ABSTRACT
Cyber-physical systems are nowadays used as the term for systems which perceive data from their surroundings for algorithmic processing and enrichment by digital information to generate interactions in their specific context. In recent days, a prominent example for these systems are autonomously driving vehicles that continuously have to sense their environment to calculate the next set points for their control algorithms. The development of these complex and interconnected systems requires the combined utilization of simulations and test-runs in reality to assess the system’s quality and to increase the developer’s and customer’s confidence in the resulting implementation. The combination of interactive as well as unattended simulated system tests has to be in tight coordination with their real counterparts to derive reliable results on the one hand and to save valuable resources on the other hand. In this article, the development and test processes of two autonomously driving vehicles are analyzed to derive essential design drivers for a domain-specific language to unify the description, evaluation, and mutual feedback of simulative and real test-runs.

Categories and Subject Descriptors
D.2.5 [Software Engineering]: Testing and Debugging

General Terms
Cyber-Physical Testing with Agent-based Test Oracles

Keywords
Simulation, Cyber-Physical Systems, Agent-based Test Oracles

1. INTRODUCTION
Embedded systems like vehicle collision prevention systems with complex control algorithms serve reliably on a daily basis without attracting attention. Nowadays, these systems are additionally combined with data sources and remotely running data processing algorithms to so-called cyber-physical systems (CPS) [14] to enable a more appropriate behavior even in complex situations within their surroundings. For example, areas at intersections, which are partly occluded for on-board sensors, can be complemented by raw and preprocessed data from surveillance cameras and trustworthy input from other traffic participants like vehicles or pedestrians with smartphones.

The complexity of these systems enforces developers to not only test these CPS by test-drives in reality solely. Thus, simulation environments already play an increasing role [19] during the development. However, the increased usage of simulations raises further questions like how to effectively integrate results from simulation runs during the development process to plan, reduce, and predict results for test-runs in reality [5]. As shown exemplary in Fig. 1 and Fig. 2 and also documented by successful long test-runs in reality [18], even highly complex CPS like autonomously driving vehicles gain more public interest because they enable and foster the development of new safety vehicle functions, which additionally rely on sensor data from a vehicle’s surroundings.

An effective quality assurance of these systems need to include the planning, execution, and evaluation of both simulative and real test-runs in a feedback loop to optimize and focus each of them, respectively. Thus, the development of CPS should be supported by a “cyber-physical testing” that integrates the benefits from both test environments to provide the required answers for developers. Therefore, the quality
assurance processes for the autonomously driving vehicle “Caroline” and the experimental vehicle from the University of California, Berkeley are analyzed in this paper because both utilized elements from simulative and real test-runs.

The rest of the paper is structured as followed: First, related work is discussed followed by a description of design drivers for the domain of quality assurance for autonomous driving vehicles. Based on that discussion, essential design drivers for a domain-specific language are derived that allow a test case description and evaluation for simulative, mixed reality, and real test-runs as referred to “cyber-physical testing”. The behavior evaluation of CPS relies on “agents”, which are software components for simulative and mixed reality test-runs on the one hand, and human test engineers for test-runs in reality on the other hand. Their effective interplay is discussed in the last part of this article.

2. RELATED WORK

During the development of “Caroline” and the experimental vehicle from University of California, Berkeley, several articles were published, which however did not focus yet on the combined usage and mutual feedback of simulative and real test-runs. For example, [2] describes preliminary results from using simulations during the actual software construction process for the vehicle software from “Caroline”. A very detailed description of her development process is given by [7], which extends [3] significantly. The development process based on “story cards” which were derived from the actual requirements specification from DARPA [9]. These cards contained a small sketch of the described traffic situation, a behavioral description of all involved traffic participants, and a sprint roadmap consisting of decomposed technical requirements for each software module together with measurable goals. These cards were intensely discussed with all developers at the beginning of each sprint and used for designing virtual test-drives for the dedicated simulation environment. Thus, the evolving software modules could be tested in a simulate-first approach before the real test-drives were planned and carried out as described on the story card. In [6], especially the impact of these regression simulations for “Caroline” during the semi-final of the 2007 DARPA Urban Challenge was analyzed.

The experiences from “Caroline” resulted in a development framework, which was used during a joined project with the Center for Hybrid and Embedded Software System (CHESS) at University of California, Berkeley [4], which itself has experience from the 2007 DARPA Urban Challenge [16]. In contrast to the previous simulation environment, raw sensor data for cameras and single layer laser scanners could also be simulated in an OpenGL 3D environment on a GPU based on a formal description of the CPS surroundings. Furthermore, the entire simulation environment could be embedded into self-contained unit tests to be executed in an unattended manner by a continuous integration system.

In both projects, the dependencies between simulative and real test-runs were not analyzed yet. Thus, this article discusses preconditions and potentials when combining both approaches on a domain-specific, formal level together with necessary transformations to support essential use cases.

Besides the directly related work, there are further publications available aiming in related directions. The authors of [10, 11] for example suggest to augment realistic scenarios by virtual objects to enable “what would happen if”-simulations. Thus, they are focusing on a realistic environment enriched by simulative elements. In contrast to their approach, this article focuses on the combined description for planning, utilization, and evaluation of separately carried out simulative and real test-runs to optimize both in a feedback loop.

Several mixed reality approaches are described by the authors [17] who also describe a use case where they combine a head-mounted display (HMD) and a physical car to present different interiors to customers. Combining this approach on a real proving ground allows the execution of even complex traffic scenarios without the need for several test engineers as described by [8]. However, HMDs augment only the visual perception by the human driver and thus, an agent-based evaluation of the CPS behavior is not addressed compared to this article.

The authors of [12] present a vehicle hardware in the loop (VEHIL) approach which replaces some simulated vehicles by real moving mock vehicles while the actual tested CPS vehicle is fixed on the proving ground. Thus, its entire environment is moved relatively to the CPS allowing a safe, controlled, and reproducible evaluation. While their approach focuses mainly on the precise reproducibility of complex sensor-based systems, some restrictions for real test-drives are also applicable; for example, the required time for executing the planned choreography can only be limitedly accelerated compared to a pure simulative environment.

The current literature shows a trend to substitute parts from real test-runs by simulative elements (cf. [8, 11]) to reduce resource utilization and to increase the overall safety, or to map characteristics from purely simulative approaches to real test-runs (cf. [12]) to increase reproducibility. However, an analysis of how to efficiently utilize both approaches in a combined manner and therefore design drivers, transformations, and metrics are still not available to the best knowledge of the author.

3. DESIGN DRIVERS FOR CYBER-PHI-

SICAL TESTING

During the development of “Caroline”, 95 virtual scenarios for the simulation environment were defined covering two test sites in Braunschweig, Germany, the Southwest Research Institute in San Antonio, TX, and Victorville, CA. Additionally, the proving ground Richmond Field Station in Richmond, CA was used during the development of the experimental vehicle in the joined project with CHESS. In the following, characteristics of the domain for realizing test processes for CPS on the example of autonomously driving vehicles are outlined, and common design drivers are derived.

The setup of the real test-drives on special proving grounds for both projects can be divided into three different categories. First, unplanned experiments to quickly try improvements and changes to an algorithm with the goal to get an impression if the changes evolve in the intended direction; second, regular and scheduled vehicle test-drives, which follow a predefined test plan with a scenario setup, validation of mea-
measurable goals, and a written report about the current quality and maturity of the vehicle software—official certifications would also be covered by this category; third, demonstration test-runs, which are used to present the current status of a project.

On the other hand, software evaluations with simulations could also be classified in a similar manner: First, developers simply run the algorithm with selected data from a very specific scenario that covers the developer’s intentions; second, unit tests or regression simulations (cf. Sec. 2), which evaluate an algorithm based on a formal test model to produce a test report; third, algorithm demonstrations, which are only limitedly applicable for CPS without the technical environment. In the following, only the second classes of each group are of interest.

Figure 3: The cyber-physical testing-loop for evaluating CPS consists of the CPS itself, its necessary environment and an appropriate evaluation approach. Depending on the simulative level, the environment itself is either part of the simulation or elements from the simulation are used to augment the reality like object models; furthermore, virtual sensors and actors can be used as replacements in real test-runs.

In Fig. 3, the testing loop for CPS is depicted. Besides the CPS, appropriate surroundings as well as suitable evaluation approaches are required. Depending on the intended level of simulation, all environmental aspects including sensors and actors are part of the simulation as described in [4]; alternatively, real test-runs can be augmented by virtual object models to evaluate a potentially dangerous situation for example. Furthermore, real actors can be substituted by mocks in case of irreversible acting components like airbags. Moreover, also real sensor data can either be transformed to imitate another detecting device or to artificially reduce the sensor data quality to emulate other weather conditions like fog for example.

CPS—In real test-runs, the CPS does not only include the algorithms to be tested but also the hardware platform including all sensors and actors. In contrast, the simulative test-run requires additionally the correct configuration and mounting positions for sensor and actor models to reflect the real hardware accordingly. Moreover, noise models like drifting clocks and error models like communication errors would improve the simulated reality.

Environment—The behavior of CPS can only be evaluated properly within its intended surroundings. Thus, a realistic environment needs to be set up for both approaches. While structural means like curbs, intersection layouts, or lane markings are necessary for the former approach, a “suitable” model for the latter is required which must be defined in close conjunction with models for the intended sensor and actor models, which rely on a certain level of details to operate accordingly; for example, data for materials of 3D models is necessary to simulate raw data from a sensor more precisely.

Figure 4: Digital road networks as environmental models to navigate through urban-like environments.

In Fig. 4, two examples for environmental models for simulative environments are depicted. In Fig. 4(a), a 3D OpenGL model of the proving ground Richmond Field Station from University of California, Berkeley is shown. That stationary model also served to simulate a stereo vision sensor and a single layer laser scanner model running on a GPU to realize more realistic raw data for the CPS. In contrast, Fig. 4(b) shows the 2D model from the semi-final test area B at the 2007 DARPA Urban Challenge that served as the basis to generate input data on object level in simulations for “Caroline”.

Scenario—Besides the structural definition of the stationary
contents, the situation to be handled by the CPS itself has to defined precisely. For real test-runs, several human test engineers have to carry out a predefined choreography while considering a precise timing to allow comparisons between repeated runs of the same situation; furthermore, the actual time that is required to execute the planned choreography cannot be accelerated. Both aspects differ simulative test-runs from the former ones because repeatability and precision is enabled by principle, and the required time for executing a simulation depends in the best case only on the underlying computing hardware. Thus, simulations could be executed faster than real time and in consequence be carried out more often than real test-runs.

Evaluation—The most important part however is the evaluation process during a specific test-run to classify a test as passed or failed. The evaluating entities will be called “agents” including human test engineers operating on proving ground on the one hand, and software components acting as evaluating processes, which are executed during an unattended simulation run, on the other hand. While the former agents are able to interpret a real test situation and thus, could evaluate possibly differently a repeated test-run, software agents are formal and less flexible in contrast. Furthermore, in simulative environments, all evaluable aspects have to be explicitly supervised by software agents while human testers monitor a real test-run also based on their former experience and with less formal and more naturally described instructions.

Setup—The setup phase describes in both approaches the steps that are necessary to acquire the required resources like the real proving ground or the computation cluster, to configure the stationary and dynamic environment, and to instruct the required evaluation agents. Furthermore, the actual beginning of a test case has to be reported in case of unexpected exceptions for both approaches.

Reporting—A test-run is completed by generating a report containing all information that could support the developers to improve the quality of the CPS’ algorithm. Thus, not only binary results from all agents need to be provided but also all available unprocessed raw data from sensors for further inspection afterwards.

4. MODELS AND TRANSFORMATIONS FOR CYBER-PHYSICAL TESTING

The aforementioned discussion unveiled that evaluating a CPS involves both simulations and real test-runs, which may contain simulative components due to resource restrictions or safety concerns. Thus, modeling both separately from each other does not leverage all possibilities when borders between pure virtual simulations and real test-runs disappear. Hence, test cases for both approaches shall be derived from a common formal description being the single-point-of-truth (SPoT) concept to avoid redundancies in related test cases.

Furthermore, the following concepts must be considered to support both test approaches from a SPoT: First, a unique and consistent model of the CPS’ environment including its sensors and actors; second, a formal description of all evalutable behavioral aspects; and third, transformations to derive purely virtual simulations, mixed reality test-runs containing simulative components, and real test-runs with human test engineers. Furthermore, post-processing is required to link both approaches by transferring raw data from real test-drives into simulative environments or by predicting and planning real test-runs based on results from simulations.

Environmental structure—The formal description of a test case contains a precise definition of the surroundings in which the CPS shall operate. Hereby, the term “precise” is limited by sensor and actor models to realize a purely virtualized test-run. While for actor models differential equations may be sufficient, the generation of sensor raw data directly depends on the modeled details in the environment and simplifications from the physical reality. For example, simple sensor models that rely on detecting angle and range may be sufficient for first drafts of an algorithm; more elaborated models as described by [4, 13], which reflect the underlying physical measurement principle more precisely, are necessary to analyze an algorithm under different environmental conditions.

Besides accurate models for the environment and required sensors and actors, noise and error models are necessary to reflect more realistic conditions. The former cover real effects like communication delays, drifting clocks, and varying uncertainties in input data for example, the latter refer to transmission errors, missing or incomplete input data, and unexpected software component behavior. These models reduce the optimal data quality in simulative environments to enable a more realistic surroundings for virtual evaluations.

As depicted by Fig. 3, environmental components may be substituted by simulative components and vice versa. Thus, the formal model of the environment shall contain concepts like refinement and filtering. By refinement, less accurate models for sensors, actors, and environment objects can be substituted by more realistic ones; for example, simple 2D rectangular vehicles could be refined by complex 3D realistic models. A similar concept needs to be applicable for real sensors, actors, and environmental objects in real test-runs: Hereby, simple sensor models could generate mixed reality input data (cf. [8]) until real sensors are later available. In contrast, the concept “filter” is used to modify data streams according to a given model: As already mentioned, filters can reflect noise and error models on the computational level, but they can also be used to adapt sensor data to imitate changing environmental conditions like fog, rain, snow, or even changing lighting conditions.

Furthermore, it shall be possible to apply these refinements and specialized filters to previously recorded simulative and real test-runs in a post-processing stage. Thus, the number of further test-runs under varying conditions like changed weather or other and additional environmental objects can be extended without carrying out the test-runs again. Furthermore, previously generated test data can be re-used with exchanged sensor models during an evolving development.

Finally, the environmental description shall be visualized in an intuitive manner to ensure a correct modeling. For simulative environments the model can simply be rendered while for real test-runs, a unique and consistent description of how to set up an existing proving ground is necessary.
The same also applies to previously recorded test-runs in both approaches to allow a fast access and visualization of all available data; furthermore, experiments with refinement components or filters can be carried out easily.

Besides the structural components, a formal and hence precise and consistent description of the surroundings’ situation is necessary to allow reproducibility. For simulative environments, these behavioral models are part of the environmental objects like other vehicles; during real test-runs, these behavioral models must be transformed to human-readable instructions for all test engineers who are part of a test-drive. Furthermore, allowed variances shall be specified to enable the generation of slightly differently behaving environmental objects; on the other hand, the variances are required to use human test engineers who can only reproduce a choreography like a traffic situation with pedestrians at an intersection with slight variances without the assistance of technical means.

*Composing agents to test oracles*: A similar concept needs to be available for the evaluation of test-runs. A unique, consistent, and formal description is required to realize software agents, which are continuously evaluating a running simulation. Furthermore, various agents must be composable to analyze different aspects of the behavior of a CPS. On the example of an autonomously driving vehicle, one software agent is evaluating safety distances to other objects while another process is supervising the lane following capability. For real test-runs, these formal descriptions have to be transformed into human-readable instructions.

However, results from both the simulative environment and real test-runs shall be analyzed transparently for developers and engineers. Thus, aggregated and binary results, recorded data, and further log data shall be available in an identical format to enable reporting and comparison later on. Furthermore, it is necessary to evaluate the successful fulfillment of a simulative or real test-run quickly and to drill down to the problematic part of a concrete situation for replay and further inspection.

Finally, all results from simulative and real test-drives shall be archived and retrieved effectively to allow analysis of historic data to predict the maturity of an algorithm. Furthermore, this data can be used to derive further test cases for quickly changing parts of the implementation on the one hand. On the other hand, derivation of similar environmental situations is possible to analyze challenging difficult conditions in the CPS’ surroundings.

*Operations and transformations*: Some transformations were already previously mentioned to enable the re-use of formal software descriptions as human-readable instructions. However, further operations are necessary to ease up the handling of large data which is generated during simulative and real test-runs. For example, it shall be possible to compute the difference between two similar environmental situations where a CPS’ algorithm behaved differently and not as expected. Thus, the analysis to find the erroneous part in the implementation can be accelerated. Furthermore, inconsistencies in test cases like conflicting agents could be calculated and indicated.

In Fig. 5, an instance of a modeling language inspired by both exemplary projects and the previous discussions is shown. The instance is used to evaluate a collision prevention system based on a camera system. First, the actual proving ground is defined with no further real objects besides the CPS. Next, an idealized object for a pedestrian is placed 40m in front of the vehicle to cross the street; its action is depending on the CPS and described as a movement over time in the function ped0 (not shown for the sake of clarity). Afterwards, the sensor for the CPS is defined which is a simplified camera sensor for the simulation and the real camera for the real test-run. The CPS itself is placed at the origin heading to the north; its action is also a movement over time and defined in drive. Finally, two agents are defined which observe the CPS’ behavior regarding its distance to other objects and its maximum velocity. This instance is transformed into a test case for a simulation environment like [4] as well as into
a human readable representation for test engineers. Besides
that, the real test-run’s results must be fed back into a central
measurements data management system like [1] as well as
the recorded data to be accessible for developers; this step
can be easily carried out by the simulation after completing
a test case.

5. CONCLUSION AND OUTLOOK
The quality assurance for increasingly complex and inter-
connected CPS involves both simulations and real test-runs.
However, a combined representation to serve both approaches
is to the best knowledge of the author not available so far
for automotive safety-critical systems which base on sensor
data from their surroundings. In this article, design con-
siderations to realize a cyber-physical testing language that
involves software-based and human evaluation agents as test
oracles are outlined and discussed. Finally, a sketch of such
a language for a collision prevention system is presented.

Future work needs to be done in the area of agents that are
composed to a test oracle because in real test-runs, implicit
experience knowledge from test engineers is available that
influences his interpretation of the CPS’ behavior. This
experience needs to be modeled as software agents to get
comparable and reproducible results for simulative and real
test-runs. Furthermore, feedback from real test-runs for
simulative environments on the level of recorded sensor data
could be analyzed to enable variances of a previous real test-
run. Thus, robustness tests could be executed by adding
further environmental objects into the recorded scenarios.

Moreover, the combination of cyber-physical tests and their
corresponding requirements needs to be further inspected. Fu-
ture work should be carried out to analyze how requirements
for these CPS can be modeled in a central and consistent way
so that test cases for all variants of cyber-physical testing
can be generated.

Acknowledgments
The author is very grateful Prof. Shankar Sastry, Jan Bier-
meyer, and Humberto Gonzalez from the Center for Hybrid
and Embedded Software Systems (CHESS) of University of
California, Berkeley. Furthermore, the author would like to
thank team CarOLO and “Caroline”.

6. REFERENCES
Design and Quality Assurance of Intelligent Vehicle
Automobiltechnische Konferenz - Virtual Vehicle
Systems Engineering Process and Tools for the
Development of Autonomous Driving Intelligence.
Journal of Aerospace Computing, Information, and
and Actuator-based Systems on the Example of
Autonomous Vehicles. Shaker Verlag, Aachener
Informatik-Berichte, Software Engineering Band 6,
Aachen, Germany, 2010.
[5] C. Berger. From Autonomous Vehicles to Safer Cars:
Selected Challenges for the Software Engineering. In
Proceedings of the Conference Automotive - Safety &
Management by Scenario-Based Regression Simulation.
IEEE Transactions on Intelligent Transportation
Driving Software. In C. Rouff and M. Hinchey, editors,
Experience from the DARPA Urban Challenge, pages
[8] T. Bock, M. Maurer, and G. Färber. Validation of the
Vehicle in the Loop (VIL); A milestone for the
simulation of driver assistance systems. In IEEE
Intelligent Vehicles Symposium, number Vil, pages
Criteria. Technical report, DARPA, Arlington, VA,
USA, 2006.
Simulation with Augmented Reality. Performance
Tools and Applications to Networked Systems,
autonomous agents in augmented reality. Journal of
[12] O. Gietelink, J. Ploeg, B. De Schutter, and
M. Verhaeghen. Development of advanced driver
assistance systems with vehicle hardware-in-the-loop
simulations. Vehicle System Dynamics, 44(7):569–590,
July 2006.
Simulation Framework for Time-Of-Flight Sensors. In
2007 International Symposium on Signals, Circuits and
IEEE.
Technical Report UCB/EECS-2007-72, University of
California, Berkeley, CA, USA, 2007.
K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl,
W. Schumacher, J.-M. Wille, P. Hecker, T. Nothdurft,
M. Doering, K. Homeier, J. Morgenroth, L. Wolf,
C. Basarke, C. Berger, T. Gülke, F. Klose, and
B. Rumpe. Caroline: An Autonomously Driving
Vehicle for Urban Environments. Journal of Field
B. Upcroft, A. Makarenko, W. Uther, M. Moser,
Model-based design: a report from the trenches of the
DARPA Urban Challenge. Software & Systems
Reality: Future Dreams Seen at the Border between
Real and Virtual Worlds. IEEE Computer Graphics
Virtual Test Drive - Provision of a Consistent Tool-Set
Simulation Conference, Monaco, 2009.