VIRTUAL TEST DRIVE
PROVISION OF A CONSISTENT TOOL-SET FOR [D,H,S,V]-IN-THE-LOOP

Kilian von Neumann-Cosel[1,2], Marius Dupuis[3], Christian Weiss[2]

1) INI.TUM, 85055 Ingolstadt, Germany
neumannc@ini.tum.de

2) AUDI AG, 85045 Ingolstadt, Germany
christian.weiss@audi.de

3) VIRES Simulationstechnologie GmbH, 83026 Rosenheim, Germany
marius@vires.com

Abstract

This paper focuses on the concepts behind a tool-set developed for the provision of a (virtual) test environment which has been specified and developed by the authors and which is being used for the development of active safety and driver assistance systems in the automotive industry. The key aspect behind the tool-set is a wide range of applications which shall be covered consistently throughout the entire development process.

Résumé

The Task

Driving simulators tend to be specialized packages of software and hardware. They either concentrate on soft real-time issues with a high degree of interactivity, on hard real-time conditions or on non real-time data generation.

In the automotive industry, and especially in the use cases described in this paper, various types of simulators are increasingly used for the development of active safety and driver assistance systems. These systems have become key properties of the resulting product, and are, therefore, involved in the development of a new vehicle from early concept stages up to prototype and series testing.

Figure 1 illustrates the development stages vs. the test tools used during the stages:

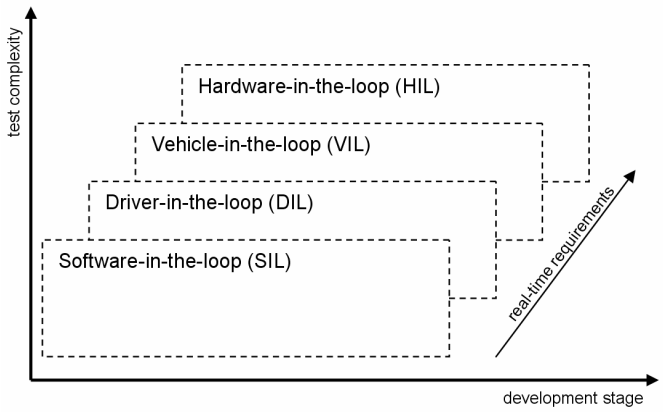

Figure 1: Development stages vs. test tools

- Software-in-the-loop (SiL) systems may be used for early testing of algorithms and for the verification of test data at a later stage. SiL testing will be performed with automated software setups using driver and vehicle models for both the own ship (i.e. the vehicle under investigation) and the surrounding environment (vehicles, pedestrians, objects etc.). With SiL systems, a large number of tests may be run.
- Driver-in-the-loop (DiL) systems may be used for interactive testing of algorithms within test scenarios that have been identified as “interesting” during the earlier SiL stage. They, typically, consist of a full-size mockup with virtual environment but may also be reduced to some kind of single-computer solution with only a joystick or a game-wheel connected as input device. Due to the real-time constraints of the human being, the maximum number of interactive tests at reasonable costs is limited.
- Vehicle-in-the-loop (ViL) systems will be used at an even later stage when the focus is shifting to the limits of vehicle dynamics or safety and assistance systems. Also, data may be broadcast into the vehicle's bus systems for already attached devices which are under investigation. Another key aspect of the ViL stage is the complete realism of the vehicle behavior which helps reducing simulator sickness considerarbly.
- Hardware-in-the-loop (HiL) systems will be from the availability of prototypes of “black-boxes” up to the testing of systems which are used as series standard.

The common property of the above configurations is that “test drives” are performed on more or less virtual levels. Therefore, the effort of this paper's authors to standardize and increase

portability of test scenarios and resulting data between the simulators and, hence, throughout the development stages, is called the “Virtual Test Drive” (VTD).

Harmonizing Data Flows

The functionality of active safety and assistance systems is based on data collected by various kinds of sensors. The data collected by the sensors is typically processed by some kind of algorithms or “functions” and the resulting actions are introduced into the system by various kinds of actuators (see figure 2)

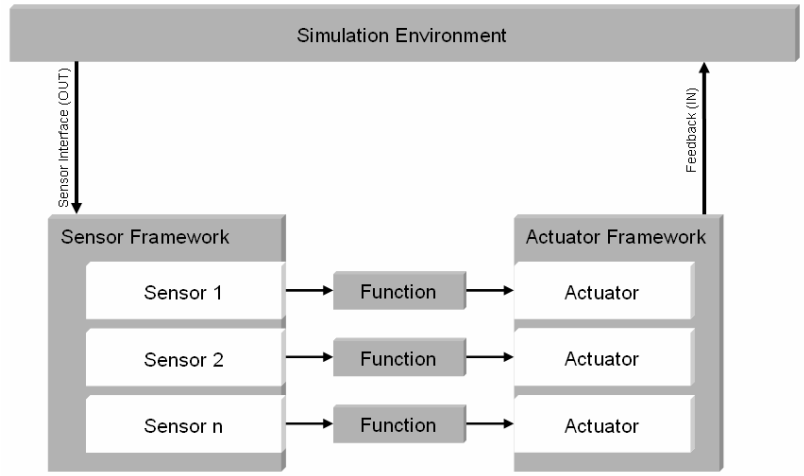

Figure 2: Data flow between simulation environment, sensors, functions and actuators

In the simulation, most items which are to be sensed or which influence the sensing have to be made available as simulated data sets. For pure software solutions, vehicle data may be provided by a large range of commercial vehicle dynamics software packages of varying complexity. Environment data may be provided by a virtual environment simulation package generating data of the road network, of other participants, i.e. vehicles, pedestrians and obstacles, and ancillary data, e.g. weather conditions. It may be required as sets of “perfectly sensed” digital data and/or as image data (for image processing sensors).

All data must be available throughout all stages of the development either as run-time data (i.e. computed on demand) or as playback data (i.e. computed once and fed back into the system on demand).

The system whose development and actual use is described in this paper tries to harmonize the data generation, distribution, storage and playback means while adapting them to the actual simulator setup.

Figure 3 illustrates the data flows that are typically required between the various simulator setups:

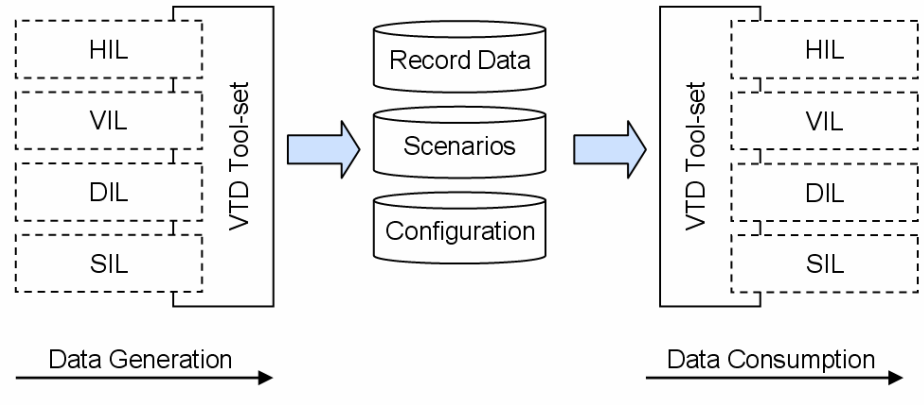

Figure 3: Data flow between simulator setups

Within a simulator setup, a high degree of flexibility concerning the components involved is required. As mentioned above, various vehicle dynamics packages or driver models may be used. Therefore, not only the data flows between the basic simulator types have to be harmonized, but also the data flows within a simulator. New components of any kind must be provided with an easy means to attach, adapt and interact with the simulator (see figure 4).

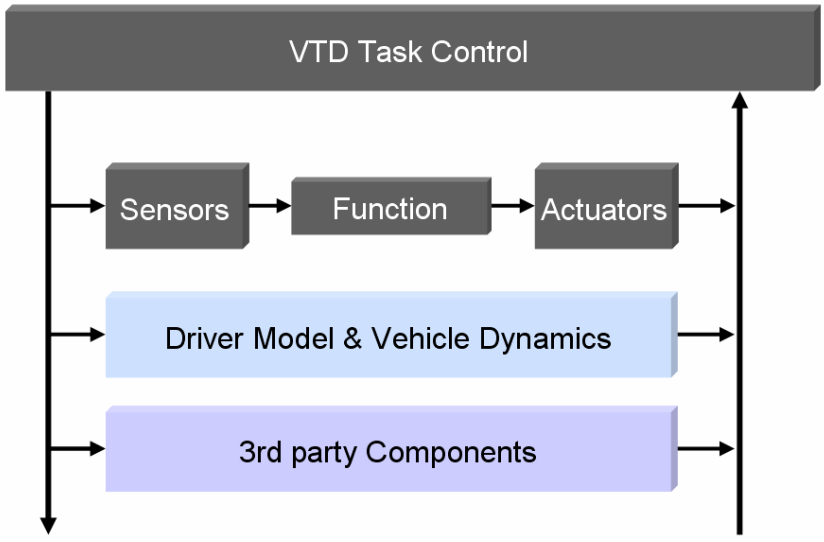

Figure 4: Harmonized I/O between VTD and other components

The Tool-Set

Corresponding to the data flows identified above, the design of the “Virtual Test Drive” tool-set is based on a distinction between core components of the simulation, extended components and target components.

Core components make up the actual simulation framework which handles the data flow within the simulator and comprises basic functionality like record, playback, data collection, and data distribution. **Extended components** are the vehicle and environment simulation (dynamics, driver, traffic, IG). **Target components** are the ones which will actually be contributed and/or modified by the user in his role as developer of an active safety or assistance system (sensors, functions and actuators). See also figure 5.

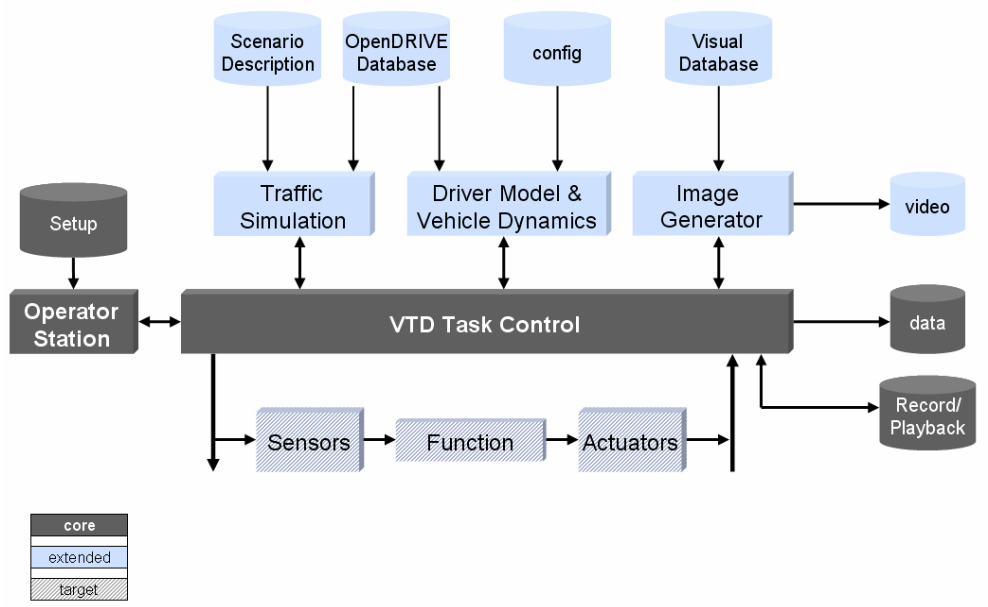

Figure 5: The categories of VTD components

As one can see from the figure, it makes sense to introduce a high degree of modularity within each class of components, so that adaptations to the various use-cases (see title of this paper) and user-specific components can be performed fast, consistent and with minimum impact on the system architecture.

In this context, consistency is the primary key since it shall be possible to use all test setups, use cases etc. throughout the entire range of applications with seamless adaptation to the actual target configuration.

The tool-set consists of a core application, so-called TaskControl, and a set of attached components. The TaskControl controls, conditions and handles all data flows within the simulator setup. In addition, the simulator's state is being controlled.

Internally, the TaskControl is further modularized so that software modules exist for single or various instances of:

- communication interfaces
- players (all participants)
- environment
- cameras
- output devices
- data sets

The virtual environment of the simulation consists, as usual, of some static database and dynamic elements. Key environment elements for active safety and assistance systems are the own vehicle and the other participants, all referred to hereafter as “players”.

The players may be realized as a mixture of external and internal players with the latter being generated by a traffic simulation and the former being controlled by external sources. These external players provide the main distinction between [D, H, S, V]-in-the-loop. They are equipped with a so-called “mock-up module” which is the physical interface between player input, player dynamics and player output.

So, once again emphasizing the players, the different use cases are realized as follows:

	software-in-the-loop (SIL)	driver-in-the-loop (DIL)	hardware-in-the-loop (HIL)	vehicle-in-the-loop (VIL)
driver model	software	human	software	human
vehicle dynamics	software	software	software	real
mock-up	none	various (game wheel, joystick, full-scale simulator)	various (real-car I/O devices)	real

Pedestrians, who make up another large fraction of players, are in all cases assumed to be pure software models.

With the actual distinction between the applications taking place at the I/O level of the players, the influence of additional setup-specific components on the data flows within and outside the simulator can be neglected. Data sinks and source just need to comply with a narrow set of harmonized communication and data protocols.

Communication and Data Protocols

The data flows in the Virtual Test Drive are based on three harmonized definitions:

1) OpenDRIVE, an industry standard for the description of road networks
2) “Generic Simulation Interface” (GSI) for run-time data
3) “Simulation Control Protocol” (SCP) for control data

All definitions are well documented and open to each user (for OpenDRIVE, see www.opendrive.org).

In the tool-set described here, **control data** has been defined as an ASCII data stream containing an XML-formatted set of commands. This instruction set and its network format have been named “Simulation Control Protocol”. It contains genuine simulation control commands (like start, stop), configuration commands (e.g. ports, data flows), environment control commands (e.g, maneuvers for traffic vehicles) and target control commands (e.g. actions for a safety system). So, it basically comes as a language for traffic and experiment control.

With the syntax being XML, commands can easily be scripted and understood even by casual users of the system. Tools allowing the direct input of plain-text command sequences into the system facilitate further the interaction with the system. The hierarchy of XML provides a comfortable means to make commands only available to certain components of the tool-set by having them interpret only the XML tags they know and skip any unknown tags and their respective children. So, users only have to learn the commands actually affecting their use cases.

For the **run-time** data binary data flows are preferred due to higher throughput. The tool-set described here consists of some internal data flows between components which already existed when the project started, and an "outside" world which was designed according to the different use-cases and which could be provided with a harmonized data interface.

The data protocol arising from this harmonization is the so-called "Generic Simulation Interface" (GSI) which is an abstract super-set of all data that may be required by outside data consumers (like sensors) from the virtual environment, mock-ups etc. This GSI is complemented by a GSI-I which describes the data flow from outside components into the tool-set (with the -I meaning "in").

Data Recording

As stated at the beginning of this paper, a means is required to port test data from one application to another (e.g. from DIL to HIL). This may be done on two levels:

- Scenario Definition
- Data Recording

Using harmonized scenario definitions, as is being done in the tool-set, guarantees that test scenarios may be run on all test setups. However, each setup may use different components (e.g. different vehicle dynamics packages), so that the resulting run-time data may differ considerably.

This problem is overcome by a data recording functionality within the TaskControl which records all data of the virtual environment that may change during the simulation of a scenario (players, objects, environment data etc.).

In order to keep data volumes at a minimum, all that can be derived from this data within the core components of the tool-set is not being recorded. At the playback stage which may be run at any of the foreseen setups, the full data content will be re-created from recorded and derived data and all interfaces, basically reduced to the SCP and GSI, will be fed with data as if they were running in a live simulation.

So, coming back to the original intention of the tool-set and the work performed during its creation, it will be possible to run one of the early SIL tests that have been used for the definition of e.g. an active safety system, "some" time later on the actual hardware in a HIL and verify the algorithms put into hardware.

Usability

Usability of the tool-set has been one of the key aspects during its design. Users vary from the ones who just want to run pre-defined scenarios in order to generate, e.g. some video data, to the ones who want to interface with the tool-set in order to control simulations and manipulate run-time data.

The control and run-time data interfaces have been described above. On top of this, the tool-set is complemented by an API which allows the user to write his own components (e.g. sensors) and attach them as libraries to various manager processes (e.g. sensor manager). The sometimes painful task of dealing with data flows, ports, network etc. on a low level is hidden from the user and he may concentrate on using abstracted data interfaces and extending methods provided by the libraries.

Running Installations

The Virtual Test Drive started at the end of 2007 and is meanwhile being used in all types of applications referred to in this paper. Separate papers at DSC 2009 will be giving a deeper insight into the actual use of VTD.

References

[1] OpenDRIVE – An Open Standard for the Description of Roads in Driving Simulations, M. Dupuis, VIRES Simulationstechnologie GmbH, H. Grezlikowski, Daimler AG, Proceedings of the DSC 2006.

[2] Evaluation of an Active Safety Light - Using Virtual Test Drive within a Vehicle in the Loop Simulator, von Neumann-Cosel e.a., INI-TUM / Audi AG, Germany, Proceedings of the DSC 2009

[3] Virtual Testing of an Image Processing ECU - Using Virtual Test Drive within a Hardware in the Loop Simulator, von Neumann-Cosel e.a., INI-TUM / Audi AG, Germany, Proceedings of the DSC 2009