Requirements-based Simulation Execution for Virtual Validation of Autonomous Systems

Florian Pudlitz
Technische Universität
Berlin, Germany
florian.pudlitz@tu-berlin.de

Abstract

The complexity of software is rapidly increasing in many domains. Therefore, simulations have become established as a testing tool in recent years. Especially the virtual validation of autonomous systems leads to increasingly complex simulation environments. Nevertheless, the scenarios and the simulation results are not linked to the requirements. To close this gap, we develop a lightweight approach that allows the user to extract functional information. Simulation results can then be presented in different levels of detail in the original requirements. This replaces difficult translations of requirements and allows permanent comparison at all test levels.

1 Introduction

In recent years, the number of complex and autonomous systems has increased significantly [PBKS07]. Most of present innovations are driven by software solutions. Software components enable newest assistance systems of autonomous driving as well as intelligent robots in medicine, or increased productivity in the context of Smart Factory. Validating and testing the behavior of these complex systems is challenging because of their dynamic nature and open context. New testing methods like simulations enable a fast, comprehensive, and automated test implementation. Simulations can accurately model complex environments and the behavior of systems. Through self-learning behavior models and dynamic adaptation-based systems, the scenarios in the simulations are less and less defined at runtime. But, today's computer-based simulations are standardized processes, focusing on small specific situations. The used scenarios do not necessarily mirror reality to the full extent. The results of previous simulations were limited to the detection of occurring software and signal errors. In limited cases, due to the absence of errors, the correctness of the functionality can be inferred. Complete verification of requirements and analysis of values in every time step are not performed. A further aspect is the selection of scenarios or simulation objects. Currently, test engineers create simulations manually, but mostly those do not cover the complexity of the entire Software. In my dissertation, I aim to develop a test environment for autonomous systems that is based on simulations. It automatically connects requirements specifications with results of simulation runs. The goal is an automated computer-based long-term simulation. This ensures that the simulation contains all objects of the software description. Requirements specifications of software are often given in natural language or models. A lightweight approach allows a fast and simple extraction of information for test engineers. The

In: P. Mäder, P. Spoletini (eds.): Joint Proceedings of REFSQ-2019 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track, Essen, Germany, 18-03-2019, published at http://ceur-ws.org

selected information is used to control dynamic objects of the simulation.
To handle the large amount of data generated in the simulations, log files will be transferred to a model. The test engineers gain a clear representation of results and a quick overview of the correctly working software parts. The analysis of log files has two main points. Firstly, log files make it possible to go to selected time steps in the simulation and show the specific situation. It facilitates the identification of the causes of errors. Secondly, accuracy of software can be checked in every time step. This helps to display the results of the simulation directly in the given requirements specifications. In addition, this approach allows a quantitative analysis. Within the context of autonomous driving, it is possible to see values like driven kilometers, frequencies of activated functions, weather conditions, or manual corrections by the driver.
This new approach connects different types of requirements with a computer-based long-term simulation. This test method enables the execution of complex and realistic scenarios in which mandatory requirements influence dynamic objects. Large-scale evaluation of log files is an innovative approach to test complex software. The results of this project can be useful for diverse research partners in several industries. This method analyzes whether the information in requirements documents reflects the complex situations in reality, and emphasizes shortcomings. It is a new approach to connect requirements and long-term simulations by means of lightweight extraction of information. The large-scale evaluation investigates various kinds of result presentations and possible conclusions for simulation execution.

2 Related Work

The use of simulative test environments requires not only the modeling of the system under test but also a precise modeling of the scenario. Two essential approaches are in the focus of science. On the one hand, systematic approaches are explained in the works [BOS16, MKK15] in order to create scenarios step by step. On the other hand, the works of [BRSM16, IK14] are based on scenarios with especially critical situations. In both cases, requirements that derive from the functionality of the system are missing. To support scenario generation or automated test case generation, requirements are translated into structured models. Several authors propose different sets of sentence patterns that should be used to formulate requirements [EVFM16, MWHN09]. For natural language requirements, however, methods are missing which reliably support the scenario generation process. Alignment of requirements and test cases is a well-established field of research and several solutions exist. Barmi et al. [BEF11] found that most studies of the subject were on model-based testing including a variety of formal methods for describing requirements with models or languages. One problem in this area is that the generated tests from the model cannot be executed directly against an implementation under test because they are on different levels of abstraction. In contrast to state of the art procedures, in my approach is no necessity for translation into executable languages [BLC$^+$] or formulate requirements [MWHN09, EVFM16]. Therefore, the entire scope of simulation options of the natural language requirements remains without any translation loss. Another improvement of today's standards lies in the testability of software at any state of development.

3 Idea

My principal idea is schematically displayed in Figure 1. The starting point is a document of requirements formulated in natural language containing software specifications. The present requirements are written without using pattern or other grammatical restrictions. Important keywords or sentence phrases are marked by the engineer and then extracted. These are matched with signals of the simulation and system, which is called a mapping. The creation of the simulation scenarios can be strongly supported by the information from the requirements. The engineers thus receive an overview of the necessary simulation properties (e.g. weather conditions, road characteristics or information about other road users) that must be included in the scenarios. In this way, the formulated requirements enter into the process of scenario generation. Depending on the simulators used and their connection, a partially automated or fully automated scenario generation is conceivable. Simulation results are output as log files, which in connection with the mapping results, are fed back to the original requirements document. In addition to log data from traffic simulations, it is also possible to use real driver log data. In this way, real log data can be matched with natural language requirements. The simulation results or real data are displayed directly in the originally analyzed phrases.

First behaviors of the software can be simulated with simple markings early in the development process. Especially new assistance systems or functions such as autonomous driving are very complex and can only be tested with complex simulations. The test engineers therefore need a lightweight approach to evaluate requirements without formal translation. This increases acceptance and does not restrict the requirement engineers.

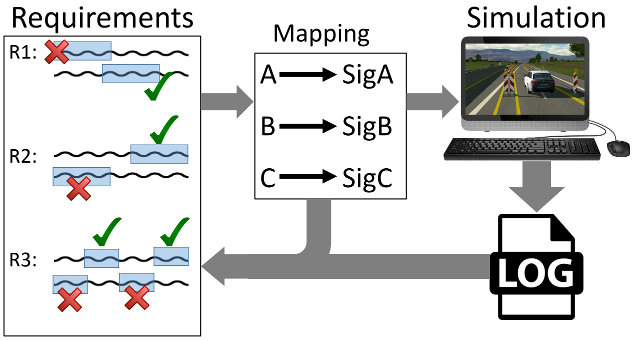


Figure 1: Schematic representation of a requirements specification linked to a simulation with influencing intermediate steps

There are two main motivations to use this approach: to find information needed in order to choose a suitable simulation scenario and to check or monitor functionalities of a software component in different states of development.

In my work, I want to bring together natural language requirements and simulative test management. The methods used allow an evaluation of complex system simulations and a direct comparison to the legally binding requirement documents.

3.1 Research Questions

The aim of this work is the review of natural language requirements by simulative long-term tests. In order to examine the various aspects of the approach, we have pre-set the following research questions:

RQ1: Which scenario generation information can be derived from natural language requirements?

RQ2: How many natural language requirements can be tested with simulations?

RQ3: Can simulations fulfill the requirements equivalent to real test driving?

RQ4: How can simulation results in natural language requirements support the testing process?

To answer the research questions, I would like to develop an integrative test approach that links natural language requirements and simulations. The core idea is to let the engineer decide which software features are observed in the simulation. For this purpose, the requirements are not translated automatically but are intuitively marked by the engineer.

4 Research Method

The research method has two main aspects. First, the natural language requirement must be easily accessible for the engineer to process. For this purpose, the requirements are initially marked manually using a markup language. To address RQ1, this tagged information is used to support scenario generation. The second main aspect of the approach is the processing of the log data. With this processing, RQ2 can be examined. Since the approach is independent of the simulation tools used, simulations at different test levels can be used. To answer RQ3, the log data from Hardware-in-the-Loop and Vehicle-in-the-Loop simulations are compared and displayed in the requirements. RQ4 will be answered as far as possible by qualitative surveys. The individual parts of the approach are explained in more detail below.

4.1 Markup Language

For marking software functions and environment conditions, we developed a lightweight multilevel markup language to connect requirements specifications and simulation runs. While there are many efforts within the research community to explicitly formalize requirements to improve on their validation possibilities [BEF11], this markup language aims to provide the requirements engineer with a means to intuitively annotate natural

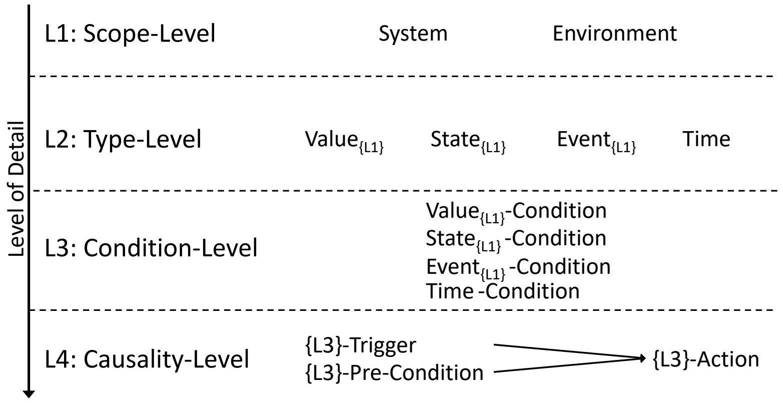


Figure 2: Overview of levels and elements

language requirements in order to unfold the implicitly contained information in a way it can be used for validation purposes within a simulation. The resulting language consists of elements, which are assigned to phrases in the natural language requirements documents with defined content characteristics. This part of the process is performed by an engineer and is the starting point for the automated evaluation by the tool. Figure 2 shows, each element is assigned to one of four levels, which define the level of detail of the evaluation.

The *Scope-Level* is used to differentiate between information on the system and on the simulation environment. As a result, the appearance of the objects in the simulation is displayed. However, no further information is available.

The *Type-Level* distinguishes the phrase of Level 1 into different types of text phrases depending on the behavior in the system. The different Level 2-types influence the type of evaluation and are the basis for the definition of conditions in Level 3.

The *Condition-Level* connects a type of Level 2 with a specific value via comparison operators to create condition statements. However, the formulated conditions have no connection among each other.

The *Causality-Level* establishes a relationship between the conditions of Level 3 and creates causal relationships. This requires detailed knowledge of the system and the necessary work process performed by the user is time consuming. The result however is an in-depth evaluation.

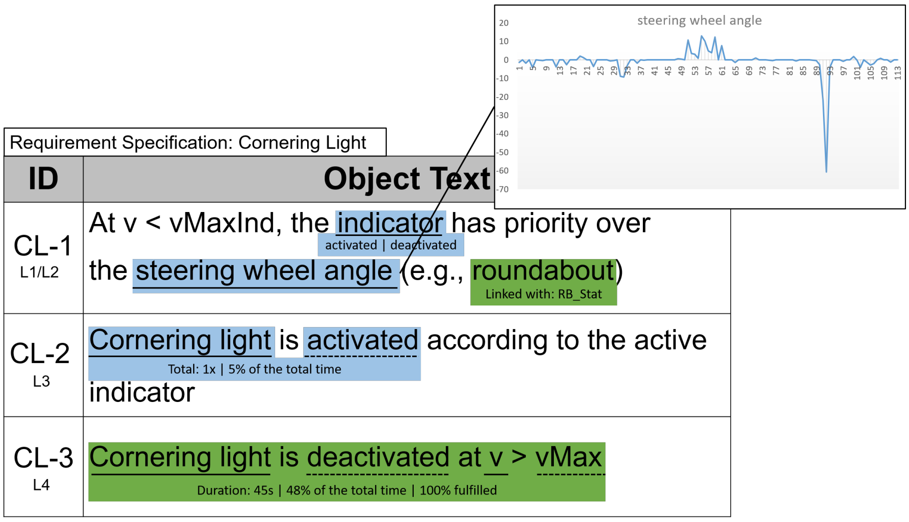
4.2 Log Data as States

The evaluation of the test systems, for example the vehicles, is realized by processing the log data. This has the decisive advantage that simulation steps can be summarized and understood as a status of the system. Every status can then be compared with the natural language requirements. Another advantage of the systematic evaluation of the log data is the traceability of the simulation process. The user can jump back to the simulated environment in case of occurring errors, special situations or vehicle-related maneuvers. In this way, the overall context can be examined more closely. The evaluation of the log data is independent of the test level used. If the simulations are used in different test stages, they can also be compared with each other. In relation to the V-model [FM92], which describes the development process of software in the automotive industry, the test levels Model-in-the-Loop, Software-in-the-Loop and Hardware-in-the-Loop can be compared with each other. Additionally, the log data of real trips can be compared with the simulation runs.

5 Approach

5.1 Marking Requirements

For the extraction of simulation-relevant information, the engineer marks the important sentence phrases in the original requirements. The engineer can choose himself whether the information serves as a requirement for scenarios or the sentence phrases are tested in the simulation. In order not to have to mark all parts of the requirements so that evaluations are possible, a lightweight approach is used. It allows you to observe complete requirements or parts of them in the simulation at different levels of detail. Here we could show how any phrase phrases were marked by the user and evaluated in different levels of detail. To use this method, a tool is developed



Requirement Specification: Cornering Light

ID	Object Text
CL-1 L1/L2	At v < vMaxInd, the indicator has priority over activated | deactivated the steering wheel angle (e.g., roundabout) Linked with: RB_Stat
CL-2 L3	Cornering light is activated according to the active Total: 1x | 5% of the total time indicator
CL-3 L4	Cornering light is deactivated at v > vMax Duration: 45s | 48% of the total time | 100% fulfilled

Figure 3: Resulting marks of a simulation evaluation

in which the natural language requirements are inserted. The tool allows the user to mark the requirements at the four previously presented levels.

5.2 Simulation Execution

Software systems are becoming more complex and can no longer be validated by conventional single tests. Simulations are becoming an increasingly important tool for the validation of software systems for test management. Using long-term simulations and complex scenarios, a wide range of situations can be presented. VSimRTI [Sch11] links different simulators and provides a framework to simulate software from vehicles. As traffic simulator SUMO [BBEK11] is used, which works with real OpenStreetMaps data and traffic with up to thousands of vehicles. The modular design allows the extension, for example, by network simulators or self-developed vehicle functions. This allows complex and realistic scenarios to be generated. When creating the scenarios, today the developers' experience flows in. Through the extracted information from the requirements, simulation properties can be systematically incorporated into the scenario generation and support an automated process. Other simulators have an increased focus on autonomous driving (Carla [DRC$^+$17]) or simulating assistance functions and associated sensors (VTD [vNCNL$^+$09]). Depending on the target system, the appropriate simulator can be used and is compatible with the presented approach. Regardless of the simulator used, natural language requirements can support scenario generation. The approach also allows an evaluation at the vehicle level or for all simulated vehicles in the scenario.

5.3 Representation of Evaluation

An essential feedback mechanism is the presentation of the results in the original requirements document, depending on the chosen elements in the requirements and the analysis of the log data. Figure 3 shows the presentation of the evaluation results based on the simulation run in the presented example.

The evaluation shows that the granularity of the results also depends on the marking. In CL-1 the individual sentence phrases are evaluated, in CL-2 the results of the marked conditions are displayed and CL-3 shows a completely evaluated causal relationship.

The given example illustrates the influence of the specification degree on possible evaluation options. For basic analysis or early system development stages, lower and less time-consuming evaluation levels are suitable. However the tool also includes more complex options of evaluation. Though increasing complexity requires an increasing effort, evaluation and validation of entire requirements is possible.

5.4 Qualitative surveys

With two planned qualitative surveys different aspects of the approach are to be evolved. In the first part, the 4-level markup language is examined in more detail. The following questions are interesting: How many requirements can be expressed with the language? How many tags are possible within a request? How many requirements can not be covered by the simulation?

The second part focuses on the presentation of the results. Here, test engineers evaluate which values should be displayed in the specifications and how a good representation of the results is possible. The results of the surveys have a direct influence on the development of the approach.

6 Progress

According to a literature search in the area of simulative test validation and information extraction from requirement documents, gaps in the processing of natural language requirements could be identified. Subsequently, based on existing formalization requirements, a novel markup languange was developed. For easy user application, a tool has been developed which reads in natural language requirements, makes the developed language usable and makes markings possible. We presented a first prototype at this year's REFSQ conference.

At this point in time, the marking process and the mapping are manual steps. Neural networks show promising results in previous work in requirements engineering. The automated detection of states, values and events with neural networks will also be investigated in the future. The engineer then gets a pre-marked specification and only needs to confirm the suggested markings.

In the future, the mapping should be integrated into the tool. An automated search for signal names that should support the engineer is planned. Building on this, the mapping can be fully automated using methods of machine learning. Today, the evaluation process of log data is already fully automated, but only limited to the evaluation of a simulation run. Based on this, the comparison of several log data sets should be the focus of research. Qualitative surveys aim to identify possible ways of presenting the results

The creation of scenarios of a long-term simulation is supported with presented approach and partially automated. The selected requirements have a direct influence on the scenario generation. It is planned to compare two test methods for identifying errors and evaluating system parameters. For this purpose, a standard test process of a system function from the automotive industry is compared with a traffic simulation. In both cases the same controller will be used to compare the test methods in terms of their duration, cost and identified errors. An accompanying survey is planned to evaluate the context information (frequency of function triggering, simulation parameters, etc.).

Subsequently, the approach, in addition to the automotive sector, will be evaluated in a second domain. A second large area of application for simulations is software development in the smart factory area. Here the complexity of the scenario is very limited but due to the increasing networking of the systems, the focus is more on the simulation of message transmission.

References

[BBEK11] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility. In *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, volume 42, 2011.

[BEF11] Z. A. Barmi, A. H. Ebrahimi, and R. Feldt. Alignment of requirements specification and testing: A systematic mapping study. In *IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2011.

[BLC+] Barrett R. Bryant, Beurn-Seuk Lee, Fei Cao, Wei Zhao, and Jeffrey G. Gray. From natural language requirements to executable models of software components.

[BOS16] J. Bach, S. Otten, and E. Sax. Model based scenario specification for development and test of automated driving functions. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 1149–1155, 2016.

[BRSM16] G. Bagschik, A. Reschka, T. Stolte, and M. Maurer. Identification of potential hazardous events for an unmanned protective vehicle. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 691–697, 2016.

[DRC+17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[EVFM16] J. Eckhardt, A. Vogelsang, H. Femmer, and P. Mager. Challenging incompleteness of performance requirements by sentence patterns. In *IEEE International Requirements Engineering Conference (RE)*, 2016.

[FM92] Kevin Forsberg and Harold Mooz. The relationship of systems engineering to the project cycle. *Engineering Management Journal*, 4(3):36–43, 1992.

[IK14] Masao Ito and Koichi Kishida. An approach to manage the concept phase of iso 26262. *J. Softw. Evol. Process*, 26(9):829–836, September 2014.

[MKK15] P. Minnerup, T. Kessler, and A. Knoll. Collecting simulation scenarios by analyzing physical test drives. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2915–2920, Sep. 2015.

[MWHN09] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak. Easy approach to requirements syntax (ears). In *2009 17th IEEE International Requirements Engineering Conference*, pages 317–322, 2009.

[PBKS07] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner. Software engineering for automotive systems: A roadmap. In *Future of Software Engineering, 2007. FOSE '07*, pages 55–71, 2007.

[Sch11] Björn Schünemann. V2x simulation runtime infrastructure vsimrti: An assessment tool to design smart traffic management systems. *Computer Networks*, 55(14):3189–3198, 2011.

[vNCNL$^+$09] Kilian von Neumann-Cosel, Mirko Nentwig, Daniel Lehmann, Johannes Speth, and Alois Knoll. Preadjustment of a vision-based lane tracker - using virtual test drive wihtin a hardware in the loop simulator. In *Proceedings of the Driving Simulation Conference Monaco*, 2009.