A Test-Oriented HMI Specification Model for Model-Based Testing of Automotive Human-Machine Interfaces

Linshu Duan
AUDI AG
85055 Ingolstadt
Germany
linshu.duan@audi.de

Heinrich Hussmann
Ludwig-Maximilians-
Universität München
80333 Munich Germany
heinrich.hussmann@
ifi.lmu.de

Alexander Höfer
AUDI AG
85055 Ingolstadt
Germany
alexander.hoefer@audi.de

Abstract: While model-based testing is widely-used for function tests, testing the human-machine interface (HMI) remains a manual, demanding and time consuming task. Numerous research efforts have been addressing model-based HMI testing in the last years. However the existing approaches cannot serve our test purposes or needs. In the research project at AUDI AG a model-based testing approach is proposed for testing both the logical behavior and the graphical interface of automotive infotainment HMIs. In this paper we focus on menu behavior tests and introduce some important details of the test-oriented specification model. A test-oriented HMI specification model describes the expected menu behavior and contains needed information for the test generation. Methods of test generation and execution will be introduced briefly.

1 Introduction

The complexity of infotainment HMIs is growing with the functionalities of infotainment systems [Sys05]. A user interface giving a faultless experience is one of the most important requirements of today's infotainment systems. During the development phases, 50 percent of the time and more than 50 percent of the costs are expended for testing [MS04]. While testers of non-HMI applications have been enjoying the convenience of tools to automate their tests, GUI testers still toil in their tedious works. Difficulties of HMI testing compared to function tests are fully discussed in [CYM10]. In infotainment testing area, HMI tests are usually distinguished from function tests. Accordingly we distinguish HMI errors from function errors. A function error is e.g. deleted entries are still existing, while HMI errors can be classified in three categories: menu behavior error, widget behavior error and display error. Accordant tests detecting these errors are menu behavior tests, widget behavior tests and display tests [DHH10]. In [Ben10] and [Ben07] task models are proposed for generation of integration tests which belong to functions tests. In our research a model-based approach for testing the infotainment HMI is proposed. Testing goals, specific coverage criteria and evaluation methods are introduced in previous paper [DHH10]. In this paper we focus on the menu behavior tests and introduce in Section 3 the test-oriented specification model for the menu behavior tests. Test generation and execution will be introduced in Section 4. Firstly related researches are discussed.

2 Related Work

In the previous paper [DHH10] some related works in GUI application testing were introduced. It was explained why these existing approaches cannot fulfill our testing demands. Automotive manufacturers and suppliers have also been engaged in model-based testing of infotainment HMIs for some years. In [Fle07] a model-based HMI prototyping tool is suggested from the prospect of a supplier to manufacturers for a model-based specification. One of the motivations is that a model-based HMI specification should allow model-based HMI testing. In [SBK05] discontinuities are regarded as the reason why model-based testing is still not accomplished for infotainment HMIs. Numerous stake-holders participate in the development process and each of them has his own perspectives, approaches and tools. To resolve this problem a HMI framework, which is quite similar to the one in [Fle07], is proposed for the whole HMI development process: HMI specification, implementation and testing. In [GB09] domain specific languages (DSLs) are defined for each layer of an infotainment HMI. That means the HMI specification is composed of models described with the defined DSLs. Such specification should allow automated test generation and execution.
To accomplish model-based testing for infotainment HMIs, the mentioned approaches are looking for solutions in the HMI framework. However our experience has shown that the absence of a suitable HMI framework is not the core problem. There exist varied problems, e.g. the process problem and designer vs. IT-engineer problem [PVH07]. In our opinion, another problem is the lack of a complete model-based HMI testing approach covering the whole testing process, which can serve as a pilot solution. A pilot solution, which is proved to be possible and useful, is the best motivation for the management to resolve the process problem or other no-technical problems.

3 Test-Oriented HMI Specification Model

A test-oriented HMI specification model is a model which describes the expected HMI behavior and contains sufficient information for testing. Depending on the HMI development process, a test-oriented HMI specification model could be created by HMI designers and test engineers corporately or only by test engineers in the test generation phase on the bases of a normal HMI specification. A normal HMI specification is a product of the HMI design phase, it could be informal or model-based. As shown in Figure 1, a test-oriented HMI specification model can be made with the information contained in the normal HMI specification and additional information needed for testing which is defined by the test engineers. In this case the test-oriented HMI specification model does not have to own a visual form, it could be a composition of information in the memory. We have created a normal HMI specification and a test-oriented specification model with a HMI prototyping tool in order to show them visually.

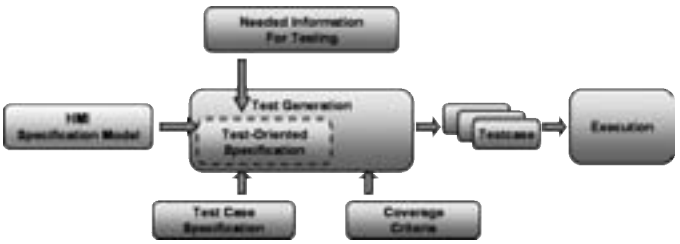

Figure 1: A Potential HMI Testing Process

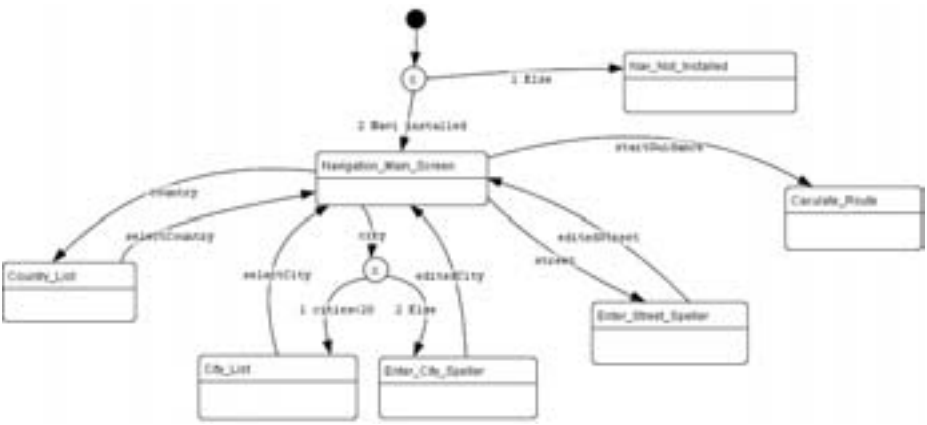

Figure 2: Simplified Cutout of a normal HMI Specification

Figure 3: Screen and Widgets

3.1 A normal HMI Specification

An HMI specification is usually composed of two parts according to the two layers of an HMI: i)Statecharts in Figure 2 defining the menu behavior. The menu behavior is how the HMI switches from one screen to another as reaction to incoming inputs. ii)Graphical elements i.e., screens and widgets shown in Figure 3 defining the HMI looks.

Figure 2 demonstrates a strongly simplified cutout from the statecharts of a normal HMI specification. The statechart cutout describes the simplified menu behavior of the feature navigation. Each state in the diagram is a view state, i.e., associated with a defined screen. As soon as the HMI enters a view state the according screen will be displayed. The HMI

stands in *Navigation_Main_Screen* when the feature navigation is accessed. Event action-reaction details of widgets are contained in the graphical layer, but not in the statechart. The first left outgoing transition in the diagram is executed by the event *country* which is fired by the button widget as soon as the user has pressed the button. The HMI returns from the state *Country_List* to the main screen, when the user has selected a country. From the main screen it is possible to enter a city as destination. If the selected country has less then 20 cities, the screen *City_List*, from which the user can directly select the expected city, will be entered. Otherwise, the user will be led to *Enter_City_Speller* which offers a speller to the user for character inputs. So the next state after the event *city* depends on the selected country and the used navigation database. By a complete user input the route guidance can now be started. The HMI accesses the screen *Calculate_Route*.

3.2 A Test-Oriented HMI Specification Model

A normal HMI specification defines only the expected behavior of an HMI. Tests can not yet be generated from this specification. In this paper two mostly important information needed for testing will be introduced. Firstly, in a normal specification many important conditions are not accessible for the test generation. The statechart in Figure 2 defines the possibility to reach *City_List* from the main screen. However, the condition to do this is hidden in the widget. The widgets used in the specification phase are usually implemented by the supplier and are available as program libraries. The hidden logic is that as long as the widget city is active, it fires the event *city* as reaction to the keystroke. The widget city is only active if the underlying application, which is not a part of the model-based specification, has evaluated the earlier selected country as valid and then set the widget active. This condition is hidden in the widget library and is not accessible for test generation. Similarly start guidance can only be performed if the supplied destination was valid. Without these conditions many invalid tests will be generated, e.g., starting the route guidance immediately from the main screen without any destination inputs. Secondly, contents of user inputs e.g., the city name to be entered do not exist in the specification. Generated tests are not complete without these user input contents.

Figure 4 shows the result after adding the lacking information into the specification. Actions and conditions are inserted into some of the transitions. For example, to be able to enter a city the condition *countrySelected==true* must be satisfied. This condition can only be satisfied by the action of the transition from *Navigation_Main_Screen* to *Country_List*. This condition constraints that the country must be selected at first. The conditions will be considered by the test generation going to be introduced in the section 4. In this way, it is ensured to generate correct and usable tests and to avoid invalid tests. Some self-transitions indicating the needs for user input contents are inserted into the diagram. Section 4 will introduce how user inputs can be integrated into generated tests.

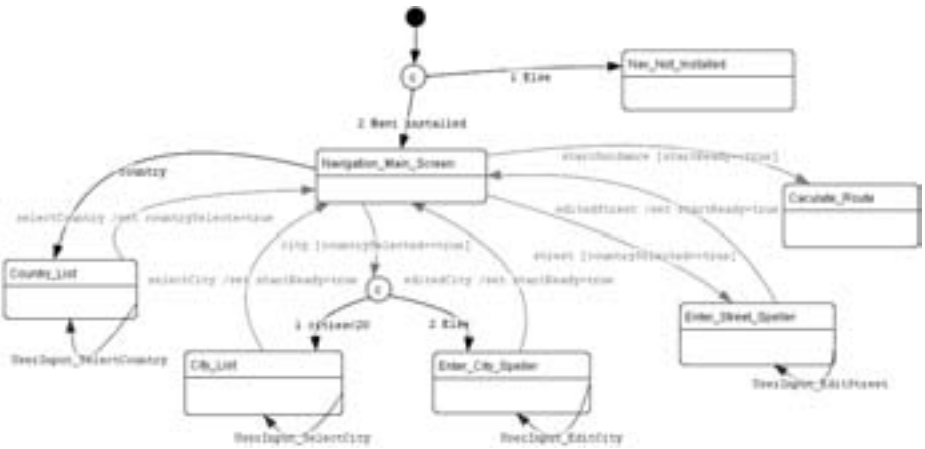

Figure 4: Sample of a Test-Oriented HMI Specification Model

4 Test Generation and Execution

In order to generate tests, test case specification has to be defined by testers and a test coverage criterion has to be selected for the forthcoming generation. Test case specification defines the contents and ranges of tests to be generated, e.g., the states and transitions a test should cover. Test generation algorithms need this information to select tests from a test-oriented HMI specification model. In our work, infotainment system specific coverage criteria are identified. Our previous paper [DHH10] has introduced them in detail.
Before a transition labeled with a condition is selected as the next test step, first of all it has to be verified whether actions of other transitions make the condition fulfilled. For example, to select the transition with event *city*, the transition with event *country* has to be selected as previous step. Transitions with *UserInput* events will be selected as steps before all other outgoing transitions from the same state. We explained that the event *country* is the reaction event fired by the widget country. So the test generation selects the action event *enter on widget country* as the test step instead of the reaction event *country*. Generated tests contain both test steps and expected screens or state names.
In the previous paper test tools for infotainment system function tests were introduced. Tests generated from the test-oriented HMI specification model can be executed automatically with the help of such test tools. Test actions performing needed user input contents have to be defined in the test tools before. They will be executed as sub-sequence before the rest of the generated tests is executed. Image processing is intensively used to locate the correct widget on the screen and verify whether the HMI stands in the correct menu as expected. Details of test execution, observation and verification will not be explained in this paper to avoid exceeding the paper range.

5 Conclusion

The purpose of our research is to find a model-based testing approach for infotainment HMIs. We focused on the testing of the logical behavior in this paper and introduced a test-oriented specification model which contains both the expected behavior of the HMI and sufficient information for testing purposes. Tests can be generated from the test-oriented specification model and automatically executed with the help of test tools for infotainment system function tests. In future papers, we will introduce details about testing the widget behavior, displaying effect as well as the test generation and execution. We believe that we have provided a practical and complete model-based testing approach for infotainment HMIs whose methods could also be used for other advanced HMIs.

References

[Ben07] Sebastian Benz. Combining test case generation for component and integration testing. In *A-MOST '07: Proceedings of the 3rd international workshop on Advances in model-based testing*, pages 23–33, New York, NY, USA, 2007. ACM.

[Ben10] Sebastian Benz. *Generating Tests for Feature Interaction*. Dissertation, Technische Universität München, München, 2010.

[CYM10] Tsung-Hsiang Chang, Tom Yeh, and Robert C. Miller. GUI testing using computer vision. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pages 1535–1544, New York, NY, USA, 2010. ACM.

[DHH10] Linshu Duan, Alexander Hoefer, and Heinrich Hussmann. Model-Based Testing of Automotive HMIs based on a Test-Oriented HMI Specification Model. In *Proceedings of the the 2nd International Conference on Advances in System Testing and Validation Lifecycle (VALID 2010), August 22-27 2010, Nice, France*. IEEE, August 2010.

[Fle07] Thomas Fleischmann. Model Based HMI Specification in an Automotive Context. In *Human Interface and the Management of Information. Methods, Techniques and Tools in Information Design*, volume 4557/2007, pages 31–39. Springer Berlin / Heidelberg, 2007.

[GB09] Holger Grandy and Sebastian Benz. Specification based testing of automotive human machine interfaces. In *GI Jahrestagung*, pages 2720–2727, 2009.

[MS04] Glenford J. Myers and Corey Sandler. *The Art of Software Testing*. John Wiley & Sons, 2004.

[PVH07] Andreas Pleuss, Arnd Vitzthum, and Heinrich Hussmann. Integrating Heterogeneous Tools into Model-Centric Development of Interactive Applications. In *MoDELS*, pages 241–255, 2007.

[SBK05] Reinhard Stolle, Thomas Benedek, and Christian Knuechel. Model-based Test Automation for Automotive Human Machine Interfaces. Technische Universität Berlin - Forschungsberichte der Fakultät IV, Bericht-Nr. 2005-1, ISSN: 1436-9915, 2005.

[Sys05] QNX Software Systems. Ready for the Future: Software Trends for In-Car Infotainment Systems. 2005.