mobility for our continent. Brussels (2006)
3. Commision of the European Communities: Europe at a crossroad - The need for sustainable transport. Manuscript of the European commission, Brussels (2003)
4. Savelsberg, E. (ed.): Lastenheft für elektronisch gekoppelte Lkw-Konvois, vol. 22(21). VDI, Reihe, Düsseldorf (2005)
5. Henning, K., Wallentowitz, H., Abel, D.: Das Lkw-Konvoisystem aus den Perspektiven Informations-, Fahrzeug- und Automatisierungstechnik. In: Mechatronik 2007 - Innovative Produktentwicklung. Hrsg. v. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, pp. 133–147. VDI, Düsseldorf (2007)

6. Henning, K., Preuschoff, E. (eds.): Einsatzszenarien für Fahrerassistenzsysteme im Strassengueterverkehr und deren Bewertung, vol. 12(531). VDI, Reihe, Düsseldorf (2003)
7. Friedrichs, A., Meisen, P.: A Generic Software Architecture for a Driver Information System to Organize and Operate Truck Platoons. In: Vortragsveroeffentlichung, International Conference on Heavy Vehicles (HHVT 2008), Paris, May 19-22 (2008)
8. Meisen, P., Henning, K., Seidl, T.: A Data-Mining Technique for the Planning and Organization of Truck Platoons. In: Proceedings of the International Conference on Heavy Vehicles (2008)
9. Friedrichs, A.: A Driver Information System for Truck Platoons, vol. 12(673). VDI, Reihe, Düsseldorf (2008)
10. Wietzke, J., Tran, M.: Automotive Embedded Systems. Springer, Berlin (2005)
11. Balzert, H.: Grundlagen der Informatik. Spektrum (2005)
12. Buschmann, F., Meunier, R., Rohnert, H.: Pattern-Oriented Software Architecture, A System of Patterns. John Wiley, Chichester (1998)
13. Posch, T., Birken, K., Gerdom, M.: Basiswissen Softwarearchitektur. dPunkt, Heidelberg (2007)
14. Reenskaug, T.: Thing-Model-View-Editor, An example from a planning system. In: Xerox PARC technical note (May 1979)
15. Budszuhn, F.: Visual C++-Programmierung mit den MFC. Addison-Wesley, Reading (2002)