Model Car Testbed for Development of V2X Applications

Thomas Hecker, Jens Zech, Bernd Schäufele
TU Berlin/Daimler Center for Automotive Information Technology Innovations (DCAITI), Berlin, Germany
Email: { thomas.hecker, jens.zech, bernd.schaeufele }@dcaiti.com

Ronny Gräfe, Ilja Radusch
Fraunhofer Institute for Open Communication Systems, Berlin, Germany
Email: { ronny.graefe, ilja.radusch }@fokus.fraunhofer.de

Abstract— **Testing real world Vehicle to X communication (V2X) based applications is expensive and can be dangerous with security-relevant applications. This article describes a model car test bed for research and developing automotive V2X-based applications. With this test bed, V2X-based applications can be tested quickly and efficiently. The article describes the test bed infrastructure with its components. To demonstrate how V2X applications can be investigated with this test bed an exemplary application is presented.**

Index Terms—**test bed, automotive, vehicle, model car, V2X, C2C, DSRC, ITS, indoor positioning, communication, application**

I. Introduction

Intelligent Transportation Systems (ITS) and communication between vehicles and infrastructure has the potential to further increase safety in road traffic as well as to increase driving efficiency. Many applications have been developed and field tests are ongoing or have just been finished [1], [2], [3], [4]. These applications mainly concentrate on driver information or warning about hazardous situations. The driver himself has to cope with the situation, sometimes supported by electronic automotive systems. In this scope, V2X communication provides mainly an information advantage.

Cooperative driving goes beyond this scenario. Vehicles communicate with the infrastructure or with each other in order to avoid hazardous road situations. V2X communication can be used in all phases of a communication-based assistant system: Detection, analysis and solution. While classical V2X applications concentrate on the first two phases, cooperative applications choose the solution in a mutual manner.

During specification and developing of cooperative V2X applications, different steps have to be processed. In the beginning of V2X application development, there is a theoretical analysis phase, in which the problem is discovered and a suitable solution approach using V2X technology is designed. After specification of the overall system the algorithms needed for solving the problem and the application within your V2X application framework are implemented. Finally, to verify implementation as well as efficiency of the algorithms, an intense testing phase is mandatory.

Simulation is a suitable methodology to develop and verify algorithms and applications that work for wide-area efficiency with more than a few vehicles. However, simulations take higher efforts in scenario description and set up and are highly dependent on the used driving behavior model. The two most relevant simulators SUMO and VISSIM currently do not support detailed in-depth real-world physical metrics required by some micro-scale V2X applications [14]. For these applications, real world testing is still mandatory.

However, when testing cooperative applications different challenges must be addressed. These challenges can be addressed very well with a model car test bed.

a) Scalability
Testing systems in the area of communication usually means that there is more than one autonomous system. With more complex applications, the amount of required communication units grows quickly. A model car test bed is a good way to handle resulting scaling efforts due to lower dimensions and fewer costs per unit.

b) Costs
Testing of cooperative applications with multiple cars results in high costs. Using model cars as development prototypes helps to save budget. Depending on the equipment of the model cars, a multiple number of cars can be purchased, equipped with technology and integrated into the testing environment. When testing safety related applications, there is a high risk that experimental vehicles are damaged. Using model cars, the risk of damaging is quite lower as the overall system contains less energy.

c) Safety
When testing automotive assistant systems, there is a big overhead due to safety efforts. Especially for cooperative autonomously driving vehicles these safety efforts are very high. An automotive

model car test bed is a way to reduce these safety requirements dramatically, as the potential damage a model car can cause is quite low in comparison to real cars.

d) Development cycles
A model car test bed allows very short development cycles. Changes in software and algorithms can be deployed within seconds to the cars and improvements can be seen almost immediately.

For this reasons the DCAITI developed a modular model car test bed to investigate cooperative V2X applications. The goal of this test bed is to achieve a flexible in-door solution to test V2X applications. Therefore, model cars in a scale of 1:18 are used. This scale allows small cars that are easy to handle and safe to operate in-door.

Section II gives an overview of the model cars and other connected components. Section III introduces the test bed software stack running on the model cars and server components. Section IV discusses a sample implemented cooperative V2X application. Section V shows how the model car test bed interacts with other ongoing research in the field of V2X applications. The paper closes with a conclusion and outlook section.

II. Test Bed Architecture

The test bed consists of different components designed in a modular approach (see figure 1).

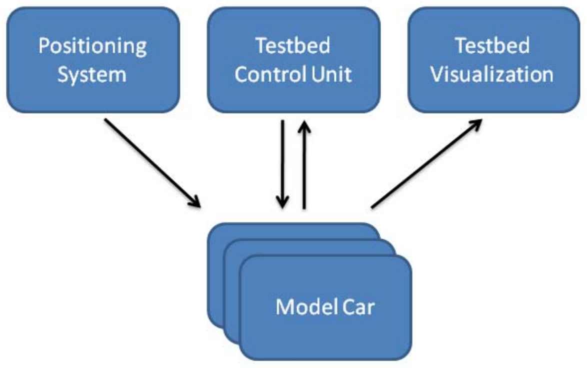

Figure 1. Test bed components overview

A. Model Car

The model cars consist of a classical model car subsystem combined with special hardware (see figure 2). This hardware is necessary to attach different kinds of sensors as well as a WiFi communication unit. As processing unit, a commercially available single board computer was integrated. This computer runs a Windows CE operating system with an automotive V2X software stack (see section III). A specialized sensor board is connected to this single board computer. It provides various hardware interfaces, like I2C, UART, and SPI, to connect different sensors.

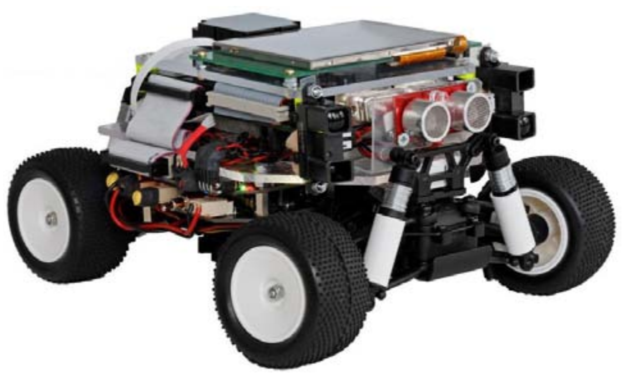

Figure 2. Model Car with infrared and ultrasonic sensors

For middle-field environment analysis, a standard ultra sonic sensor is integrated. It allows the model car to detect obstacles in a range of 5 cm to 100 cm. However, due to the characteristics of ultrasonic this is a wide angle sensor that does not allow detecting the angle at that the obstacle is located.

To handle this drawback there are two infrared spots mounted at the vehicle's top corners. Infrared spots are characterized by a middle-range narrow beam. These spots observe 20 degrees to each side and allow the car to compute an obstacle direction.

An integrated electronic brushed motor with low rotational speed accelerates the model car to a maximum speed of 1.5 m/s. Considering the scale of 1:18, this corresponds to 100 km/h or 62 mph. This speed range allows running cooperative driving scenarios from low-speed up to highway scenarios.

A lithium Polymer (LiPo) accumulator provides energy to operate more than 2.5 hours with all features activated. This battery pack is removable so that it can be exchanged quickly.

B. The Course

The cars run on a 4 m x 4 m quadratic zone, on which different courses can be defined. This dimension results from the positioning system (see section II C). If a larger testing area is needed, it is possible to adapt the components of the positioning system and increase the dimensions. A 16m² installation is sufficient to evaluate compact scenarios with moderate vehicle speeds (see section IV). If evaluating more complex scenarios, it's recommended to enlarge the course.

C. Positioning System

Positioning in vehicular test beds is an important topic for two reasons. First, from an individual point of view every communication and application unit needs to know where it is. This corresponds to GPS in real world applications. Vehicular applications that build on local dynamic data need positioning information for their own application logic as well as communication logic like geo-routing algorithms. The precision of location based applications corresponds directly to the quality of positioning information [5].

Second, a positioning system is needed to control the test bed itself and visualize the current status of the test bed. When model cars are controlled autonomously or semi-autonomously, the driving algorithms require very high accuracy positioning information. One great advantage of a vehicular test bed is repeatability. A scenario can be replayed with different parameters in order to evaluate alternations. The quality of replays is directly linked to high accuracy positioning information.

On that score, the main requirements to a vehicular testbed positioning system are high absolute precision and high dynamic behavior resulting in low detection delay.

As commercially available positioning systems did not fit these requirements, a camera based positioning system has been developed especially for this test bed. Every vehicle transports a 4-square RGB color code on its top. A 50 fps color camera that is fixed on the ceiling captures the scenery and sends a frame stream to a computing host. The host analyzes the frame stream for color codes and tracks movements. An OpenCV [6] based image analyzing algorithm detects the color codes and computes the real position of the vehicles based on calibration matrices. The computed position, orientation, and speed information is broadcasted to all test bed entities.

With a 4-square RGB color code (see figure 3) it is possible to distinguish 27 different model cars on the test bed.

The computation algorithm analyzes the stream with 45 fps. Thus, every 22 ms a new position is broadcasted. The cars and TCU receive this position with a delay of about 10 ms. Due to multi-core software support, these computation rates can easily be increased in the future.

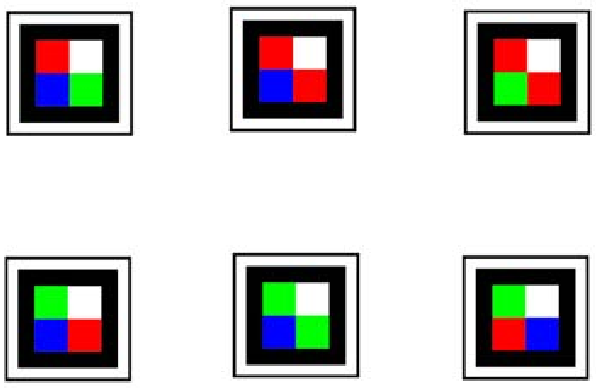
Figure 3. Six different color code examples

The current installation covers a test bed course of 4 m x 4 m. The camera is fixed at 3.5 m above the floor and views straight from the top.

With this installation an absolute precision of 2mm and an orientation deviation of 1° can be achieved. Regarding a scale of 1:18 this corresponds to a real world position detection precision of 3.6cm. The quality of positioning can be further improved with a high resolution camera in combination with a more powerful computation unit.

D. Test Bed Control Unit

The Test Bed Control Unit (TCU) is the central component that manages the test bed and scenarios. It offers central infrastructure services, such as car management, car configuration and car control.

With the car management, available model cars are detected and can be registered to participate in a scenario. For example, the cars' battery status is presented.

Car configuration offers the functionality to change the car parameters and settings. Every scenario has different requirements for the cars. Running applications can be deactivated and activated on the car and the sensor behavior can be configured. The cars have USB ports as well as wireless network connection. This allows for quick and simple deployment of new configurations. Scenario configuration profiles can simply be saved for loading them into the cars at the next test run.

When the cars are configured, scenarios can be started. Depending on the kind of the scenario the cars can be steered manually or controlled autonomously by the system. For individual driving, an integrated hardware steering wheel can be used. With this wheel all cars can be controlled individually. If manual driving is not desired, an autonomous driving algorithm is provided. This enables precise scenario replays as well as rule-based autonomous driving as used in the following-scenarios.

During a test run the cars produce log data. Individual sensor values and communication data are recorded by the TCU for scenario analysis afterwards.

E. Visualization Unit

The Visualization Unit is used to present the test bed status on a Graphical User Interface (GUI) to the users. It shows which cars are involved in a scenario and the cars' status (see figure 4). Furthermore, it visualizes, what is not visible for the human eye: communication. On the Visualization GUI one can watch which model cars are communicating with each other and the type of communication. The visualization can be filtered with a level of detail so that the relevant parts of a scenario can be observed better.

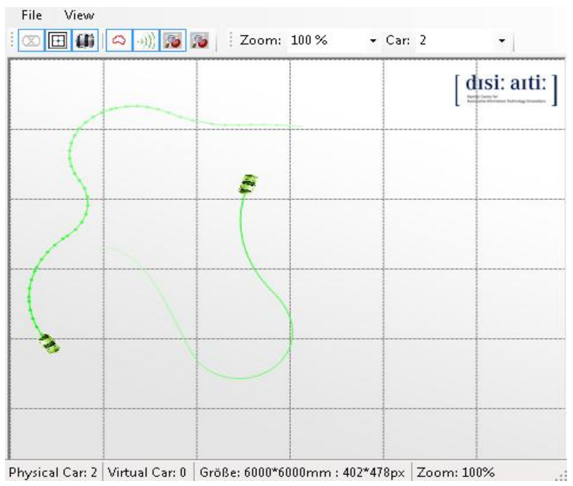

Figure 4. Test Bed Visualization Unit

F. Communication System / Protocols

In this model car test bed, different communication technologies and protocols are used, which are necessary to ensure the communication between the model cars. For the implementation part of these protocols, proposed standards of the Car-2-Car Communication Consortium (C2C-CC) [7] and the European Telecommunications Standards Institute (ETSI) were investigated and applied.

For communication in Intelligent Transportation Systems (ITS) the European Telecommunications Standards Institute (ETSI) and the US Federal Communications Commission (FCC) both reserved frequency bands in the 5.9GHz spectrum. Based on these allocations, they built the communication standards ETSI ES 202 663[15] and IEEE 802.11p[16] specifically to support automotive requirements. These are high Doppler shifts and high dynamically changing multipath propagation due to high velocities up to 200kph. However, in order to keep the components in our test bed simple, small and cost-efficient, we use standard 802.11b consumer WiFi. As communication channel characteristics are an important aspect in V2X technology and need to be considered in this test bed, too, the communication module supports integration of channel modelling to integrate realistic radio communication behaviour.

To expand the horizon of the model cars, which could be disturbed by obstacles, the model cars must have different information about the environment, e.g. the position of other model cars. To gather necessary information and data, the concept of environment tables is used. An environment table holds all information and data of other model cars within a given radius. The environment table is a hash table of Cooperative Awareness Messages (CAM), i.e. for each node, that a car is aware of, one message is stored. Figure 5 depicts the implementation of the ETSI standard [26] in the model car test bed. As the CAM contain a timestamp, each application knows if the information is still valuable for it. These messages are sent via a User Datagram Protocol (UDP) broadcast to other model cars in the test bed.

With the usage of the environment tables, it is possible to perform different analyses of the environment. The results and information of these analyses are very useful in a variety of V2X applications.

Each V2X packet contains a protocol type and a protocol version. These information elements are used to identify the message as a C2C-CC protocol message and the version of the underlying protocol. The Originator object contains information which identifies the originating node and is of the type NetOHEAD.

Figure 5 depicts also that two Originator objects exist in the V2X packet. This is necessary, if a V2X packet is forwarded by a model car. In this case, the Originator in the root of the V2X packet should be replaced with own information about the forwarding node (e.g. NodeType, NodeSpeed, Position, etc.), and a new ProtocolMessage structure should be created. The information and data of the origin node are still present in the Originator object of the ProtocolMessage. This ensures that the forwarding node can also provide own information and data. The receiving node can deduce from this that the packet was forwarded. The ProtocolMessage object is the payload of the V2X packet and contains a CAM or a Decentralized Environmental Notification Message (DENM) message. The payload of the CAM message is the MessageBody. This CAM payload can be differentiated between VehicleAwarenessData and RSUAwarenessData. In the example implementation of the model car test bed, the VehicleAwarenessData contains information about the VehicleType, the length and the width of a model car and a TaggedList. This TaggedList contains further relevant information, e.g. sensor data and power data. This also applies to the implementation of the RSUAwarenessData, which consists of an RSUType and a TaggedList.

There are two approaches to deliver the environment information to the other model cars in the test bed. On the one hand it is possible to send the V2X packets in a static interval. This interval is set to 250 milliseconds in the model car test bed. Investigations have shown that these values are sufficient for basic applications. On the other hand it is possible to send the V2X packets in a dynamic interval depending on the current speed of the model car. In the model car test bed, a minimum interval of 3ms and a maximum interval of 111ms are used for the dynamic approach. Depending on the investigations and use cases, it is possible to switch between these two approaches.

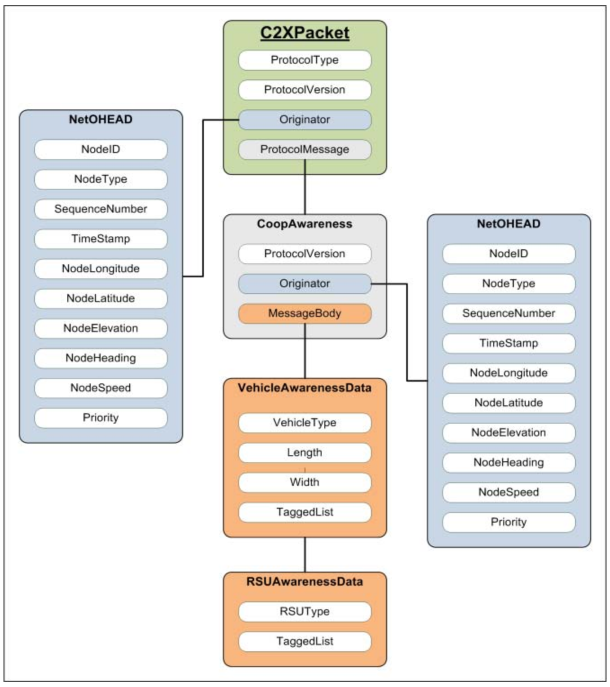

Figure 5. Implementation of a Cooperative Awareness Message

III. Software Architecture

The test bed software stack is divided into three different layers. Figure 6 depicts the Functional Design Architecture (FDA), which gives an overview of the layers and components.

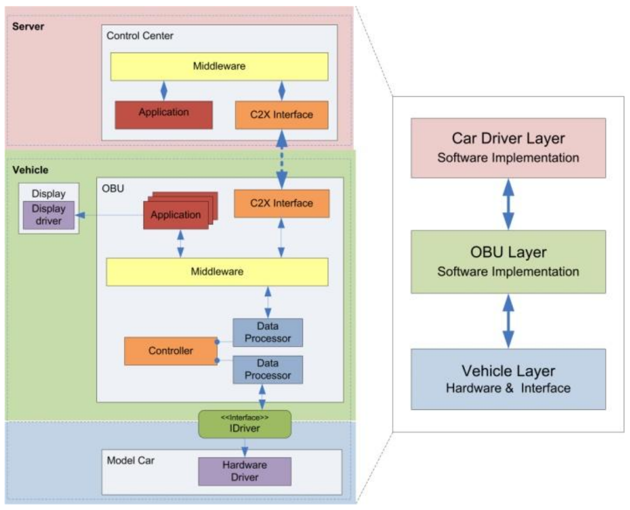

Figure 6. Functional design architecture of the test bed software

The first layer, the “Car Driver Layer” deals with different components running on the TCU. These can be applications (e.g. the control of a model car with a steering-wheel), tools for development or visualization.

The second layer, the “On Board Unit (OBU) Layer”, contains the software on the mobile devices such as the vehicle control, sensor monitoring and network connection.

The third layer, the “Vehicle Layer“, is connected with the OBU and provides the interface for the connection of external hardware. This is mainly used for the request of data from different sensors and for the control of several actors.

Furthermore, these layers contain different elements, e.g. controller, hardware driver, interfaces, middleware, applications and data processors. These elements will be described in the following.

A. Controller

The main components of the architecture are the controllers. These controllers process different autonomous control and data processing tasks. Controllers receive data, process it and return after processing. This processing takes place at fixed intervals of 20, 10 or 5 Hz.

Data to process can be received from and saved to the middleware and hardware drivers. For example the Sensor Controller retrieves its external sensor data via a hardware driver. It then generates a complete map of detected obstacles and sends the result to the middleware. In contrast, the Drive Controller receives control data (velocity and steering angle) from the middleware, verifies the permissibility (e.g. correct direction of motion) and calls a hardware driver. This driver then accomplishes control of velocity and generation of Pulse Width Modulation (PWM) triggering.

The controllers work independently of each other. That way, it is possible to deactivate a single controller or to switch between different implementations. To achieve individual configuration of controllers, functionality is implemented within separated controllers.

B. Driver

Hardware drivers are used to access external hardware. The driver mostly implements direct access to the hardware components and links them to the software. Additionally, this task can be solved virtually without hardware.

The following example of a sensor driver describes the task in detail. A driver is communicating via Inter-Integrated Circuit (I^2C) with infrared sensors and requests a measurement voltage. The driver transforms input values into distance values with a reference matrix. These values can be directly requested from an interface. The driver has a direct hardware component. The same interface could be implemented by a virtual driver that calculates the distance to a wall depending on the current position. Thus, it is also possible to provide sensor data without using real hardware, e.g. with simulations.

C. Interfaces

To easily replace hardware and drivers, interfaces are used between these parts. Interfaces define specific functionality. Thanks to these interfaces, the implementation is hidden to the other components. This allows a fast and easy change of implementations (e.g. via a configuration file).

The following example of the *IMotionFeedback* Interface illustrates this. The interface calculates the motion (steering angle and velocity) of the model car. This can be achieved by an infrared positioning system, camera positioning or GPS. Each alternative is realized in a separate driver class which implements the interface *IMotionFeedback*. To switch between these different implementations, the Motion Feedback controller can be configured. The controller itself does not know about the underlying implementation.

D. Middleware

The middleware uncouples the various controllers and the applications so that there are no dependencies between them.

In recent vehicles, different assistant systems are connected. For example, the combination of front light and steering results in a adaptive curve headlight. This connection is mostly realized by bus systems (e.g. Controller Area Network (CAN) bus). To emulate this system on the model car, the middleware has been developed. The different systems (controllers) can communicate via the middleware.

The middleware implements the observer software design pattern [10], i.e. all data is passed through the middleware. The middleware provides the means of channels that are realized as queues. Applications that want to retrieve certain data have to subscribe to the corresponding channel on the middleware. Applications that want to send data have to publish them on a channel on the middleware. The distribution and notification of data to the registered receivers is performed by the router component in the middleware. Thereby, not all receivers get all data that is published on the middleware. Rather, they are able to register only to channels that they are really interested in. The main benefit of this middleware concept is that sender and receiver do not need to know of each other.

E. Data processor

As already described, later developed applications require access into the control of existing processes. For example, „Adaptive Cruise Control“ and „Collision Avoidance” basically interfere with the vehicle control, but, related to further use cases, it can concern all other areas. For both functions, the driver specifies the velocity which is transformed by the vehicle. If one of the functions recognizes that the distance to the car or obstacle in front is too small and a collision impends, the system automatically overwrites the input. The possibility of intervention into the control is an important requirement that is met by data processors.

Each logically independent part is implemented by an own controller (see section “Controller”). For external applications it is only necessary to implement the *IDataProcessor* interface which receives data via API-Calls, TCP or MessageQueues.

The usage and integration of data processors can be configured via XML.

In the example shown in figure 7, three data processors are configured. Before execution of *DriveControlStation* (*TIME=“pre“*) the processors *SensorOverride* and *SpeedOverride* should be performed. As *SensorOverride* is an emergency handling and as the processor has to modify the control data, no other processor must be executed (*DISCARD="true"*). After determination of execution time (*TIME=“pre“*), a Kalman-filter will be performed.

Figure 8 illustrates a general overview of the final implementation of the software stack that runs on a model car.

```xml
<?xml version="1.0">
<config>
  <DataProcessors>
    <Processor TYPE="DriveControlStation"
               TIME="PRE" DISCARD="true">
      de.tuberlin.Applications.SensorOverride
    </Processor>
    <Processor TYPE="DriveControlStation"
               TIME="PRE" DISCARD="false">
      de.tuberlin.ACC.SpeedOverride
    </Processor>
    <Processor TYPE="PositioningControl"
               TIME="POST" DISCARD="false">
      de.tuberlin.PositionFilter.Kalmanfilter
    </Processor>
  </DataProcessors>
</config>
```

Figure 7. Data processor configuration XML file

IV. Sample Scenario – Crossing Assistant

This section shows exemplary how to develop and verify V2X-based applications with this test bed. It describes a sample crossing assistant application. This application supports drivers at intersections by preventing

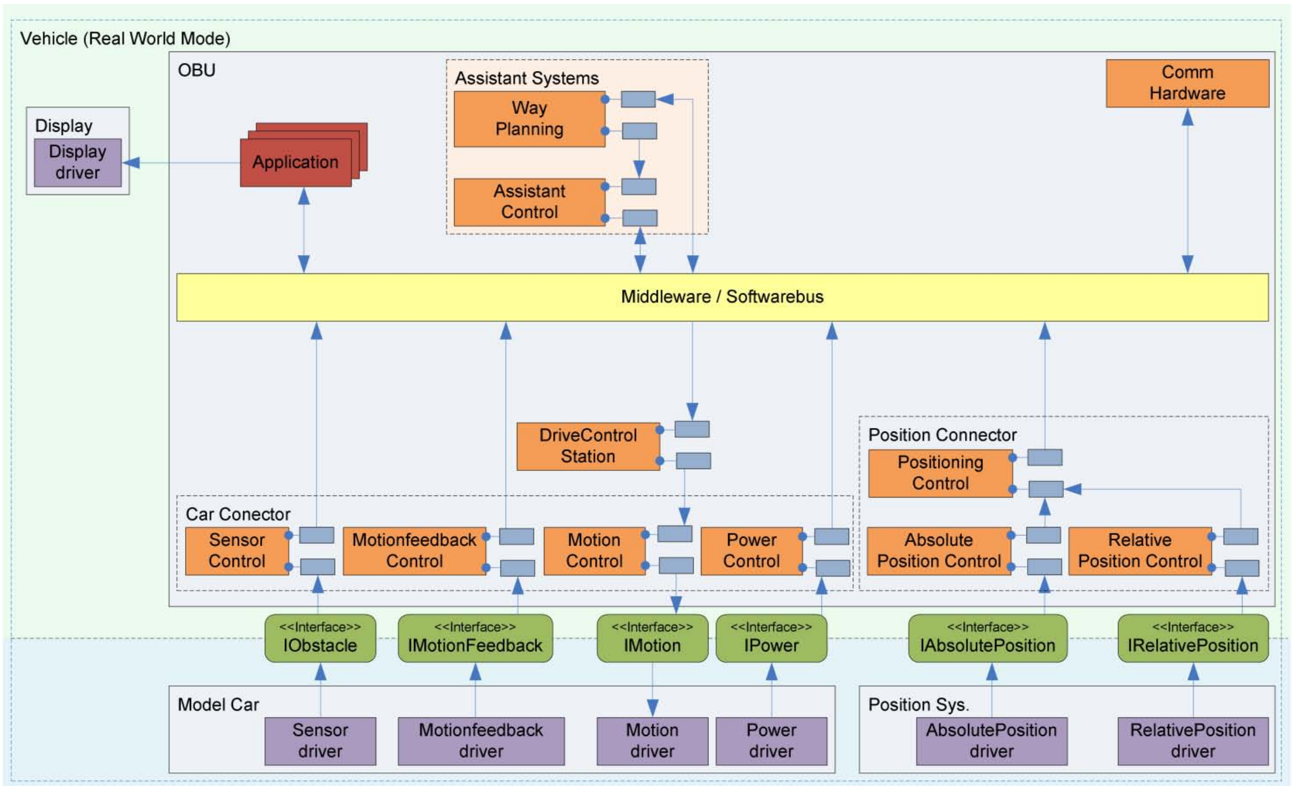

Figure 8. Functional design architecture - car detail

collisions. First, different scenarios and solutions are described. Then the implemented solution is explained and details about the implementation are presented, before the section closes with a validation of the scenario.

A. Crossing Assistant architecture

One of the major fields of application of V2X communication is the improvement of traffic safety, because information about other cars and road conditions can be used to detect hazardous situations. This information can be used either to warn drivers of events that they might miss by their perception or automatically take corrective actions. At intersections without light signal systems, cars from different directions cross each other. It is up to the drivers to identify approaching cars, estimate their velocity and apply right-of-way traffic rules in very little time. Therefore, intersections are an accident hotspot, accounting for 30 to 60 percent of all injury accidents [8].

In order to demonstrate a support for intersections, a crossing assistant for the test bed cars has been implemented. There are several possible scenarios for a crossing assistant for cars that are equipped with communication devices. Moreover, a stationary road side unit can be involved in the collision detection and notification. Thereby, collisions that would occur if the cars had no means of communication can be avoided.

The model cars permanently know about their position within the test bed, their speed and their orientation. This information is regularly broadcasted by all cars with CAM. Moreover, the communication devices can be used to directly warn each other of possible collisions.

In the first scenario, all cars broadcast their position, speed and orientation with CAM. Each car permanently calculates, if the cars are in danger of collision. As soon as in one car the danger of an upcoming collision is detected, the driver is warned and a warning message is sent to the other car, so that its driver can be warned, too. It is the driver's duty to react to this warning. This scenario nowadays is very feasible to install in real world cars. Apart from the communication and the warnings in the on-board information system of the car, no technical upgrades are necessary. There is no direct interaction with the chassis and no liability risk for the manufacturer, because the driver is always controlling the car. In order to make this solution more useful, an emergency stop can be introduced. As in commercially available pre-crash systems, an emergency brake could be started 0.6 seconds before the collision [13]. Even though that might not always be enough to stop the car completely, the car can be slowed down sufficiently to avoid serious injuries or fatalities.

Even though such a system is the most feasible, the approach that is implemented for the test bed model cars, goes beyond this. On the one hand, the crossing assistant shall avoid collisions, but on the other hand, it shall be avoided that a car stops unnecessarily.

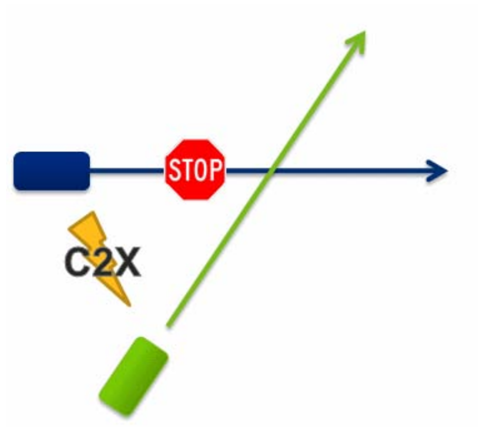

Figure 9. Intersection scenario

Therefore, if the danger of a collision is recognized, only one of the two involved cars shall stop. Moreover, the test bed cars can drive autonomously. Hence, the crossing assistant shall also work without human interaction. Thus, the model cars must be able to decide by themselves, which car to stop. There are several ways to realize this behaviour. Both cars can determine which car should stop upon a shared algorithm that will lead to the same result on both cars. Another option is to select one of the vehicles as coordinator of the situation and let this car decide, which car is supposed to stop. However, there is a possible security threat, if one car is the coordinator and is allowed to make another car stop, because security measurements might be by-passed. Hence, this option was abolished.

The decision, which car is stopped, is based on a common rule. Therefore, the "priority to the right rule" is applied, which is common in most European countries, i.e. the car that is left of the other must stop. Figure 9 shows this scenario. The car coming from the left has to stop after the negotiation by V2V communication. However, which car is on the left and which one on the right can only be calculated with help of the position and the orientation of the cars. As normally the information is the same on both cars, they must come to the same result. As a CAM might be delayed, the cars must perform an emergency brake, in case that they are not sure, whether the other car stops. Hence, the car that stops sends a message to the other car informing it that it has come to this conclusion. If such a message is not received 0.6 seconds before a possible collision, the other car stops with an emergency brake.

The CAM contain the absolute position of the model car in the test bed, its speed and its orientation. When a car receives a new message, it processes this message. Firstly, the direct distance between the two cars is calculated. If the distance is above 2.5 meters, no further calculations are made based on this message. When the direct distance between the two cars is below 2.5 meters, the situation is examined, i.e. it is checked whether the two cars are in danger of collision.

For simplification, it is assumed that the model cars are moving on straight lines. With the position and the heading, the intersection point of the cars can be calculated. As the cars are approximately 20 centimetres wide, the intersection point is calculated for both edges of the cars. Moreover, due to information about the orientation, it can also be determined, whether the cars are moving towards the intersection point or away from it. If at least one of the cars is moving away from the intersection point, the processing is interrupted. Other than that, it is checked how far the cars are from the crossing point.

If the cars are within a certain distance, the car coming from the left has to stop. There are two implemented options to determine this distance. The first option is to use a static value of one meter. The second option is to estimate the way that the cars need to stop. For this, the well-known rule of thumb for the braking distance $\left(\frac{speed\ \ in\ km/h}{10}\right)^2$ meters is used. In case that both cars are within their respective braking distance, they check which car is coming from the right. For this calculation, the headings of the cars are used. The right car is allowed to drive, whereas the left car is supposed to stop. The left car stops and sends an acknowledgement message to the other car to signal that it is aware of the situation. Thus, the right car can pass the intersection.

Furthermore, upon the current speed and the intersection point, it is calculated how much time is left until the cars crash. If the right car has less than 0.6 seconds to a possible impact and it has not yet received an acknowledgement message from the other car, it performs an emergency break.

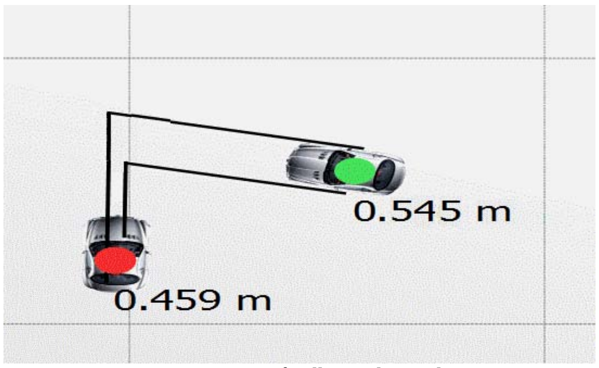

Figure 10. Danger of collision detected

Figure 10 shows a screenshot from the visualization of the implementation of the crossing assistant. It depicts a typical scenario. The cars are close enough to the intersection and hence start to calculate the collision point. The car on the left has reasoned that it needs to stop, whereas the right car can continue. This is symbolized by the coloured dots.

B. Crossing Assistant Validation

For the validation of the implementation and the used algorithms, a series of three different tests was carried out. These tests had always the same setup. Each test was repeated 50 times. Thus, in total 150 tests were made. Both cars started at a distance of 2 meters to the intersection. The position detection system had a refresh rate of 5 Hz. Depending on the test series, the cars had different settings of ranges for stopping. For the first test series with static settings, a stop range of one meter and a static setting for the CAM sending interval of 250 milliseconds was used. For the second test series, also a static setting for the stop range of one meter was used. The difference from the first test series consists in the dynamic transmission interval. The model cars were calculating their CAM send interval depending on their current speed:

$$\Delta_s = velocity * (\Delta_{min} - \Delta_{max}) + \Delta_{max}$$

The third test series had a static CAM send interval of 250 milliseconds. The model cars calculated their stop range and their stop notification range depending on their current speed with the rule of thumb mentioned in section IV A.

Each test series had a 100% positive result. In these series no collisions or errors occurred. Thus, a proper implementation of the given standards and the application can be concluded.

V. Related work

This model car test bed aims to simplify research and development of V2X applications. An in-depth introduction into this technology and suitable applications gives [19]. An overall ITS communication architecture is standardized by ETSI[21]. It describes communication entities, systems and sub-systems needed for complete ITS roll out.

Currently, there are many national or EU Field Operational Tests (FOT) on-going to investigate V2X technology. Some of them focus on architecture for ITS, others develop and test V2X applications in real world scenarios.

For example, the German funded project SIM-TD investigates 21 V2X applications in the scope of road safety, vehicle safety, road efficiency and vehicle efficiency. A description of these applications can be found in [12] and [22]. The project plans real world testing with up to 400 vehicles, 100 ITS Roadside Stations as well as traffic simulation and experiments in driving simulators.

The project SAFESPOT [9] is an integrated research project co-funded by the European Commission. It focuses on collision prevention and investigates 10 use cases like intersection safety and lane changing.

Contemporaneously to these FOTs standardization activities are ongoing. The ETSI defines a Basic Set of Applications which are considered as deployable into market few years after standardization has finished [17]. These applications use CAM messages implemented in this test bed (compare section IIF) for environmental awareness. For transferring application specific content ETSI standardizes Decentralized Environmental Notification Messages (DENM) [18].

Another method for investigating V2X technology is simulation. However, simulating V2X applications needs to combine networking models with communication models and traffic simulators. One solution to achieve this is presented in [20]. It combines multiple well-established simulators of each topic into an integrated system that allows V2X simulation.

A similar approach for a collision avoidance assistant in a model car test bed is presented in [23]. However, in this work the cars broadcast their planned trajectories and adapt them in order to avoid collisions. Therefore, this solution only works with cars that calculate trajectories, but not in a mixed environment with cars that drive autonomously and with cars that are steered manually.

Mobile Emulab [24] is a test bed with car-like robots. In contrast to our test bed, the focus of this project is not on vehicular applications, but on sensor networks and communication channels. Hence, no message format that is similar to real-world V2X standards is used, such as CAM or DENM. However, the positioning of the cars is also camera-based.

The work in [25] also presents a model car test bed. Like in our approach, camera-based positioning is used. The software and hardware architecture is also distributed, i.e. the central unit, the positioning system and the micro controller board are divided into separate units. The cars used in this test bed are much smaller than our ones, the scale of the cars is 1/64, whereas the scale of our cars is 1/18. As a consequence, the steering and velocity control is limited.

VI. Conclusion and Outlook

In this paper, a model car test bed that facilitates the development and testing of V2X applications has been presented. With this test bed, V2X application can be developed faster and less expensive than with real-world cars.

The test bed is based on a distributed architecture. The major components are the positioning system, the central control unit, the visualization unit and the model cars themselves. Even though this requires increased communication and coordination effort, each specialized component is independent and self-contained and thus easier to maintain.

The software is built with a three-layer plug-in architecture. The layer architecture of the software, just as the distribution of the systems, enables better maintenance, because each layer has a certain purpose and adaptations can be applied more easily. The plug-in architecture allows for rapid development, what is essential for research purposes. New V2X applications can be implemented and new sensor and hardware drivers can be integrated in an easy and quick way.

One example application has been shown in this paper. The crossing assistant helps to avoid collisions at intersections. If two cars are approaching a crossing and might collide, one of the cars is forced to stop in time.

Even though the test bed already provides an environment to test V2X applications, further improvements are planned. One point of extension is the communication channel between the cars. New models that simulate communication issues, e.g. packet loss, duplication of packets etc., can be introduced. This way, it can be tested, how the message problems influence the reliability of applications and how applications have to be adapted.

Furthermore, an Decentralized Environmental Notification basic service conform to ETSI standard will be available in the near future. That enables V2X applications to communicate totally standard conform within the test bed.

Another way to improve the test bed is to increase its size. So far, a size of 4x4m is used. With a larger area, more complex scenarios involving more cars can be tested. With the improved, more realistic infrastructure, the results of applications can be compared to real world tests.

Furthermore, while the infrastructure is improved and increased, also new applications will be developed. This includes also use cases that already have been implemented for real world research projects, e.g. emergency vehicle warning or green light optimal speed [11], [12]. These use cases can be run more easily and less costly in the test bed than in the real world. The existing crossing assistant application will also be improved. First of all, better algorithms, can be used to detect possible collisions. For example, different trajectory models can be considered. Moreover, an extension of the assistant for scenarios that involve more than one car is planned.

References

[1] INTERSAFE. http://www.prevent-ip.org/intersafe
[2] Cooperative Intersection Collision Avoidance Systems. http://www.its.dot.gov/cicas/
[3] PRE - DRIVE C2X, http://www.pre-drive-c2x.eu/
[4] SimTD, http://www.simtd.de
[5] Peter Knapik, Diploma thesis: "Multi Sensor Positioning for Scaled Real World Test Environments", Technische Universität Berlin, Berlin, 2009
[6] G. Bradski and A. Kaehler: Learning OpenCV, “Computer Vision with the OpenCV Library”, *O'Reilly, Cambridge, MA,* 2008
[7] Car-to-Car Communication Consortium, http://www.car-to-car.org
[8] B. Roessler and K. Ch. Fuerstenberg, “Intersection Safety with V2I Communication in the EC Project INTERSAFE”, 3rd International Workshop on Vehicle-to-Vehicle Communications, Istanbul, Turkey, http://www.ibeo-as.com/images/stories/pdf/4_v2vcom_07_communication_ibeo.pdf, 2007
[9] “Specifications for Intelligent Cooperative Intersection Safety”, SAFESPOT Deliverable D5.3.3, Stuttgart 2008
[10] E. Gamma, R. Helm, R. Johnson and J. Vlissides, “Design Patterns. Elements of Reusable Object-Oriented Software”, *Addison-Wesley Longman, Amsterdam,* 1994
[11] W. Enkelmann et al. “Detailed description of selected use-cases and corresponding technical requirements”, *PRE-DRIVE C2X Deliverable D4.1, Sindelfingen, 2008*
[12] K. Naab “Beschreibung der C2X-Funktionen“, *simTD Deliverable D11.1, Sindelfingen, 2009*
[13] M. Fach, D. Ockel “Evaluation Methods for the Effectiveness of Active Safety Systems with respect to Real World Accident Analysis”, *International Technical Conference on the Enhanced Safety of Vehicles, Stuttgart, 2009*

[14] D. Rieck, B. Schünemann, I. Radusch, C. Meinel „Efficient Traffic Simulator Coupling in a Distributed V2X Simulation Environment“ in *SIMUTools '10: Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Torremolinos, Malaga, Spain, 2010. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, pp. 1-9*

[15] ETSI: *Intelligent Transport Systems (ITS); European profile standard for the physical and medium access control layer of Intelligent Transport Systems operating in the 5 GHz frequency band (ETSI ES 202 663 V1.1.0).* European Telecommunications Standards Institute, January 2010

[16] IEEE: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments (IEEE Std 802.11p-2010).* Institute of Electrical and Electronics Engineers, Inc., July 2010

[17] ETSI: *Intelligent Transport Systems (ITS); Vehicular Communication, Basic Set of Applications, Part 1: Functional Requirements (TS 102 637-1 V2.0.4).* European Telecommunications Standards Institute, June 2010

[18] ETSI: *Intelligent Transport Systems (ITS); Vehicular Communication, Basic Set of Applications, Part 3: Specifications of Decentralized Environmental Notification Basic Service (TS 102 637-3 V2.1.1).* European Telecommunications Standards Institute, April 2010

[19] *R. Popescu-Zeletin, I. Radusch, M. A. Rigani*, “Vehicular-2-X Communication: State-of-the-Art and Research in Mobile Vehicular Ad hoc Networks”, Springer, Berlin, 2010

[20] T. Queck, B. Schünemann, I. Radusch, and C. Meinel, “Realistic simulation of v2x communication scenarios,” in *APSCC ’08: Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference.* Washington, DC, USA: IEEE Computer Society, 2008, pp. 1623–1627.

[21] ETSI: *Intelligent Transport Systems (ITS); Communications Architecture (EN 302 665 V1.0.0).* European Telecommunications Standards Institute, March 2010

[22] C. Paßmann, G. Schaaf, K. Naab “Ausgewählte Funktionen“, simTD Deliverable D11.2, Sindelfingen, 2009

[23] W. Sheng, Q. Yang, and Y. Guo, "Cooperative Driving based on Inter-vehicle Communications: Experimental Platform and Algorithm," *IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5073-5078, October 2006*

[24] D. Johnson, T. Stack, R. Fish, D. Flickinger, L. Stoller, R. Ricci, J. Lepreau, “Mobile Emulab: A Robotic Wireless and Sensor Network Testbed”, i*n Proceedings of Infocom, 2006*

[25] C. Hsieh, Y. Chuang, Y. Huang, K. Leung, A. Bertozzi, E. Frazzoli “An economical micro-car testbed for validation of cooperative control strategies”, *Proceedings of the 2006 American Control Conference*

[26] ETSI: *Intelligent Transport Systems (ITS); Vehicular Communication, Basic Set of Applications, Part 2: Specification of Cooperative Awareness Basic Service (TS 102 637-2 V1.1.1).* European Telecommunications Standards Institute, April 2010

Thomas Hecker is Research Fellow at the Daimler Center for Automotive IT Innovations (DCAITI) of the Technische Universität Berlin. He received a Dipl.-Ing. in Electrical Engineering from the Technische Universität Darmstadt in 2007. His Research topics include Vehicle to X Communication, Automotive Infotainment and Mobile Services.

Ronny Gräfe was born in Gera, Germany in 1981. He holds a Dipl.-Ing. in Computer Engineering from the University for Applied Science of Mittweida, Germany with specific emphasis on multimedia technologies since April 2009. Currently he is working in the competence centre for Automotive Services and Communication Technologies at the Fraunhofer institute for open communication systems in Berlin. Ronny Gräfe received the Carl-Georg-Weitzel award for his investigations in the field of Next Generation Networks and communication technologies at the University for Applied Science of Mittweida, Germany.

Bernd Schäufele was born in Göppingen, Germany in 1983. He received a M.Sc. degree in Software Systems Engineering from the Hasso Plattner Institue in Potsdam, Germany in May 2009. Currently he is working at the Daimler Center for Automotive IT Innovations (DCAITI) of the Technical University of Berlin. His major research interest is autonomous driving.

Jens Zech was born in Berlin, Germany in 1977. He holds a Dipl.-Ing. in Computer Engineering from the Technische Universität Berlin, Germany since December 2008. Currently he is working at the Daimler Center for Automotive IT Innovations (DCAITI) of the Technische Universität Berlin. His major research is managing of a test bed and autonomous driving.

Dr. Ilja Radusch is head of the competence centre for Automotive Services and Communication Technologies (ASCT) at the Fraunhofer-Institute FOKUS. He is actively working in the field of Car-2-Car Communication, Sensor and Ad-hoc Networks, and Context-aware Services. He is leading various national and international joint research efforts including projects like simTD, PRE-DRIVE C2X. Furthermore, he is giving several lecture courses at the Technische Universität Berlin on vehicular communication.