F2010-C-123

SIMULATION ENVIRONMENT FOR THE DEVELOPMENT OF PREDICTIVE SAFETY SYSTEMS

[1]Dirndorfer, Tobias[*], [1]Roth, Erwin, [1]Neumann-Cosel, Kilian von,
[2]Weiss, Christian, [1]Knoll, Alois
[1]TU München, Germany, [2]Audi AG, Germany

KEYWORDS – simulation, predictive safety, pre-collision-phase, pre-crash-scenarios, test and optimization

The continuously growing vehicle density on European roads leads to a higher risk for traffic participants to be involved in accidents. In order to mitigate this risk both for vehicle occupants as well as unprotected traffic participants, the automotive industry seeks for solutions in the intelligent combination of active and passive safety systems towards an integral approach.

Safety applications like an active emergency brake that can reduce the consequences of an accident or even avoid a crash completely and predictive passive safety systems that feature optimized deployment characteristics of restraint systems (airbags, belt pretensioners) both depend on anticipatory sensor signals concerning the vehicle environment in the pre-collision-phase as a basis for their crash prediction algorithms. The development, test and validation of predictive safety systems require efficient simulation-based methods in order to be able to achieve a large test space coverage and to generate reproducible sensor signals for the respective test scenarios.

In this paper a highly configurable and flexible method for the simulation-based development and testing of predictive safety algorithms is presented. The method is based on a synchronized data connection between MATLAB/Simulink/Stateflow and “Virtual Test Drive” (VTD).

MATLAB/Simulink/Stateflow allows the intuitive model-based rapid prototyping of safety function algorithms using predictive sensor information as input data. These algorithms can easily be transformed into ANSI/ISO C-compliant code for diverse hardware targets e.g. by the Real-Time Workshop and tested in an identical form in the vehicle after the optimization and validation process in the simulation environment.

VTD consists of the components driving simulation, traffic simulation, visualization and sensor models, which supply the algorithms running in MATLAB/Simulink/Stateflow with the required sensor input data concerning the virtual vehicle environment.

This combination offers the possibility to easily implement a large variety of relevant traffic situations and environmental conditions in order to test, optimize and validate the predictive safety systems under repeatable conditions. Simulation data can be accessed via interfaces for an on-/offline data evaluation and visualisation by independent analysis applications. The complete simulation environment can be distributed over several computers connected via IP-network and executed in real-time or on the basis of a common simulation time.

The simulation environment was exemplarily used to test and optimize an anticipatory algorithm characterizing an imminent collision by the prediction of representative collision parameters. The testing was done on the basis of a huge pool of characteristic pre-crash-scenarios statistically representing the GIDAS database (German In-Depth Accident Study).

VIRTUAL TEST DRIVE – An integrated test environment for the automotive industry

Audi and Volkswagen are developing an integrated and highly modular computer based simulation system called “Virtual Test Drive” (VTD) (1). Its main focus lies on the support of automotive development engineers throughout the entire design, testing and validation process of predictive safety functions.

VTD`s architecture allows realistic closed-loop simulations to investigate the interactions of

- vehicle driving dynamics
- vehicle sensor systems
- vehicle actuators
- driver
- driver assistance systems
- environmental conditions
  (weather, road conditions, traffic situation, Car2x data, …).

As different technical applications and development stages require specific test methods and tools, VTD supports multiple simulation variants by means of reusable components, interfaces, models and tools, see figure 1.

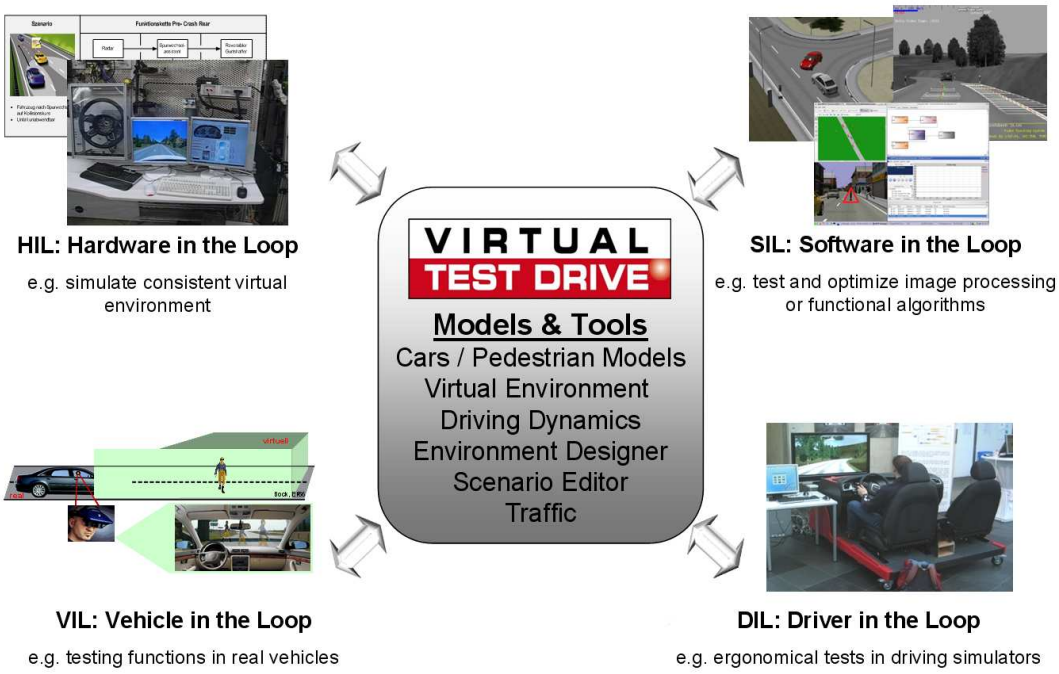

Figure 1: Virtual Test Drive

The operation mode Software-in-the-loop (SiL) allows the early testing of algorithms on ordinary computer hardware in a closed loop and the automatized verification of test data sets at a later stage. Driver-in-the-loop (DiL) simulation may be used for the interactive testing of algorithms in order to get feedback of human test probands at an early stage. Hardware-in-the-loop (HiL) offers the possibility to test and validate systems on already defined target hardware (e.g. Electronic Control Units for mass production) in a closed loop. Vehicle-in-the-

loop (ViL) can be used in parallel at different development stages when the focus is shifting to the limits of vehicle dynamics or safety and assistance systems.

Especially within the SiL operation mode as the focus of this paper the easy and flexible access of VTD simulation data from a runtime environment with effective visualization and evaluation tools is of great importance. In order to increase the efficiency of the software development process the runtime environment should allow model-based algorithm implementation and automatic code generation for diverse hardware targets. The subsequently described application programming interface for MATLAB/Simulink (2) offers all these possibilities.

MATLAB/Simulink-API – An automatically generated application programming interface

In order to allow a bidirectional access to VTD simulation data for algorithms implemented in Simulink, an application programming interface (API) was developed for MATLAB/Simulink called VTD Communication Library (VTDComLib), see figure 2.

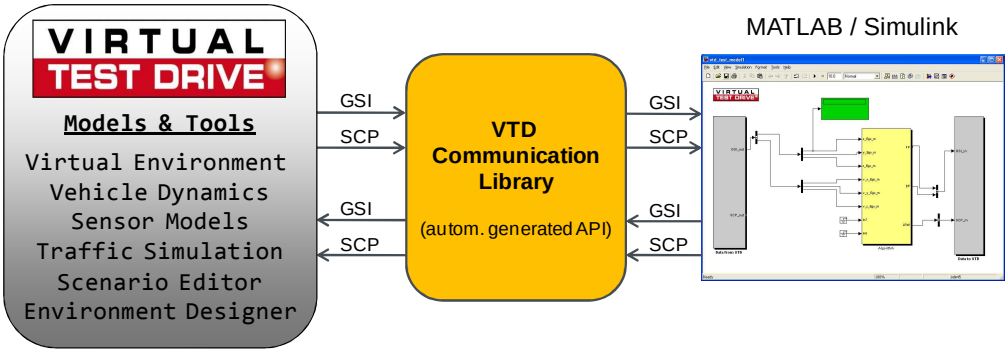

Figure 2: VTD Communication Library

The library uses the Generic Simulation Interface (GSI) and the Simulation Control Protocol (SCP) interface to establish a real-time data connection between Simulink and the VTD simulation environment, see figure 3. The GSI interface provides read and write access to a large number of simulation variables, e.g. position, dynamics and state values of vehicles, sensor outputs, road marks, environment conditions, etc., while the SCP interface allows to query and set parameters which control the behaviour of the simulation environment itself, e.g. start and stop the simulation, trigger next frame computation, enable/disable event triggers, etc..

The required API code to access GSI and SCP data from Simulink is generated automatically, as it is expected that continuous development efforts related to the extension of VTD and the respective GSI interface require frequent changes in order to be able to fulfil new requirements from algorithm developers and testers. Therefore a code parser and code generator tool chain based on the software ANTLR (3) was implemented, which uses the C-programming language header file declaring the contents of GSI network packets as parser input and generates the resulting code for the API in Simulink based on a given Parser Grammar definition and code templates for C-, MATLAB- and Simulink .mdl files, see figure 4.

The data structures and declarations within the GSI C-Header file are described by comments in a Doxygen (4) compatible format. The grammar provided to the ANTLR parser contains expressions for dealing with C-language constructs, pre-compiler definitions, Doxygen style

comments and Doxygen style Meta-Tags. The latter ones are used to have a more fine grained control over the code generation process, e.g. whether to create special Matlab-code for the handling of C-Language Enum variables or to provide information about the SI unit of a certain variable to the Simulink API user.

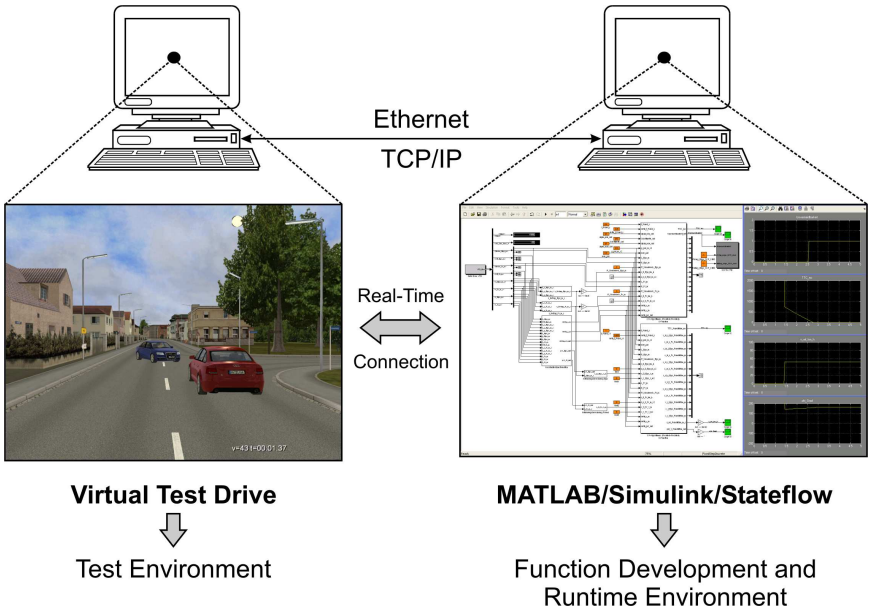

Figure 3: Architecture of the simulation environment

The parser and code generator tool chain generates the C-language code for a MATLAB/Simulink s-Function, a MATLAB Bus Object Definition File for a GSI input and output bus as well as a Simulink .mdl file specifying Simulink VTD Communication Block Elements for sending / receiving GSI / SCP data.

The automatically generated Simulink API achieved with this tool chain allows the comfortable access of VTD simulation variables by means of the Simulink VTD Communication blocks and the MATLAB Bus Selector Dialog for selecting individual data signals.

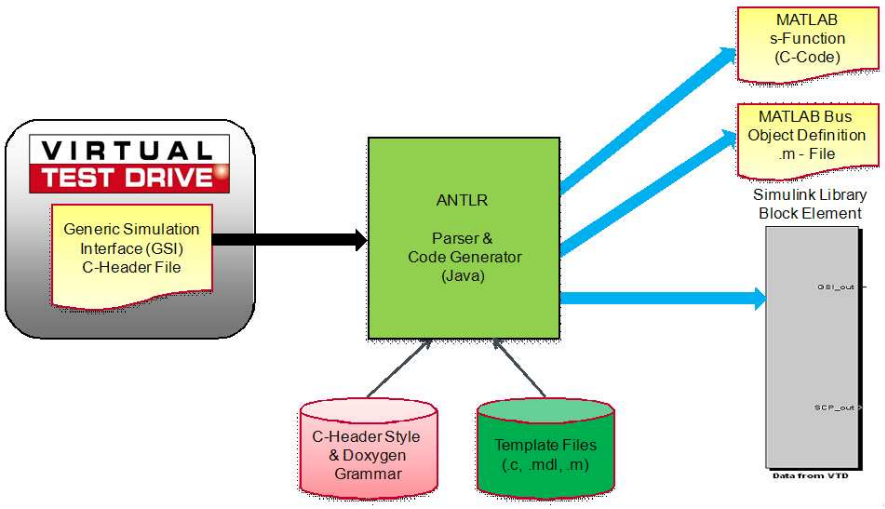

Figure 4: Parsing and Code Generation

ALGORITHM DEVELOPMENT – Testing results of a predictive safety algorithm

Various automotive applications in the active and passive safety use anticipatory sensor signals concerning the vehicle environment. Aspects like the collision probability, the unavoidability of a crash or the initial parameters of an arising accident are the basis for the triggering or adaptation of the application specific safety actuators. In this case an algorithm using anticipatory sensor data during the pre-crash-phase to predict the parameters Time-To-Collision (TTC), relative collision velocity (v_{rel}) and collision angle (φ) at the time of contact was implemented as a Simulink s-Function and exemplarily tested in the presented environment. The testing was done on the basis of a huge pool of characteristic pre-crash-scenarios statistically representing the GIDAS database (5).

In the following the algorithm testing procedure and the resulting prediction outputs are demonstrated on the basis of two characteristic collision scenarios taken from the extensive testing pool available in VTD. The selected scenarios are shown in figure 5.

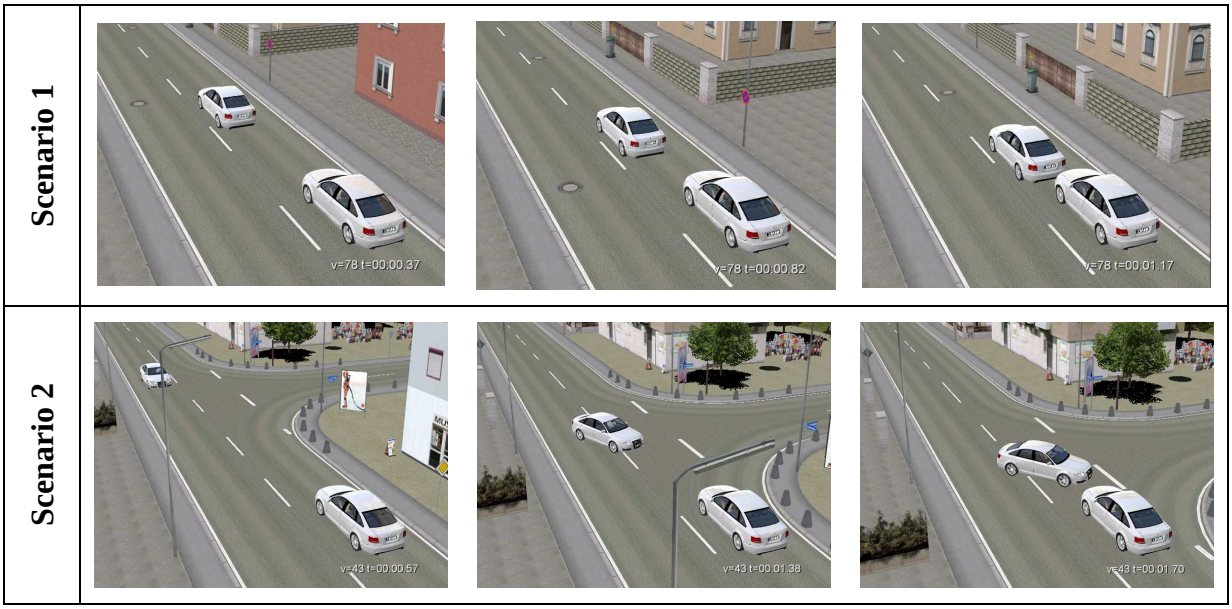

Figure 5: Test scenarios in Virtual Test Drive

In the first accident scenario the virtual test vehicle (ego vehicle, v_1 = 78 km/h) collides frontally with the rear end of a laterally slightly staggered vehicle driving ahead (v_2 = 52 km/h). In the second scenario the ego vehicle (v_1 = 43 km/h) is frontally hit by an oncoming vehicle turning left at an intersection (v_2 = 20 km/h). These two scenarios are typical examples for real-world situations where predictive frontal safety applications can significantly reduce the consequences of an accident or even avoid the accident completely.

For both test cases the types of the two colliding vehicles (in this case identical) and a virtual pre-crash-sensor for the ego vehicle were specified in VTD. In the defined configuration the predictive sensor is mounted on the back side of the ego rear-view mirror and geared towards the driving direction. The sensor position and the resulting three-dimensional sensor cone are illustrated in figure 6.

Figure 6: Sensor cone for the virtual pre-crash-sensor

The shown virtual sensor has a longitudinal range of 20 meters, a symmetric horizontal aperture angle of 60 degrees and a vertical aperture angle of 6 degrees. The currently implemented idealized sensor model offers exact data concerning the actual geometric and kinematic state for all objects intersecting with the defined sensor cone.

When the described scenario is started in VTD the simultaneously running Simulink-API-interface described above offers the mentioned predictive sensor data as well as kinematic data concerning the ego vehicle to a Simulink model containing the mentioned crash prediction algorithm.

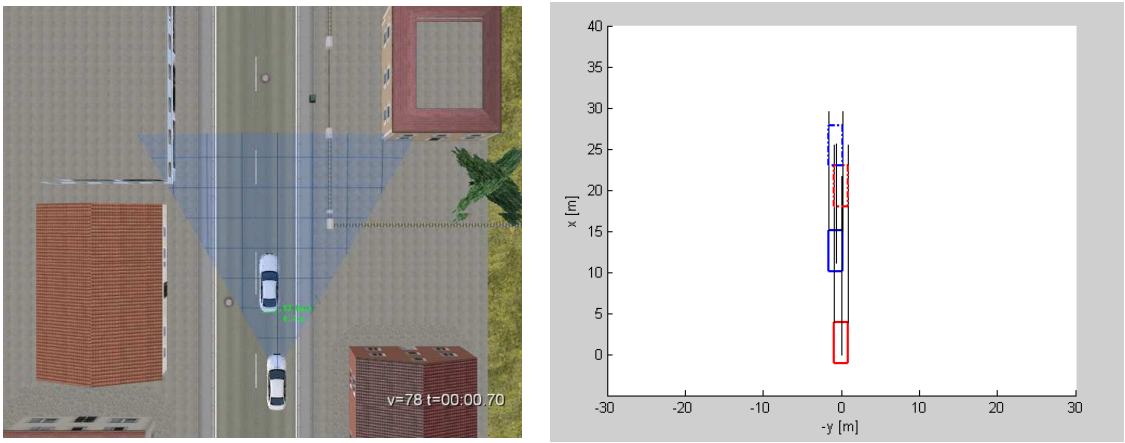

Figure 7: Prediction of the crash constellation and the collision parameters

Based on the longitudinal velocity, the longitudinal acceleration and the yaw rate of the ego vehicle as well as the predictive sensor data concerning the collision opponent (distance and relative velocity in longitudinal and lateral direction and dimensional information) the algorithm estimates the expected collision constellation and returns the TTC, v_{rel} and φ as output parameters.

The algorithm output for the two accident scenarios presented in figure 5 over the time period of about one second before the mechanical contact of the colliding vehicles is shown in the following plots.

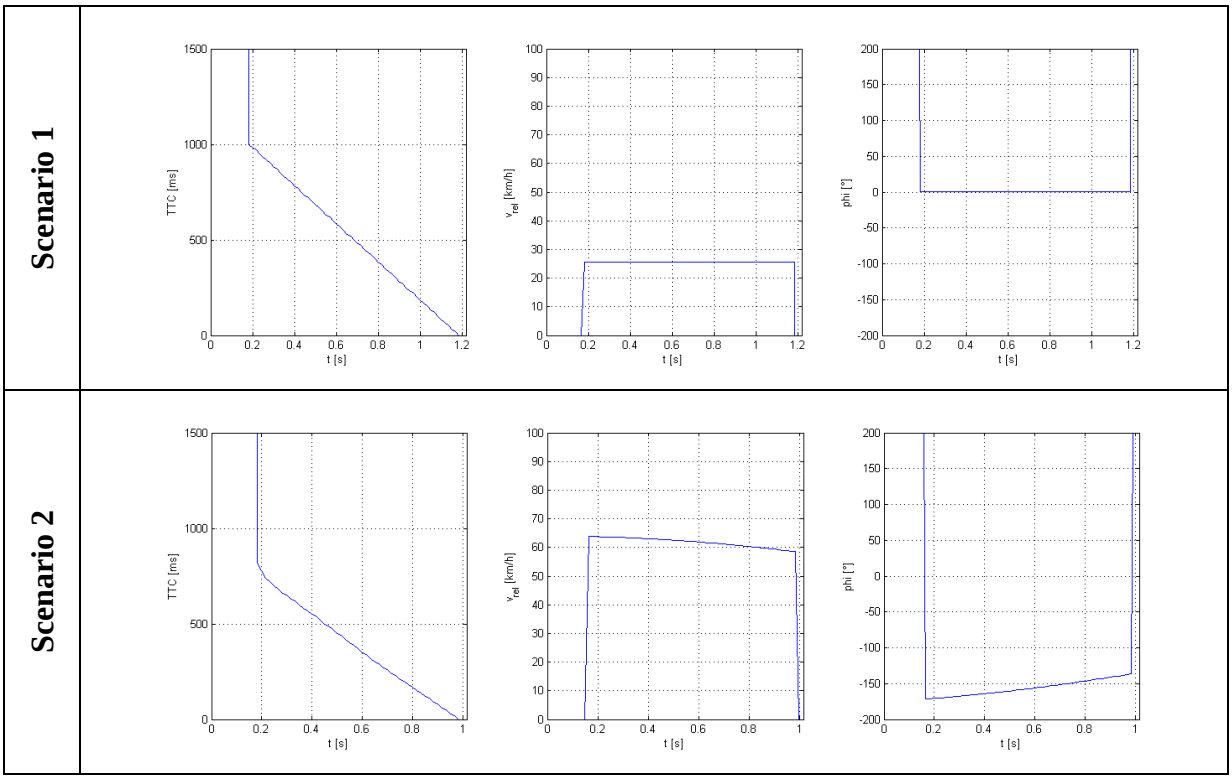

Figure 8: Algorithm output in the selected collision scenarios

As expected in both scenarios the TTC which is predicted with a forecast interval of 1 second continuously decreases till zero at the time of contact. The predicted relative velocity at the collision time amounts to about 26 km/h over the whole forecast interval in the first scenario (v_1 = 78 km/h, v_2 = 52 km/h) where both vehicles are driving constantly in the same direction. In the second scenario (v_1 = 43 km/h, v_2 = 20 km/h) the predicted relative velocity decreases from about 63 km/h to 59 km/h during the forecast because the oncoming vehicle decelerates along the curved trajectory. The collision angle is constantly estimated to 0 degrees during the prediction interval in the first scenario as both vehicles don't change their driving direction. Because of the curved trajectory the predicted collision angle in the second scenario changes from about 175 degrees to approximately 140 degrees.

By variation of the sensor origin and the geometry of the sensor cone in VTD different sensor constellations can be tested and evaluated in connection with the predictive algorithm embedded in Simulink. Thereby the moment of detection and the resulting effect of a predictive safety system under given scenario conditions can be analyzed in detail.

CONCLUSIONS

For the testing and optimization of predictive safety algorithms especially in the pre-crash-phase the use of simulation methods is indispensable. The presented simulation environment offers the possibility to test algorithms working on anticipatory sensor data securely and reproducibly with a big coverage of relevant scenarios. Furthermore the described simulation tool chain represents an efficient means for the communication of technical ideas and the descriptive functional presentation of predictive safety systems. In order to improve the simulation-based performance evaluation of safety applications e.g. under different weather or lighting conditions complex models for anticipatory sensors have to be integrated into the

simulation environment. In this context a close cooperation between the original equipment manufacturers and suppliers in the automotive industry and a standardized way of sensor modelling and sensor model exchange are necessary to be able to do the release of predictive systems with the main focus on simulation-based methods.

(1) Neumann-Cosel, K. von, Dupuis, M., Weiss, C., “Virtual Test Drive – Provision of a Consistent Tool-Set for [D,H,S,V]-in-the-Loop”, In Proceedings on Driving Simulation Conference Europe, 2009, Monaco

(2) MATLAB 2007a, Copyright © 2000-2006 The MathWorks Inc.

(3) Website ANTLR Parser Generator Software Project, http://www.antlr.org (March 15th, 2010)

(4) Website Doxygen Project, http://www.doxygen.org (March 15th, 2010)

(5) GIDAS, “German In-Depth Accident Study”, http://www.gidas.org (March 10th, 2010)