COPING WITH COMPLEX DRIVING SCENARIOS: EXPLORATORY SCENARIO DESIGN

Julian Schindler [1], Tobias Hesse [1]

(1) : German Aerospace Center, Lilienthalplatz 7, 38108 Braunschweig, Germany
+49-531-295-3510
{julian.schindler, tobias.hesse}@dlr.de

Abstract – The design of current scenarios in driving simulators can already be very challenging. It is assumed that new components of driving simulation, like coupling of simulators or the introduction and addressing of additional agents (e.g. pedestrians, cyclists or communicating infrastructure like Road Side Units) will aggravate this problem. Therefore, the issues of current scenario design have been analysed and recommendations have been extracted leading to a new suggested approach of scenario design. This new approach is driven by tools promoting the collaboration of the people involved in scenario design. Being part of a PhD thesis, this paper describes both the process and the needed tools, focussing on the operation of a multi-touch table guiding through the design.

Key words: Exploratory Scenario Design; driving simulation; collaborative platform, multi-touch table.

1. Introduction

Performing simulator studies in driving simulators is motivated by very different things, from functional testing via psychological testing to training or plain demonstration of technology. Driving simulators therefore make use of driving scenarios with different content, but mostly consisting of phases of free driving in traffic flows which should be as realistic as possible interrupted by phases with special behaviour of any involved agents. An agent could be movable like cars, trucks, cyclists or pedestrians, but also stationary like a traffic light or a Road Side Unit. The special behaviour of an agent, e.g. a strong braking of a car ahead of the ego vehicle, is used to force a special behaviour of the ego driver, e.g. by utilizing the function to be tested.

Olstam & Espié [Ols1] already described the alternation of the phases by introducing the Theater Metaphor, in which phases of "Everyday life" driving are interrupted by phases in which the automated road users have to follow certain manuscripts with special behaviour ("Play" on the "Stage"), see Figure 1.

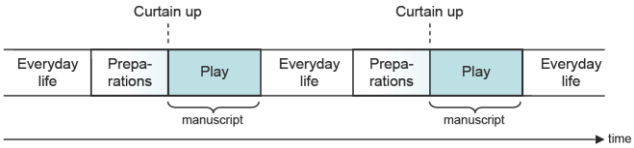

Fig. 1. Theater Metaphor by Olstam & Espié [Ols1]

In order to produce comparable results, the "Play" phases must consist of very well defined traffic behaviour leading to a strong behavioural restriction of all the involved agents, manifested in the presence of a manuscript. This contrasts to the mostly unrestricted "Everyday life" driving phases, in which all the agents may only be restricted in following road traffic regulations and optionally some additional advices by the study instructors like maintaining a minimum speed.

Therefore, a "Preparation" phase is needed used to migrate all acting agents from free driving to a well-defined starting behaviour, e.g. a defined position with a defined speed and acceleration, when the "curtain goes up" and the "Play" phase begins.

It is mandatory in the scenarios that these transitions have to take place unrecognizable by the ego driver, because the driver gets a pre-warning to the upcoming event when the behaviour of the involved agents changes too much or is not fully comprehensible to the ego driver. Therefore, the behaviour of the agents in the "Preparation" phases must be restricted as well, by providing limits of possible behaviours (e.g. a maximum acceleration) in the manuscript.

This results in manuscripts consisting not only of trivial information like the number of cars, used car models and the sometimes trivial special behaviour in the "Play" phases, e.g. the braking of a car ahead, but also consisting of fairly unknown parameters for the phase transitions in the "Preparation" phases and in some "Play" phases. Sometimes the value of parameters is unknown, but sometimes even selecting the right parameter is an issue. As a result, the parameters are frequently guessed, and therefore mostly not optimal. This introduces the following issues:

(1) The traffic situation and its parameters must be adapted iteratively in order to make a good look-and-feel.

(2) A wide range of situations must be tested to guarantee a smooth transition to the "Play" phases and the occurrence of the "Play" phases in any precondition.

In addition to this, scenarios are becoming more and more complex, as e.g. sophisticated Advanced Driver Assistance Systems (ADAS), the interaction between different kinds of agents or even the interaction between the drivers (and not the vehicles they are in) may be tested. Sometimes, on top of this, these tests also cover more complex sensor simulations, or Vehicle-to-Infrastructure/Vehicle-to-Vehicle (V2X) communication, probably resulting in additional complexity of the manuscripts.

As described in [Fis1], DLR's Institute of Transportation Systems (ITS) currently has many different simulators and test vehicles in service which may also be coupled so that various test drivers can participate in one scenario of the above mentioned complexity in the so called "Modular and Scalable Application Platform for ITS components" (MoSAIC). MoSAIC enables many new kinds of scenarios, but introduces the complexity of getting not only the automated agents to the correct positions and velocities in the "Preparation" phases, but also the human ones. As human drivers represent subjects in studies, they can mostly not be advised to follow many extra rules. Therefore, the human drivers have to be influenced by the surroundings, e.g. traffic lights, automated road users, or instructed human drivers, again resulting in a higher complexity of the manuscripts with lots of parameters not known in the beginning.

The problem now is that the scenario design process in companies or institutes in general, i.e. the process for specifying the manuscript, is very often not tailored for iterations or multiple test cases, esp. not in the case of rising complexity. Although there might not exist any specified process for this in many institutes or companies operating driving simulators, the generation of the manuscripts commonly follows a requirement-driven approach. This means, as shown in Figure 2, that the basic idea and the goals of a scenario are analysed in a first step in order to get a catalogue of requirements. The requirements are afterwards transferred into a rough plan of the scenario and the following creation of the 3D model and the implementation of the scenario. After the implementation, the scenario is getting tested. As this is a well-known procedure in other disciplines like systems or software engineering, it can be found that there are many parallels to common process models, esp. the Waterfall Model [Roy1]. The only main difference to this model is that refinements can be done by restarting any of the phases directly instead of moving up phase by phase. In addition to the often criticised linearity of this model, e.g. by [Liv1] or [Boe1], scenario design is very often challenged by the existence of two parties: One party – mostly consisting of people from the domain of psychology (esp. when performing psychological studies) – is analysing the needs of the scenario and describing the requirements of it. In the following, we therefore call this party the "requesters". The other party is responsible for the implementation (the "implementers") and therefore this party consists of people trained in the operation of manuscript editors or driving simulators. So both parties may lack a lot of knowledge of the other party, often leading to the specification of incomplete requirements and to the implementation of scenarios not complying with the initial needs.

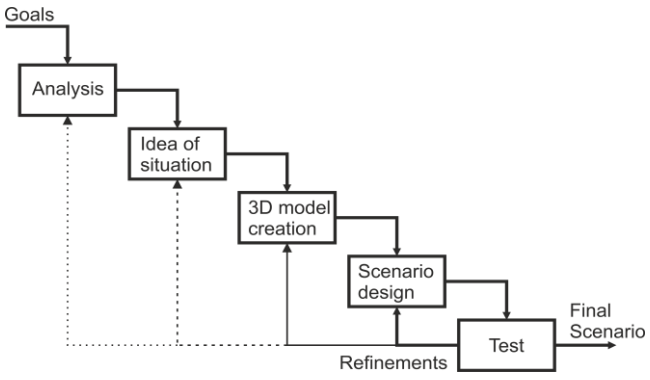

Fig. 2. Waterfall-like approach of common scenario design

As a result, the testing of the scenario script very often fails, and large refinements of the scenario design have to be performed. Due to

the waterfall-like structure of the process, these refinements are expensive and time consuming. Catalogues of requirements have to be adapted, the meaning of situations have to be explained.

There are several possibilities to cope with the occurrence of these iterations: On the one hand by changing the process and on the other by using proper tools. This paper addresses both, by introducing a new scenario design process and tools for enabling it.

2. Ideas for a better process

As mentioned before, the missing tailoring to possible iterations is not a new phenomenon, but has been widely discussed in systems or software engineering. So it is not surprising that various approaches exist for solving this issue, e.g. prototyping [Flo1 or Ril1] or the spiral model of Boehm [Boe1] (which is also based on the prototyping approach).

In prototyping, the goal is the creation of horizontal prototypes (e.g. mock-ups without function) or vertical prototypes (e.g. parts of the complete target system) which can be tested by users before the complete system has to be built. Furthermore it describes how to get closer to a final product, e.g. by rapid, evolutionary or incremental prototyping [Ril1].

Adapted to the scenario design this means that the target scenario needs to be decomposed into smaller parts, which can be implemented in a prototypic way. As a "scenario mock-up without functionality" can only be hardly imagined, we classify scenario prototypes as vertical prototypes. They therefore represent a part of the whole scenario, e.g. one special situation during one "Play" phase. The kind of the prototypes may be rapid (meaning that a developed prototype may be thrown away after instantiation) or evolutionary (meaning that a developed prototype will get more and more precise in each iteration). As a result, several prototypes may exist for the several parts of the scenario which can be merged into one scenario as done in the incremental prototyping [Ril1].

One major problem of scenario design is that parameters like acceleration or time-headways, or the limits of parameters, very often must be guessed or approximated iteratively, as their effect can only hardly be imagined. A misfit can only be recognized when testing the prototype in action. The same is true for the creation of a good look-and-feel in the "Preparation" phases, where a wide range of initial conditions has to be tested. In some situations not only the approximation of the value of any parameter, but the choosing of the correct parameter itself is already challenging. Both aspects in general are addressed in the field of exploratory research [Ste1]. This research has also been applied to the development of ADAS as "exploratory design"; see [Fle1] or [Sch2]. In the exploratory design, the complete space of design possibilities, the "Design Space", is reduced systematically in iterations in order to find an optimal design. It makes use of a method called the "integrated testing" where design alternatives get tested step by step by driving in a simulation before any line of code has been written.

It therefore makes use of a tool called the "Theater System" [Sch1], in which one ADAS designer playing the role of a potential user of a future ADAS is sitting in a simulator with active inceptors (steering wheel, pedals or side-sticks). The active inceptors are coupled to a second set of inceptors, operated by another designer playing the system, the so called *confederate*. The confederate now can directly ask how e.g. a haptic feedback should feel like while driving through the situation. As the inceptors are coupled, the driver can directly feel the actions of the confederate. Iteratively the designers may also change their roles and can therefore express their intentions directly. When a good solution has been found for any tiny step, this step is implemented quickly, and directly validated in the simulation. Thanks to tool support the implementation can be done (mostly) in seconds, so that crisp ADAS designs can be reached very fast.

The approach of integrated testing would strongly benefit the scenario design, as it enables quick iterations of prototyping with high performance and emerging scenarios of high quality.

Nevertheless, the general prototyping approach only describes how to get to a final product in smaller iterative steps, but it does not define the means used for the creation. As described, there often are two parties involved in the scenario design process, the "requesters" and the "implementers", both often with different backgrounds. Bringing both parties closer to each other would largely benefit the design process. The party of the "requesters" can be seen as "users" in a wider sense, as they want to use the scenario for the performing of their studies. Therefore, when using the term of "user", an analogy to systems engineering can

easily be found, esp. by looking at Participatory Design (PD) [Ken1], where users are directly integrated in the design of systems. This has already been done in ADAS design by the "Theater System", as a potential user can directly participate instead of a designer playing the role of him. As the changing of the roles is still possible, the designer is able to directly feel the interaction a potential user has in mind.

The participation of the "users" is a very valuable step in systems engineering. Nevertheless, it is criticised to be possibly ineffective, as the users cannot be professionals and therefore lack knowledge and tend to reinvent the wheel. Kensing and Blomberg [Ken1] state that "…design professionals need knowledge of the actual use context and workers [i.e. users in this context] need knowledge of possible technological options".

Applied to scenario design user participation as stated in PD would mean to simply let the psychologist create the scenario alone. Although scenario design has changed a lot in the last years from plain scripting to the common use of scenario editors with Graphical User Interfaces (GUI), using those tools and knowing about all the implemented features is still not fully intuitive and needs to be trained. So indeed this option would be ineffective.

The ineffectiveness in general is a well-known problem already addressed in systems engineering, e.g. in the Cooperative System Development Process (CESD) [Gro1], where "existing technological concepts and systems […] can be brought in as thought-provoking artefacts in cooperative workshops extending the participants' understanding of alternatives as well as current practice". Applied to scenario design this would mean to show the users the alternatives they have when designing the scenario.

But Grønbæk et al. [Gro1] also go a little further: "To design cooperatively, to develop visions of technology in use, it is important to give these visions a form that allows users to apply their knowledge and experience as competent professionals in the process."

Kensing and Blomberg [Ken1] therefore interpret the mentioned form as the requirement of "access to adequate prototyping tools" leading to the statement that "the development of tools and techniques is a key focus for PD projects". Applied to the scenario design this means that using special tools beyond any GUI scenario editor may enable a better cooperation between professionals and users, i.e. implementers and requesters. Proper Tools may benefit the whole process of scenario generation. These tools should bring the requester and the implementer closer together so that on the one hand the requester understands which possibilities and short-cuts exist when designing scenarios and on the other hand the implementer gets a better understanding of the broader context of the scenario and the reasons for the specified requirements.

Ideally, these tools will also support the former mentioned approach of integrated testing.

In summary, a new process for scenario design therefore should cope with the following three basic recommendations for complex scenario design:

(1) Prototypes for each part of a scenario should be created instead of complete scenarios

(2) Prototypes should be created in quick iterations, best in a form of integrated testing, as this enables the exploration of various parameters and alternatives, promising scenarios of high quality.

(3) Requesters of the scenario should participate in the scenario design actively, best by cooperating directly with the implementers. This is reached by the introduction of proper tools.

One suggestion for such a scenario design is described in the following.

3. The Exploratory Scenario Design Process

The Exploratory Scenario Design Process as shown in Figure 3 starts in the same way as regular processes, i.e. by the initial definition of the goals of the target scenario. These goals then have to be transformed into a rough idea, how a test scenario might look like. The transformation is done in an analysing phase by a decomposition of the goals into use cases, user stories and single requirements. In this context, use cases describe the general situation, e.g. being on a two-lane highway with a speed limit of 120 km/h and mixed traffic of low density.

User stories than describe the individual things happening in the use cases, e.g. a close overtaking of a slower truck when there is upcoming traffic in the blind spot of the ego car. Each "Play" phase consists of one or more consecutive user stories.

In this example, an emerging requirement would be that there is a slower truck in the

lane of the ego car. Another would be that in that precise moment there has to be another car in the blind spot.

When the requirements have been specified, they are transferred into a basic idea of how the final scenario might be composed. Afterwards, a phase of preparation is started. In this phase, e.g. the 3D model of the virtual landscape is generated and a set of road users of the needed type and density is provided to the streets in order to make the desired look-and-feel of everyday life situations.

The resulting basic scenario is afterwards set up in the simulator. In order to reduce artefacts of different simulators, the target simulator should be the one where the study will take place, if possible.

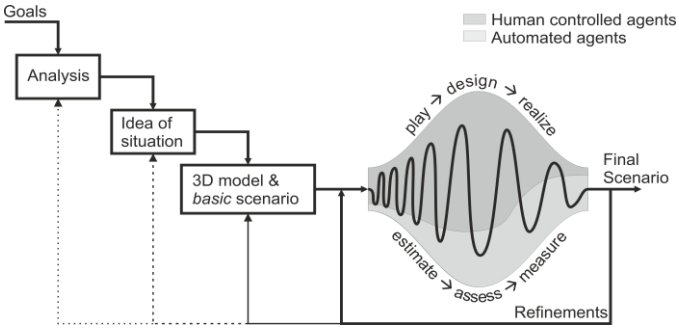

Fig. 3. Exploratory Scenario Design

As shown in Figure 3, at this point the integrated testing is started. As analogy to the "Theater System" approach when designing ADAS online in the simulation, the same can be done in the design of scenarios. The different agents involved in a situation can be controlled manually by connecting additional control entities like simulators or simple game wheels to the target simulator. In this way, humans play the interaction between the vehicles on the track before any single line of scenario code has to be written. When the involved persons agreed on a played situation, the scenario script is created directly from the manually driven test runs.

The exact procedure of the integrated testing in the scenario design is as follows:

First, the basic scenario is loaded and it is jumped to the time and/or place where the first "Play" phase is supposed to happen. Each of the agents which are going to play a specific role in the first user story of this phase, including the ego car in the targeted scenario and any other agent, is assigned to a manual driver and a control entity. One of the drivers may also be the requester of the scenario, who now has the direct ability to show his intentions. Afterwards, the scenario is started and the movements of all agents are recorded.

One special thing about the recording is that not only the trajectory of the agents is recorded but also events like indicator signals or inceptor movements. When a user story has been recorded, it can be replayed. The recording may be discarded and repeated when somebody (and esp. the requester) is not satisfied with the result.

In case of full satisfaction the recorded data is analysed by software. This step is necessary because a simple replaying of the trajectories during the study will not serve all possible behaviours of the ego drivers in the study. Just imagine a fast driving and a slow driving participant in a study: When cars simply follow trajectories the resulting situation will be completely different, as the behaviour of each agent has an impact on the behaviour of the others. Therefore, the data esp. of the movable agents must be brought to a more abstract level. This is done by categorizing the data into driving manoeuvres. Afterwards, the events not fully complying with the currently driven manoeuvre are marked. The manoeuvres and the marked events per agent are presented in form of a timeline of the run in a graphical way. An example for this with three agents is shown in Figure 4: All involved movable agents are classified as driving in the manoeuvre "follow lane" at the beginning (t_0). When the blue car – let us say18.3 meters in front of the red car - started to brake, the driver of the red car did a movement of the steering wheel resulting in a swerving of his car. The swerving does not comply with the manoeuvre and therefore it gets marked (the highlighted red area shortly before t_1). Afterwards, the red and green car continue driving, the blue one has stopped (t_2).

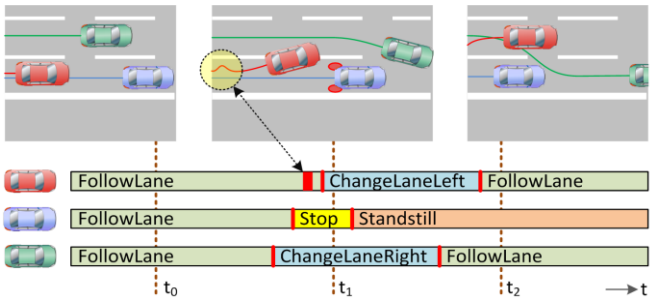

Fig. 4. Example of a scenario analysis output. Situations are marked where car behaviour changes. The upper images show how the situations and the just driven trajectories looked like at the given timestamps of t_0, t_1 and t_2. The yellow circle highlights a marked swerving situation just before t_1.

The people involved in the scenario design now have the direct ability to discuss the events. Events occurring unintendedly can be unmarked. All the other events have to be linked to triggers. Triggers can be any logical combination of one or more other events, manoeuvre changes or any thresholds of any other available parameter, e.g. distances/time headways/time to collisions to other agents or infrastructure, durations, indicator signals etc.

In the above example, the scenario designers may decide if the swerving has been intended or not. When it has been intended, it has to be linked to one or more triggers, possibly to the braking of the car ahead and the distance to it. Also the manoeuver changes have to be linked to certain triggers. Additionally, the triggers can be specified with tolerances or limits of thresholds. E.g. "braking" may be defined as "braking with more than 0.4g" or "distance" may be defined as "between 10 and 30 meters".

Furthermore, not only the trigger itself can be specified with tolerances; also the event happening because of the trigger may be performed with tolerances adapting to the surrounding. In the example, you may link the amplitude of the swerving to the width of the current lane. Another example would be the linking of the length of a triggered lane change (like the ones of the red and green car in the example) to the surrounding traffic situation.

The setting of triggers has to be done for all the not movable agents as well. Traffic light phases may be linked to events happening in the simulated world or simply to timing models.

The general advantage of the abstraction is that the intended behaviours of the agents can be separated from the unintended easily. The key-behaviour in the scenario is extracted and uncoupled from trajectories, allowing a range of initial situations to be tolerated for triggering. The abstraction of the situation furthermore enables the transferability of manually driven scenarios to automated car behavior, a necessary step for creating a script of the scenario. Driving the situation manually gives a good overview on the parameters to choose as triggers and their values.

Each user story of each "Play" phase, i.e. each situation or prototype, can be recorded consecutively in this way.

Nevertheless, sometimes situations occur, where more agents are involved than simulators or controllers are available. In this case, another way of scenario creation must be chosen, as parallel driving is not possible. This can be done by either manually script parts of the scenario so that some of the agents are controlled automatically, or by recording the behavior sequentially, or by switching between the currently controlled agents while recording.

When all situations of a scenario meet the requirements, the whole scenario script is generated, so that it can be used by single ego drivers. This procedure is also applicable for scenarios with multiple ego drivers or agents of different type.

In any case, a crisp scenario design will emerge after a short phase of preparation, as parameters and thresholds are not needed to be guessed, but are directly tangible in the simulation. Requesters of scenarios can directly feel how parameters must be chosen to create a desired output.

Therefore, the mentioned approach already copes with the three basic recommendations for complex scenario design. Nevertheless, it might be difficult for the design team to keep track on the proceeding of the scenario creation. Additionally, it would be beneficial if the scenario designers are able to discuss the recorded scenarios in detail in a collaborative way, something not so easy in the limited room available in some driving simulator cabins. Furthermore, not enough control entities might be available.

To account on these issues it is proposed to make use of an additional tool, described in the following.

4. A Multi-Touch-Table as central tool in the Exploratory Scenario Design Process

A new tool has been created to cope with the mentioned issues. It has been found (see [Sch3] for details) that the ideal basis for such a tool is a multi-touch-table showing bird views on the scenario. At DLR ITS an Ideum MT 55" Multi-Touch-Table with a maximum of 32 possible parallel touch points has been chosen for this task.

The software running on the table is a self-developed scenario editor with a graphical user interface focussing on maximum collaboration and intuitive control. In Figure 5, the table is shown running attached to the three small simulator entities of the DLR ITS MoSAIC Laboratory. Up to six bird-views of the situation are shown on the table in parallel, each of it centring on a freely selectable agent of the scenario. Each bird-view can be controlled by using standard gestures as known from current smart-phones, e.g.

zooming with two fingers moving away from each other, rotating with two fingers doing a circular movement.

Fig. 5. Scenario preparation around the Multi-Touch-Table at the DLR MoSAIC Lab

Additionally, it is possible to control the centred agents directly on the table. The controlling of movable agents is possible in three ways according to the three hierarchical layers of the driving task from Donges [Don1]: It is possible to specify and change the route of each agent (navigational layer), to change the actually driven manoeuver (guidance layer), and to directly control the movements of the agent (control layer). The route of each road user can be specified by dragging waypoints into the scenery. Manoeuvers are switched by selecting them in a small menu displayed near the car. The direct control is done in the following way as shown in Figure 6: first, one finger is put on the displayed agent who has to be controlled. Afterwards, another finger is put where it is supposed to head. This second point is also the neutral position for acceleration, so moving the fingers apart will accelerate the agent, moving them towards each other will cause deceleration.

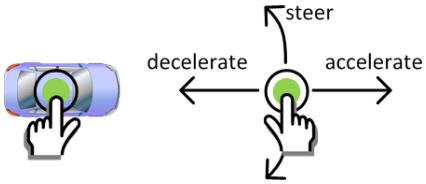

Fig. 6. Controlling a moving agent with touch gestures

Agents which are not movable are controlled similar as the controlling of manoeuvers, i.e. by small menus, e.g. showing the phases of the traffic lights.

Another aspect of the multi-touch table is that it allows the controlling of the scenario recording and basic functionality like 3D model loading, vehicle insertion etc. Therefore, dialog-boxes and menus are shown on the screen. Due to the fact that the designers are supposed to stand around the table, the position and even the orientation of the menus had to be freely adjustable. Because of this, each menu can be picked, rotated and resized with the former introduced gestures known from smart phone interaction. This makes it possible to work on a menu and to "hand it over" to another person on the other side of the table. As all the standard windowing toolkits (at least FLTK, GTK, QT) do not have the ability to perform such actions easily, it has been chosen to create a new toolkit based on osgwidgets, a part of OpenSceneGraph [Wan1]. The creation of the windowing toolkit has been discussed in detail in [Hes1].

The same menu structure can be used to directly access and manipulate all available parameters of the agents, e.g. by smoothing the recorded values, setting some initial speeds, selecting the 3D model of the agents, or by introducing threshold values etc.

Finally, the output of the scenario recording can be displayed similar to the example in Figure 4. As described, the manoeuvres and the events per agent are presented in form of a timeline of the run. The discarding, the setting of triggers of events or the modification can be done graphically on screen. The resulting script can be exported into a human-readable XML scenario script files and used for testing in the simulator.

The multi-touch table application is currently (May 2014) under development. The work on the windowing toolkit and the support of multi-touch gestures is already finished, the implementation of the scenario recording and analysis has just started and is targeted to finish by the end of 2014. Therefore, the approach of Exploratory Scenario Design has not been tested practically in any project, and there is currently no data on increasing efficiency available. As soon as the tool development is finished, the performance will be measured.

5. Conclusion

This paper has described the issues of current scenario design and the assumed aggravation of them in the near future. A new approach, the Exploratory Scenario Design, has been introduced which focusses on the direct integration of the people normally only creating requirements for scenarios into the process of the detailed design of the scenario itself. It has

been shown that the methods of prototyping and "integrated testing" used in the Exploratory Scenario Design are strongly benefitting the design of complex driving scenarios in terms of time needed for the preparation and quality of the resulting scenario. Furthermore, the integration of a multi-touch table as central tool and enabling technology for the Exploratory Scenario Design has been introduced and described in detail, including some of the available multi-touch gestures.

The utilization of the design process, the methods and the proposed tools will enable the coping with complex driving scenarios of all kinds in the upcoming future.

6. References

[Boe1] Boehm, B.W. "A Spiral Model of Software Development and Enhancement" In: *IEEE Computer*, Vol. 21, Issue 5, pp. 61-72, 1988

[Don1] Donges, E. "Aspekte der aktiven Sicherheit bei der Führung von Personenkraftwagen". In: *Automobil-Industrie*, Volume 27, pp. 183–190, 1982.

[Fis1] Fischer, M., Richter, A., Schindler, J., Plättner, J., Temme, G., Kelsch, J., Assmann, D., Köster, F. "Modular and scalable driving simulator hardware and software for the development of future driver assistance and automation systems", Paper submitted to the *Driving Simulation Conference*, 2014.

[Fle1] Flemisch, F., Schindler, J., Kelsch, J., Schieben, A., Damböck, D. "Some Bridging Methods towards a Balanced Design of Human-Machine Systems, Applied to Highly Automated Vehicles". *Applied Ergonomics International Conference*, Las Vegas (USA), 2008

[Flo1] Floyd, C. "A systematic look at prototyping". In: *Approaches to prototyping*. Springer Berlin Heidelberg, pp 1-18, 1984.

[Gro1] Grønbæk, K, Kyng, M., Mogensen, P. "Toward a Cooperative Experimental System Development Approach". In Kyng, M., Mathiasen, L. (Eds.): *Computers and Design in Context*, MIT Press, pp. 201-238, 1997

[Hes1] Hesse, S. "Development of a Graphical User Interface for the Explorative Design of Driving Scenarios", Bachelor thesis, 2014.

[Ken1] Kensing, F., Blomberg, J. "Participatory Design: Issues and Concerns". In: *Computer Supported Cooperative Work*, Vol. 7, Kluwer Academic Publishers, pp. 167-185, 1998

[Liv1] Liversidge, E., "The Death of the V-Model", 2005, Available at http://www.harmonicss.co.uk/index.php/hss-downloads/doc_download/12-death-of-the-v-model, Last accessed May 2014

[Ols1] Olstam J., Espié, S. "Combination of autonomous and controlled vehicles in driving simulator scenarios". In Andrea Benedetto (Ed.): *Advances in Transportation Studies*, University Roma Tre (21), pp. 23–32, 2010.

[Ril1] Riley, D. D. "Software Lifecycle Models". 2010, Available at cs.uwlax.edu/~riley/CS741Sum10/lectures/2_LifeCycles.pdf, Last accessed May 2014

[Roy1] Royce, W. "Managing the development of large Systems". In: *IEEE Wescon*, pp 1-9, 1970.

[Sch1] Schieben, A., Heesen, M., Schindler, J., Kelsch, J., Flemisch, F. "The theater-system technique: agile designing and testing of system behavior and interaction, applied to highly automated vehicles". In Albrecht Schmidt (Ed.): *1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pp. 43–46. 2009

[Sch2] Schindler, J., Temme, G., Schieben, A., Flemisch, F. "Exploratory design of a highly automated system for entering the expressway". In: de Waard, D., Axelsson, A., Berglund, M., Peters, B., Weikert, C. (Eds.): *Human factors. A system view of human, technology and organisation*. Maastricht: Shaker Publishing, pp. 201–216, 2010.

[Sch3] Schindler, J., Kelsch, J., Heesen, M., Dziennus, M., Temme, G., Baumann, M. "A Collaborative Approach for the Preparation of Cooperative Multi-User Driving Scenarios". In: *10. Berliner Werkstatt Mensch-Maschine-Systeme*, Berlin, Germany, 2013.

[Ste1] Stebbins, R. A., ed. "Exploratory research in the social sciences". Vol. 48, Sage, 2001.

[Wan1] Wang, R., Qian, X. „OpenSceneGraph 3 Cookbook", Packt Publishing, 2012.