Analysis, user-centered design and integration of a standalone simulation application into an existing business software suite

Olivia Waclik, Bakk.techn.

This page intentionally left blank.

Master's Thesis

Analysis, user-centered design and integration of a standalone simulation application into an existing business software suite

Olivia Waclik, Bakk.techn.

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

Supervisor:

Assoc. Prof. Dr. Andreas Holzinger, PhD, MSc, MPh, BEng, CEng, DipEd, MBCS

Graz, March 2011

This page intentionally left blank.

Masterarbeit

(Diese Arbeit ist in englischer Sprache verfasst).

Analyse, benutzerzentriertes Design und Integration einer Stand-Alone Applikation in eine bestehende Business Software Suite

Olivia Waclik, Bakk.techn.

Institut für Informationsysteme und Computer Medien (IICM),
Technische Universität Graz
A-8010 Graz, Austria

Betreuer:

Univ.-Doz. Ing. Mag. Mag. Dr. Andreas Holzinger

Graz, März 2011

This page intentionally left blank.

Abstract

AVL List GmbH, an Austrian based enterprise in the automotive sector, integrated a new simulation application (AVL InMotion) for hardware-in-the-loop tests of hybrid drives at early stages of the development process into an existing software suite. AVL started maneuver and event based testing at the test beds, which is supported by the application AVL InMotion. The application simulates virtual driving maneuvers at test beds and delivers results at the very beginning of the development process by analyzing different elements of combustion engines and hybrid systems.

At the beginning, the three-dimensional simulation software was represented by an external component used in the development process. This master's thesis focuses on the effects of this external application and how it can be integrated into an existing and quite complex software suite with special emphasis on user-centered design methods. The integration of such an application into an existing software system may have an impact on the usability of the entire framework. New elements possibly need to be redesigned in order to maintain the prior functionality of the suite and ultimately improve it. The research question of this master's thesis is whether the integration of a software component into a larger system improves its usability for users familiar with the existing system or not.

The findings of this thesis and the improvement suggestions based on them represent a highly beneficial basis for the further development of software products such as AVL InMotion in the company. This master's thesis emphasizes on the importance of usability activities, which AVL is planning on more extensively use in the field of software engineering.

Keywords
Usability, requirements engineering, prototyping, user interface testing

ÖSTAT-Topics

1161	50 %	1157	30 %	1140	20 %

ACM Classification
H.1, H.1.2, H.3.1, H.3.4, H.5.2

This page intentionally left blank.

Kurzfassung

AVL List GmbH, ein österreichisches Unternehmen in der Automobilbranche, integrierte eine neue Simulationssoftware (AVL InMotion) für Hardware-in-the-Loop-Tests von Hybridfahrzeugen in eine bereits bestehende Software Suite. AVL InMotion soll das manöver- und eventbasierte Testen unterstützen, welches virtuelle Fahrmanöver an Prüfständen simuliert. Dadurch können bereits Ergebnisse sehr früh im Entwicklungsprozess durch die Analyse verschiedener Reaktionen von Verbrennungsmotoren sowie Hybrid-Systemen geliefert werden. Vor der Integration wurde die dreidimensionale Simulationssoftware durch eine externe Komponente repräsentiert. Diese Diplomarbeit konzentriert sich auf den Schwerpunkt von Einfluss und Wirkung der externen Applikation und auf welche Weise die Integration in die bereits existierende und recht komplexe Software Suite mit dem Fokus auf benutzerzentriertes Design umgesetzt wird. Eine solche Erweiterung könnte sich auf die Usability auswirken. Elemente der Software müssten neu gestaltet werden, um die vorausgegangene Funktionalität des Softwaresystems beizubehalten oder um gar eine gesamte Verbesserung der Software Suite zu erwirken. Diese Diplomarbeit beschäftigt sich mit der Frage, ob die Integration einer Softwarekomponente in ein größeres System die Benutzerfreundlichkeit für die Anwender verbessert, welche bereits mit dem bestehenden Software System vertraut sind.

Die im Rahmen der Diplomarbeit erfassten Ergebnisse und die daraus abgeleiteten Verbesserungsvorschläge sind für das Unternehmen AVL eine nachhaltige und nutzbringende Basis für die weiteren Entwicklungen von Software Produkten wie beispielsweise AVL InMotion. Diese Arbeit unterstreicht die Bedeutung von Benutzerfreundlichkeit in der Softwareentwicklung und AVL hat durch diese Untersuchung zusätzlich die Erkenntnis gewonnen, vermehrt in den Bereich Usability zu investieren.

Schlüsselwörter
Usability, Anforderungsmanagement, Prototypenbau, Testen der Benutzeroberfläche

ÖSTAT-Fachgebiete

1161	50 %	1157	30 %	1140	20 %

ACM Klassifikation
H.1, H.1.2, H.3.1, H.3.4, H.5.2

This page intentionally left blank.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, March 16th, 2011 . . . .

Signature

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, 16. März 2011 . . . .

Unterschrift

This page intentionally left blank.

Acknowledgements

First and most of all I would like to thank my family for supporting me through all these years of my studies. Without you none of this would have been possible.

I particularly would like to thank my supervisor Mr. Andreas Holzinger, who enabled me to write a master's thesis on a context of a human-computer interaction topic. Thank you for all the advice and assistance during the long process of writing.

Special thanks also to my secondary supervisor Mr. Stephan Lenhart, who was also my direct contact at AVL List GmbH. Thank you for having time and patience and giving me valuable feedback. Your positive attitude was always an additional motivation for me to finish this paper.

I would also like to thank Mr. Gerbert Orasche who initially offered me the opportunity to cooperate with AVL List GmbH and who provided me access to all necessary resources in order to write this master's thesis. Furthermore, I am grateful for the unique opportunity to get a deeper insight into the practical world of customer needs with the support of Mr. Michael Resl.

Thanks also to my friends and my AVL colleagues who motivated me from the very beginning. Thanks to my best friend Martin for accompanying and supporting me during my entire university career. And last but not least thank you Flo for being there for me at anytime.

Olivia Waclik
Graz, March 2011

This page intentionally left blank.

List of Abbreviations

Abbreviation	**Key**
3D	Three-dimensional
ATT	Attractiveness
AVL	Anstalt für Verbrennungskraftmaschinen List
CBSD	Component-Based Software Development
CoU	Context of Use
CSUQ	Computer System Usability Questionnaire
CW	Cognitive Walkthrough
GUI	Graphical User Interface
HCI	Human–Computer Interaction
HE	Heuristic Evaluation
HFRG	Human Factors Research Group
Hi-Fi	High-Fidelity
HIL	Hardware-in-the-loop
HQ-I	Hedonic Quality-Identity
HQ-S	Hedonic Quality-Stimulation

HTML	HyperText Markup Language
IBM	International Business Machines
INFOQUAL	Information Quality
INTERQUAL	Interface Quality
ISO	International Standard Organization
Lo-Fi	Low-Fidelity
MBS	Maneuver Based Sequencer
MSDN	Microsoft Developer Network
MUSiC	Metrics for Usability Standards in Computing
N/A	Not Applicable
OVERALL	Overall Satisfaction
PQ	Pragmatic Quality
PSSUQ	Post-Study System Usability Questionnaire
PUMA	Prüfstands- Und Messtechnik-Automatisierung
PUTQ	Purdue Usability Testing Questionnaire
RE	Requirements Engineering
SUMI	Software Usability Measurement Inventory
SysML	Systems Modeling Language
SYSUSE	System Usefulness
Tcl	Tool Command Language
THA	Thinking Aloud
UCA	Usability Context Analysis
UCD	User-Centered Design

UE	Usability Engineering
UI	User Interface
UML	Unified Modeling Language
UP	Usability Problem
UX	User Experience

This page intentionally left blank.

Contents

1. Introduction and Motivation for Research

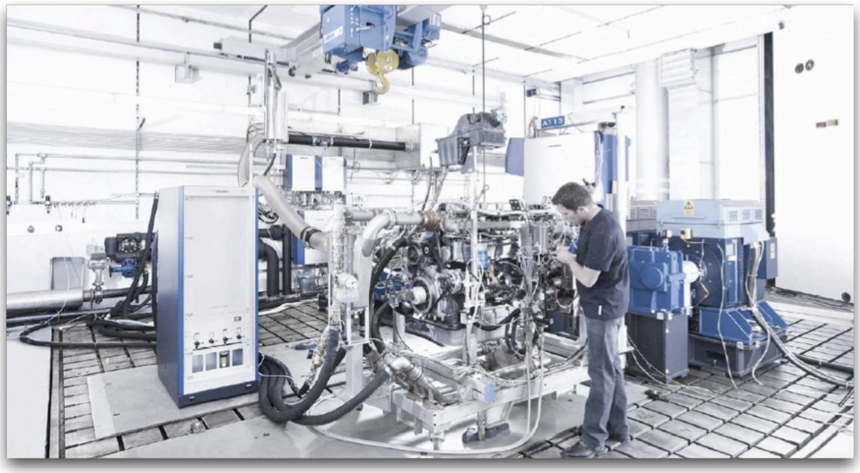


Figure 1: Testbed Operator
[Source: AVL List GmbH]

One of the biggest issues in testing vehicles for their roadworthiness is the traceability of the behavior of real drivers. In many cases, it is impossible to reproduce every detail that the test drivers have noticed or performed while they are driving along the road. For example, the action of changing gear carried out by a particular test driver is often hard to identify by engineers.

AVL List GmbH, an Austrian based enterprise in the automotive industry, used a sophisticated three-dimensional (3D) software solution in order to simulate the special characteristics of a real driver on predefined roads. This uses a virtual vehicle and aims at getting control of all relevant measured data of a vehicle in order to improve test drives.

At the beginning, this 3D simulation application was represented by an external component and AVL integrated it into their enterprise software. It is important to mention that this third-party software application could not be directly integrated into the existing business software suite (AVL PUMA Open). The adaptation with the corresponding modifications had to be considered before integration. The newly developed subproduct is called AVL InMotion and has been integrated into the test bed software of AVL.

Figure 2 indicates the three main steps as described in the previous paragraph.

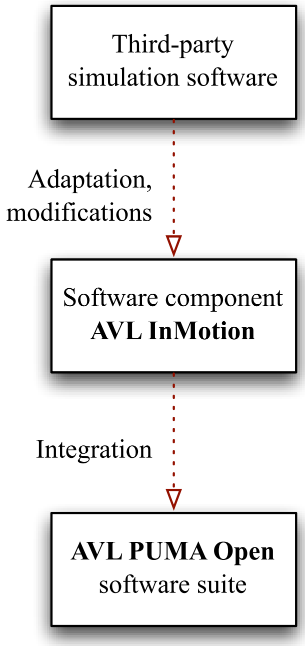


Figure 2: Key Steps of Solution Process

The enrichment of the existing software suite with the third-party software component is intended to attract more of the target group of AVL customers who require the feature of the track editor, for example, but do not need the entire functionality of the third-party product.

This master's thesis deals with the effects of integrating an externally developed standalone application into an existing business software suite, proposes design solutions and conducts an empirical research study to test a specified hypothesis relating to the usability. A comparable usability test has been conducted in order to get feedback and to evaluate the extended software suite concerning both its usage for users familiar with the existing system and its quality aspect. Finally, possible redesign solutions are provided so that the user interface (UI) may be improved by including a user perspective. Figure 3 shows an overview of the identified phases of the specified thesis topic as described previously. The presented stages highlighted in gray shadows have been processed by AVL itself. The other stages will be addressed in this master's thesis.

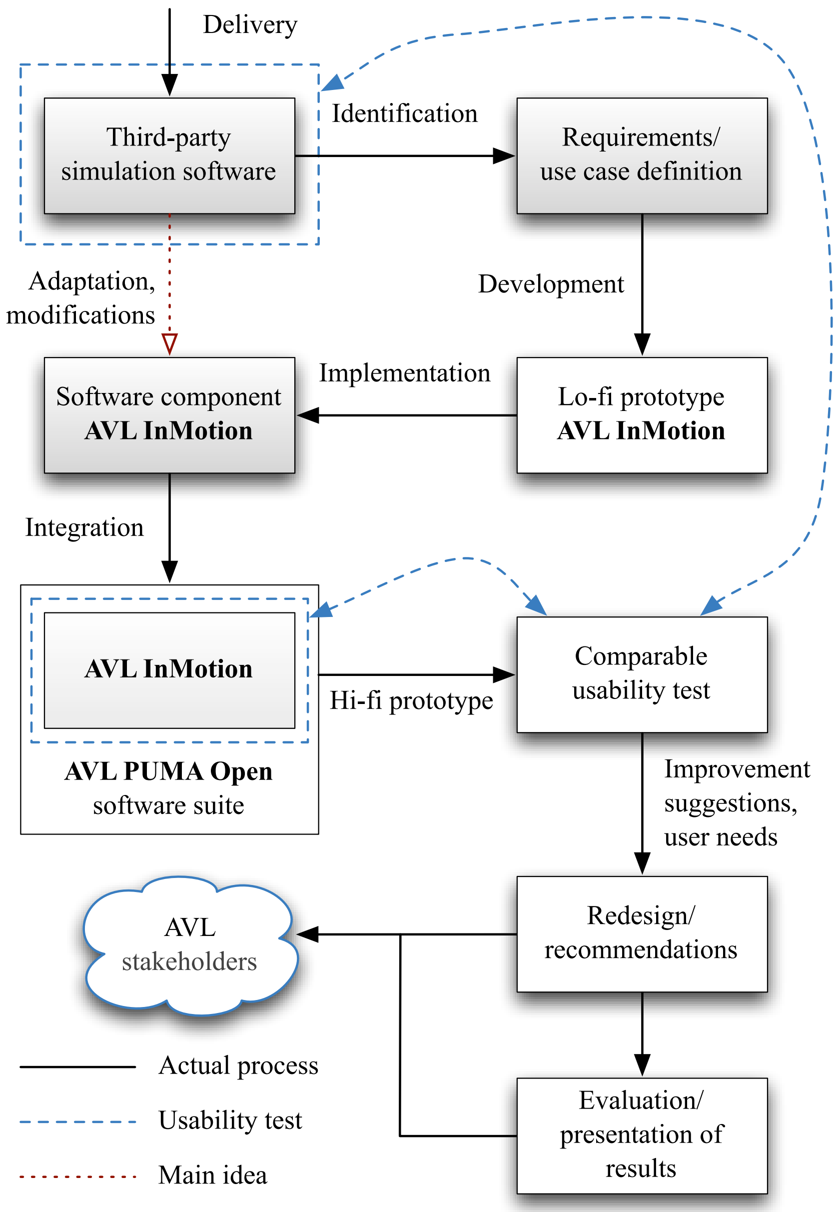


Figure 3: Overview of the Identified Phases

The writer's personal motivation for this diploma thesis was associated with her interest in UI design. Since 2003 she has been working as a freelancer at AVL List GmbH and was responsible for the development of automated tests for the business software suite AVL PUMA Open. During the long period of testing, several design issues were noticed. This was definitely one reason for the focus on writing a master's thesis within the context of a human-computer interaction (HCI) topic. However, due to the location of the department of usability engineering it was possible to combine usability at AVL with this master's thesis relating to user-centered design (UCD).

Theoretical background to this work, suited as an introduction to the topic, is provided in Chapter 2. The following section (Chapter 3) provides examples of related work that has been done in a similar context. Besides the descriptions of AVL InMotion and the test method, which is used for the comparable usability test, Chapter 4 focuses on the development of the prototype for the user interface of AVL InMotion. Chapter 5 provides an overview of the issues, recommendations and results of the usability study conducted in this master's thesis. Furthermore, a proposal for a redesigned UI is presented based on the findings of the usability test. Finally, Chapter 6 describes the lessons learned relating to the performance of a usability test within a large company. Chapter 7 concludes the subjects of this master's thesis with regard to the design and testing of a prototype for the UI. Additionally, an outlook is given into possible future work and is presented in Chapter 8.

2. Theoretical Background

2.1. Usability

"Bad Usability Equals No Customers" (Nielson, 1999, p. 14).

In literature, usability is described as a key concept of designing the human-computer interaction that covers the issue of user-friendly software design (Niegemann, 2008). Being more precise, several denotations can be added to usability such as user friendliness, easy handling, ease of operation, utility and usefulness. Usability is synonymous with the definition used in ISO 9241-11[1], which defines this term as *"the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specific context of use"* (ISO, 1998). Every system or product should be manageable, reusable and contain all these functions that are required for users to perform their work (Gould & Lewis, 1985).

Nielsen (1993, p. 26) states that *"usability has multiple components and is traditionally associated with these five usability attributes",* which subsequently will be specified:

Learnability A system should be developed in a way that all users are able to do their work productively after a short training period. The focus, therefore, lies on the intuitive operating interface and on uniformed design and navigation elements.

Efficiency This criterion denotes how efficient professional users can be if they have learned the operating sequence with the system before. Efficiency is occasionally a parameter for the amount of time it takes to complete a certain task.

Memorability In the case of users not having utilized a system or product for a long time, they should be able to reuse the system within a short period of time.

Error tolerability This principle prevents users from generating errors by using the system or providing them with sufficient information so that they can recognize an error

[1]This 9241-11 standard that is part of the ISO 9241 series provides the definition of usability.

or rather make corrections upon their failures quickly (Niegemann, 2008).
Optimally, of course, no error should occur. However, the user needs to be supported by the software to solve a problem if an error does occur. Helpful error messages contain a clear problem description, instructions for a solution and should be in plain language (vocabulary of user and no codes). *"An error dialog should always be polite, illuminating, and helpful"* (Cooper et al., 2007, p. 537). See Figure 4 in contrast to Figure 5, which have been taken from MSDN Library[2].

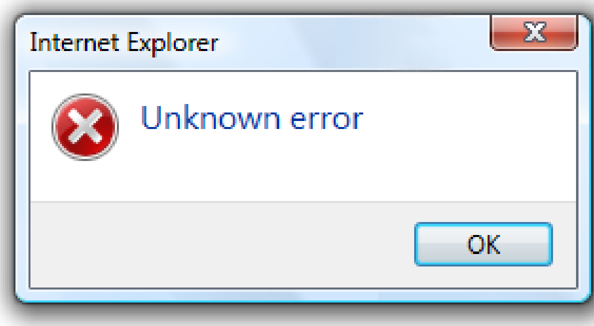


Figure 4: Example of a Useless Error Message

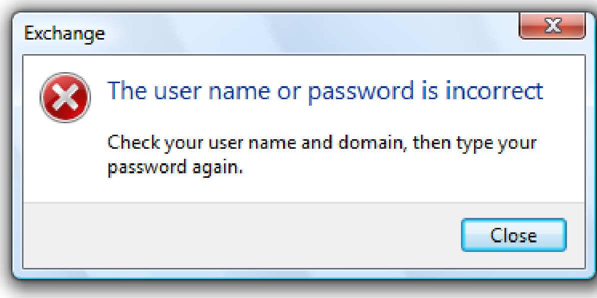


Figure 5: Example of a Helpful Error Message

Satisfaction The users trigger an inner attitude by using the application and decide subjectively if they would like to use this product or system again to do their work. Nielsen (1993) mentions that the visual design could influence the user's satisfaction, which is reflected in the acceptance of the user. It gives feedback on the suitability of the functional range and whether all objectives have been achieved.

[2]Microsoft Developer Network, `http://msdn.microsoft.com/`

2.1.1. Transparency of the Technology

"Users are not designers" and *"designers are not users"* (Nielsen, 1993, pp. 12, 13). Whenever problems with human-computer interaction arise, the majority of users troubleshoot their own actions (Niegemann, 2008). Furthermore, Wilding (1998) mentions, that communication failures result in a system whose usage is quite complex and difficult. Strategic usability quite often fails because usability experts are not able to persuade the management of the importance of integrating usability in early cycles of the software engineering process (Rosenbaum et al., 1999). In addition, *"issues such as transparency, learnability, and the support offered to users through guides, manuals, and clear and informative device feedback"* (McNamara & Kirakowski, 2006, p. 28) are being mentioned more and more often in user comments. Several usability evaluation methods (see Chapter 2.2.3) support designers in identifying the problems of the users by testing different interactions with the design and detecting the parts, which could be improved accordingly (Hornbæk & Frokjær, 2005).

2.1.2. Measurement of Usability

Usability can be measured in different ways; the most commonly used methods are questionnaires (see Paragraphs 2.2.1.3 and 2.2.3.1) and observations (see Paragraph 2.2.3.1). The following approaches provide a selection of the procedures, which are examined, discussed and applied most often in both literature and practice.

2.1.2.1. Quality of Use

According to Burmester et al. (2002), usability can be considered as the quality of use - a result of the combination of a software product and the context of its use (see Chapter 2.2.1.1). Another reference to this concern is that software is qualified *"for its intended purpose in the real world"* (Bevan, 1995b, p. 350). Macleod (1994) highlights that usability is much more than simply providing a graphical user interface (GUI) or preparing a variety of widgets for an application. Considering user-friendly design in the context of the quality of use, particular evaluations provide meaningful and valuable knowledge for design and redesign as well as technical innovation and improvement for the end users in their working environment. Several methods (see Chapter 2.1.2.2) for measuring the main properties *effectiveness, efficiency* and *satisfaction* of usability have been developed by performing usability tests supported by a prototype or an already existing application. In this respect, a real usage situation will be approximated. Test users have to go through a task list that is representative of an actual context of use which ideally also covers the complete functionality of the system (Johnson et al., 1989). The inspection of the interaction between user and system provides an informative basis of quality of use (see Figure 6). At the same time it delivers results for measuring usability (Bevan, 1995a).

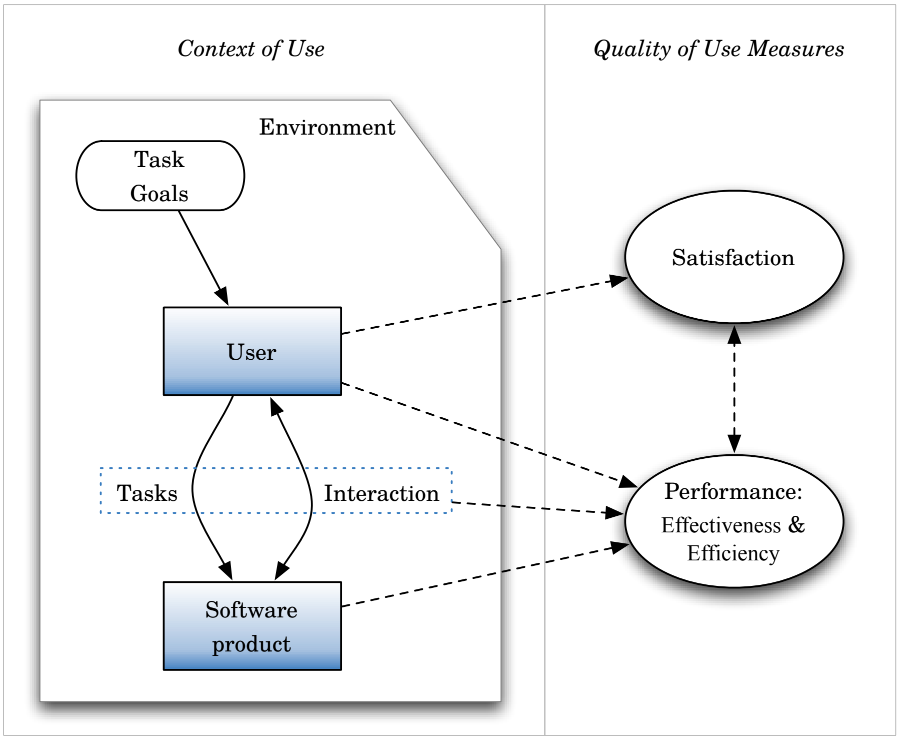


Figure 6: Quality of Use to Measure Usability
[adapted from (Bevan, 1995a)]

2.1.2.2. Measuring Effectiveness, Efficiency and Satisfaction

In the following paragraphs, an introduction of several techniques developed by the European MUSiC[3] project are listed in order to give an overview of methods that support the measurement of usability (Bevan, 1995a).

Performance Measurement Method This evaluation technique measures the user performance by using the system which is to be tested, to achieve defined task goals. Consequently, the results make a statement about the effectiveness and efficiency of the system usage. Due to the fact that time is also taken at this evaluation, both information about

[3]Metrics for Usability Standards in Computing

eventual unproductiveness and the location of such troubles are recorded by completing the specified tasks.

Measurement of Satisfaction Acceptance represents satisfaction for a user (see also Chapter 2.1). A measurement of this particular attribute is possible by making use of the SUMI[4] questionnaire, which was developed within the MUSiC project by the Human Factors Research Group (HFRG). It identifies the satisfaction of the users of a software product. This questionnaire contains 50 questions. All items include rating statements with the options *'agree'*, *'undecided'* or *'disagree'*. Furthermore, the questions are designed in such a way that end users are able to respond intuitively and quickly. The SUMI questionnaire refers to five dimensions of user satisfaction: efficiency, learnability, control, helpfulness and affect (Macleod, 1994). The meaning of the last term describes the user's emotional response to the system. For example, a representative statement for exploring the attribute learnability in this list of questions is demonstrated by the following phrase: The user will never learn to use all features that are offered in this software (Stone et al., 2005).

2.1.2.3. Measuring the Appeal

It is a matter of common knowledge that usability is generally accepted as a fundamental quality characteristic as already described in the above section. Moreover, Burmester et al. (2002) state two different quality aspects to evaluate the system's appeal: the pragmatic quality that *"refers to task/goal fulfilment, i.e., the usability and utility of a product"* and the hedonistic quality, which *"addresses quality attributes with no obvious relation to task/goal-fulfilment, such as original, innovative, exciting, or exclusive"* (Burmester et al., 2002, p. 32).

The aim of design is to produce software products whose qualities, both pragmatic and hedonistic, are highly rated. In order to measure the attractiveness of a software product, a questionnaire named AttrakDiff™ was developed (Hassenzahl et al., 2003), which enquires and further analyzes the two qualities. AttrakDiff™ can be described as an instrument that enables the measurement of the appeal of a software product in a standardized way, which is also cost and time effective (Hassenzahl et al., 2003).

This questionnaire is offered online[5], free of charge and can be used in an anonymous mode by users or customers amongst others. Furthermore, three types of using AttrakDiff™ are available at the present time, namely *Single Evaluation*, *Comparison*

[4] Software Usability Measurement Inventory, `http://sumi.ucc.ie` (retrieved on April 11th, 2010)
[5] `http://www.attrakdiff.de/` (retrieved on April 14th, 2010)

before - after as well as a *Comparison of Product A and Product B*. The mode of operation of AttrakDiff™ is based on using the *semantic differential*, a type of rating scale created to measure the surplus meaning of concepts or objects. Each question consists of an adjective word pair and the adjectives in each pair are extreme opposites. Finally, it should be pointed out that there are seven gradations between the extremes to enable the possibility to declare the intensity of the 'opinion' (see Figure 7).

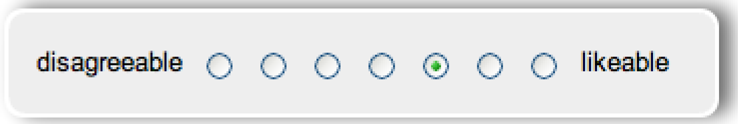


Figure 7: An Example Statement of an AttrakDiff™ Questionnaire
[from AttrakDiff[5]]

Especially notable is the fact that the perception of product qualities is subjective. Therefore, it is particularly beneficial that the potential users do this special evaluation. Commonly used techniques and methods in usability engineering emphasize the pragmatic quality only. When carrying out the AttrakDiff™ test, four dimensions are evaluated. Besides the pragmatic quality and attractiveness, a special focus is put on the hedonic quality because it is divided into the aspects *hedonic quality - stimulation* and *hedonic quality - identity* (see Figure 8).

In their paper, Hassenzahl et al. (2003) also note that a system has hedonic quality if the implementation extends the possibilities of the users through new functionalists, presents new challenges, stimulates through specific visual styling or communicates a desired identity (by being professional, cool, modern or different, for example).

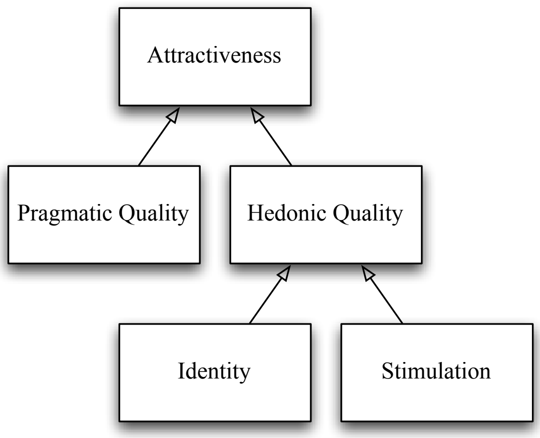


Figure 8: Concept and Interaction of the AttrakDiff™ Dimensions
[adapted from (Schrepp et al., 2006)]

2.2. User Centered Design Process

"Designing for usability involves establishing user requirements for a new system or product, developing design solutions, prototyping the system and the user interface, and testing it with representative users" Maguire (2001, p. 453).

Nowadays, users decide on usability, consistency or even longevity of a software system and they are in a position to determine whether a software solution has been developed in an optimal way, both from a technical and a customary point of view. They can also tell it is also able to cope with a task as planned. The reasons for many information technology projects having failed are poor Requirements Engineering (RE) (see Paragraph 2.2.1.2) as well as the involvement of end-users, which come to pass incompletely and late in the majority of cases (Petrovic & Weissenberger, 2007).

Based on business competition between different software producers and developers, not every single producer can afford to publish a system, which is bad quality to operate. The needs of the end-users have to be integrated in order to offer a user-oriented creation. The improvement of product quality must be achieved. Consequently, this finds the approval of customers that can help the companies increase their competitive edge. Moreover,

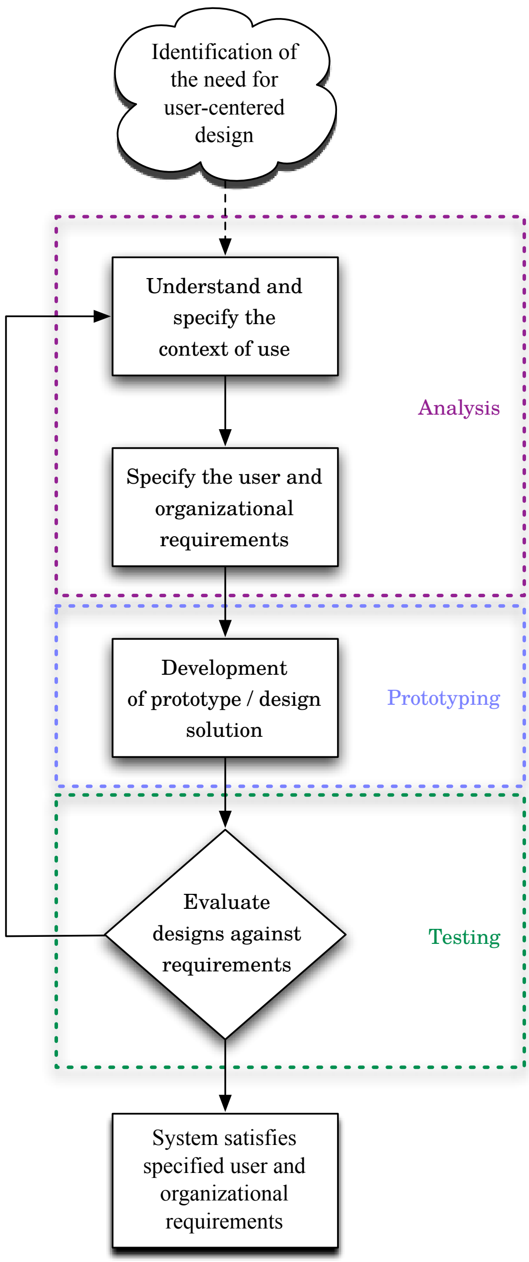


Figure 9: Iterative Approach of the User-Centered Design Process
[adapted from (Jokela et al., 2003)]

Dahm (2006) specifies key benefits of the participation of users. A considerable advantage is to reduce expenses of training courses and other additional costs because systems, as well as products, are subsequently easy for the operators to use. This group of benefits also includes the avoidance of stress, the improvement of customer satisfaction as well as an increase of productivity. Keinonen (2008) advises the introduction of ongoing integration of potential users into the process of system development at the very beginning to perform an iterative design process. This ensures system satisfaction concerning specified users and organizational requirements (see Figure 9). This approach is also reflected in three key principles, which are recommended by Gould & Lewis (1985): The first one is *"Early Focus on Users and Tasks"*, which suggests that designers have to get an idea regarding the intended users and their tasks by conducting interviews and discussions as well as observations. The second key principle is the *"Empirical Measurements"* that proposes making use of prototypes and simulations to perform real work tasks with potential users in order to get their behavior and reactions for further analysis. Finally, the third principle is *"Iterative Design"*, which stands for a *"cycle of design, test and measure, and redesign, repeated ad often as necessary"* (Gould & Lewis, 1985, p. 301).

In the following chapters, the three main stages of the user-centered design process, *Analysis*, *Prototyping* and *Testing* (as shown in Figure 9), are described more in detail. They show the principles explained above for the purpose of ensuring user-friendly design in software development.

2.2.1. Analysis

The main goal of the *Analysis* step in the context of the user-centered design process is to specify the context of use, user requirements and the identification of the user who utilizes the software system.

2.2.1.1. Context of Use

Whenever a new product or system is developed, an existing one is replaced or possibly, an entirely new one or simply a newer version is created. Regardless of these cases, the Context of Use (CoU) is different from whatever existed before. Perhaps other users will emerge or new tasks need to be identified, all of which have not been relevant in the late environment (Jokela, 2002).

In this respect, one of the first activities of *Analysis* within the human-centered design process is the understanding and specification of the Context of Use (van Wyk & de Villiers, 2008) (see Figure 10).

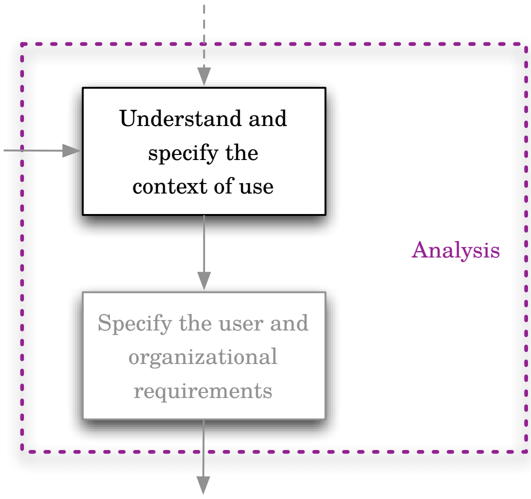


Figure 10: First Activity: Context of Use
[adapted from (Jokela et al., 2003)]

The CoU broadly determines the information and factors that regulate the usability of a product or system. *"In fact, it is incorrect to describe a product as ergonomic or usable, without also describing the context in which the product will be used - in other words, whom the product was designed for, what it will be used for and where it will be used"* Maguire (2001, p. 453). The Usability Context Analysis (UCA) is a supporting technique for the identification and gathering of the contextual aspects of usability, and facilitates the specification of users' characteristics, their tasks, the objectives of their work and the situation of use (Macleod, 1994; Maguire, 2001; van Wyk & de Villiers, 2008). In order to gain a better understanding, the progress of the methodology described above is clearly represented in Figure 11. In addition, Maguire (2001) exemplifies the CoU analysis (applied to an automated banking machine) for illustration purposes in his paper. Maguire emphasizes that the *"context analysis is an essential pre-requisite for any work on usability"* (Maguire, 2001, p. 481). With regard to Contextual Design, Beyer & Holtzblatt (1999) state that *"great product ideas come from the marriage of a designer's detailed understanding of a customer's need and his or her in-depth understanding of the possibilities introduced by technology"* (Beyer & Holtzblatt, 1999, p. 32).

Requirements for usability can be identified through CoU analyses too, which also provides a basis for testing usability. *"An understanding of the Context of Use forms a useful input to the process of specifying usability requirements, constructing a design prototype which can be evaluated and evaluating the prototype with typical end-users"* (Maguire, 2001, p. 481). In this respect, it is necessary to get findings from prospective users relating to users' effectiveness, efficiency and satisfaction (see Chapter 2.1.2) (Duechting et al., 2007; van Wyk & de Villiers, 2008).

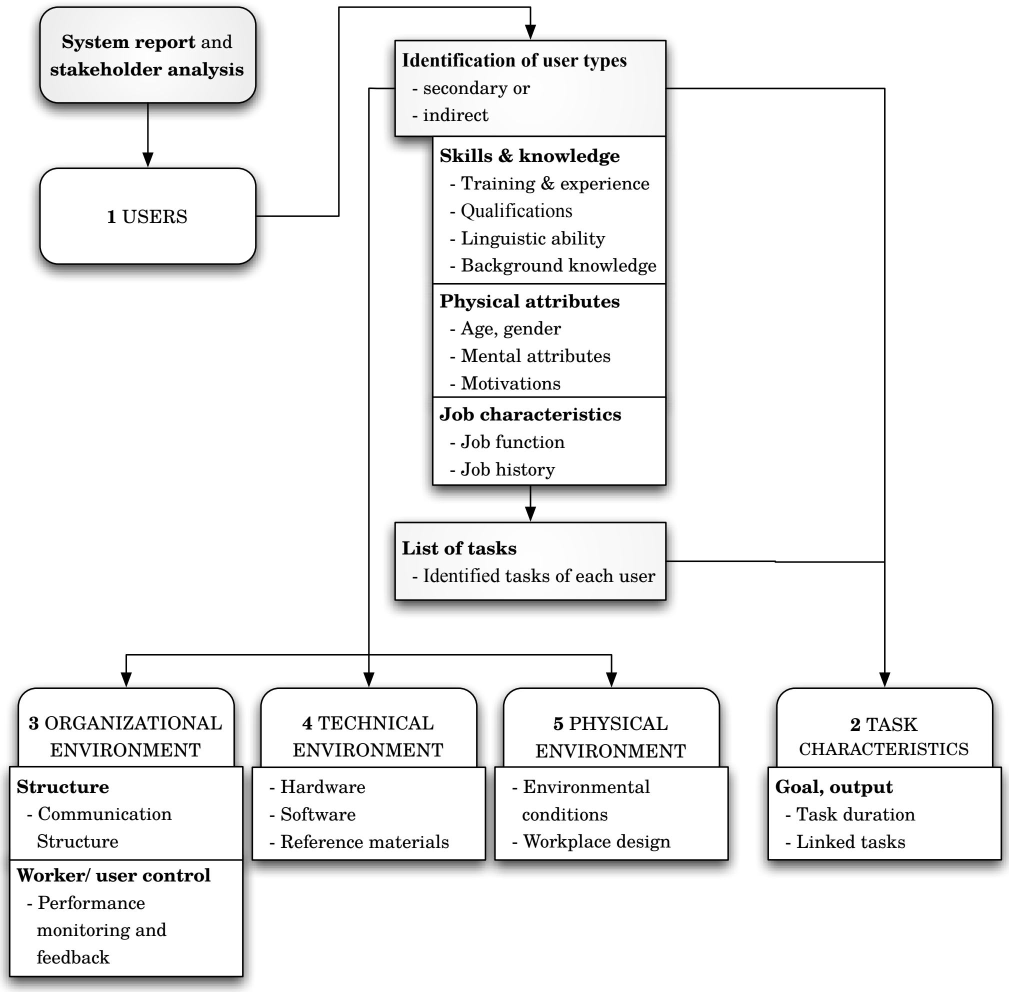


Figure 11: Context of Use Analysis Progress
[extracted from (Maguire, 2001)]

2.2.1.2. Specification of Requirements and User

The completion of the activity *analysis* of the user-centered design process further implies the specification of the user and organizational requirements, which is shown separately in Figure 12. The identification of both the requirements and the intended users is an essential task preparation for the activity *Prototyping* (see Chapter 2.2.2). *"The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended"* (Nuseibeh & Easterbrook, 2000, p. 37).

For this paper, the definition of *requirement* will be used as a documented need of what a system should be or do in software engineering. It is a statement that expresses fundamental attributes and details in order to gain value and utility for a user. Requirements of a software system need to include clear definitions of the current software performance as well as the support program for the software (Yu, 1997). Usually, developers are not able to deliver accurate results without clear and well-written specifications in the very beginning of development, clearly defining the expectations of the end users. Therefore, it is also in the interest of the customer to give the software developer a detailed brief to ensure the production of the exact software that is needed (Hsia et al., 1993).

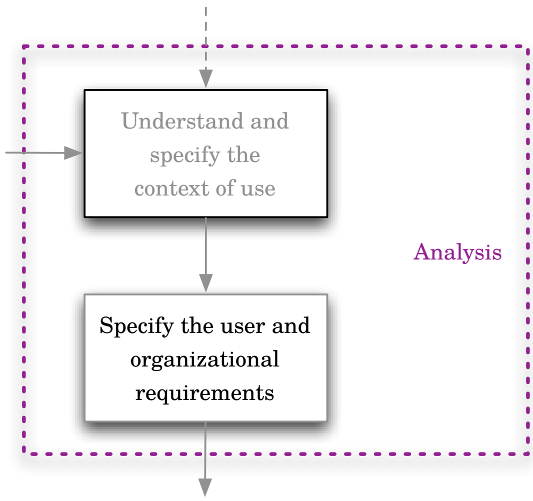


Figure 12: Second Activity: Requirements and User
[adapted from (Jokela et al., 2003)]

Needless to say, nowadays there are lots of different techniques (such as interviews, participant observation, task analysis, prototyping and so on) which have been developed in the field of HCI. They support the eliciting, specification and validation of usability requirements comprising the characteristics of users, their tasks or their working environment. In addition, it is possible to identify the usability goals that are pointed out in the ISO 9241-11 standard (Seffah et al., 2001).

Software System Requirements Engineering

„Requirements Engineering is the branch of software engineering concerned with the real-world goals for functions of and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families“ (Zave, 1997, p. 315). Nuseibeh & Easterbrook (2000) extend the above definition and state that *"real world goals"* may be contributed to gain motivation for developing a software system. Thus, they must be linked to the conceptions of a user or organization.

User satisfaction is one of the main indicators in generating success, even in software development, and offers an additional measured factor of system effectiveness (Procaccino & Verner, 2009). The success of a software system often depends on how well the application suits the needs and expectations of the end-users and their working and/or living environment. Software requirements contain the needs of the end-users, whereas RE defines the process of identifying the requirements of the ideal software system. Therefore, RE should describe the problem that needs to be solved precisely by using the newly developed software system (Cheng & Atlee, 2007). Figure 13 illustrates the previous statement – starting *"from the recognition of a problem"* that needs *"to be solved"* resulting into "*a detailed specification of that problem"* (Nuseibeh & Easterbrook, 2000, p. 38).

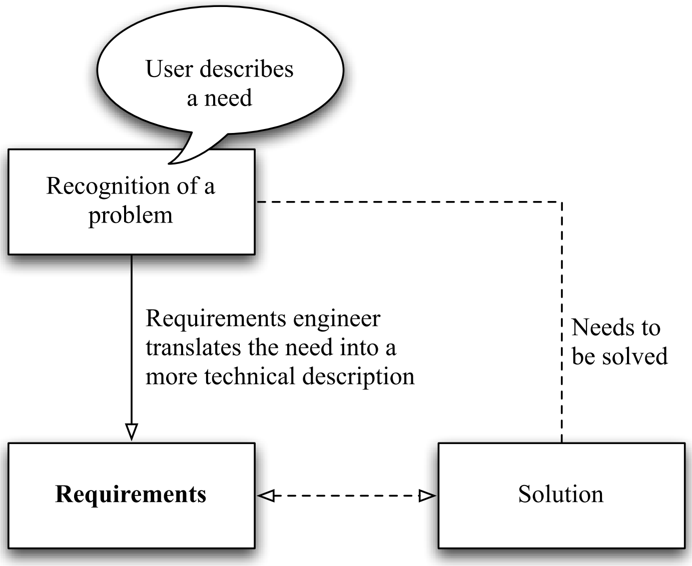


Figure 13: Development of Requirements

RE is the disciplined application to prove principles, methods, tools and notations to describe a proposed system's intended behavior and its associated constraints (Hsia et al., 1993). This process supports the identification and definition of the purpose of the software system and also *"characterizes the work that needs to be done" (Zave, 1997, p. 316)* in order to meet these requirements. Potential software problems have to be identified and user-friendly solutions need to be developed prior to the release of a software system. In addition, the RE process concentrates on the clear definition of requirements and their analysis, specification and validation (Anwer & Ikram, 2006).

RE also has an impact on the quality of the software. Errors that relate to requirements are amongst the most frequent, persistent, expensive and dangerous types of software failures and, as a result, this often implies incompleteness, inconsistency or ambiguity in a software system. The most common reasons for project delivery delays, costing more than budgeted, missing the deadlines and/or expectations are caused by requirement-related errors (van Lamsweerde, 2008). On the other hand, there are systems that are technically correct, but have lost their real needs. Therefore, Grudin (1988) refers to an example in his work. It's an important matter for the users and, accordingly, the customers to get support by determining requirements, which the system should meet.

Nevertheless, for the different RE activities, some preparatory work needs to be done before a project can actually start. For example, the way in which the various methods can be adopted or rather integrated should be determined (Nuseibeh & Easterbrook, 2000).

Supporting Goal oriented RE *"Goals are statements of intent and desired outcomes of the system under consideration"* (Anwer & Ikram, 2006, p. 121). *"Goal orientation is an increasingly recognized paradigm for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting and modifying software requirements"* (Alrajeh et al., 2006, p. 29).

In literature, there are several reasons why the listing of goals is important and should be formulated in quite an explicit way: *"Goals drive the elaboration of requirements to support them; they provide a criterion for requirements completeness and pertinence"* (Letier & van Lamsweerde, 2002, p. 119). Furthermore, the term 'goal' is defined as an interaction of multiple agents, who can be human, devices or software (Letier & van Lamsweerde, 2002), which are accountable for achieving the defined objectives. In order to identify goals, it is essential to deal with the question words *"why"*, *"how"* and *"how else"* (Anwer & Ikram, 2006).

Besides the functionality of the software system, other quality aspects also need to be met, such as performance, correctness, usability and customizability. Simultaneously, non-relevant requirements need to be identified in order to guarantee time and cost efficient software development. To understand the relevancy of individual requirements, it is helpful to separate constant data from changeable information.

The identification of goals is not always a simple task. Usually, the main objectives for the software system are either clearly defined by stakeholders or the information is made accessible to the requirements engineers in some other way. In the case of obtaining implicit information from stakeholders, the engineer has to undertake a goal elicitation to make them explicit. An evaluation of an existing system provides a set of goals, but most of the time this method leads to a list of problems and deficits (van Lamsweerde, 2001).

Supporting Scenario based RE Specification techniques of scenario based requirements support the RE and they are used to make the requirements more transparent. The usage of scenarios, which represent all possible paths of a single use case, identifies the requirements of a system. In addition, this procedure implies an amount of work to capture or rather to document scenarios (Sutcliffe et al., 1998).

This approach comes with some shortcomings, as individual elements of the requirements could be found repetitively. Therefore, (Ozkaya, 2006) refers to the use case analysis, which can be applied to structure the requirements into scenarios with the support of an object-oriented design. The focus here is increasingly on the Unified Modeling Language (UML), which is considered as an enlargement of the use case analysis when describing the needs of a system. The concept of this graphical modelling language is explained in Subchapter 2.2.1.2.

More information on this issue concerning of the relationship of scenarios and requirements can be found in Sutcliffe's reference literature (1998).

Eliciting Requirements The characterization of the requirements is the very first step in the RE process (Nuseibeh & Easterbrook, 2000) and is receiving more and more attention (Yu, 1997). All the information must be collected prior to the software development and listed in a well-structured form to ensure efficient and smooth programming (Nuseibeh & Easterbrook, 2000).

Zave (1997) introduced the following link between problems and tasks: *„A task can always be described as a problem ('How can this task be accomplished satisfactorily?') and a problem can always be described as a task ('Find a solution to this problem.')"* (Zave, 1997, p. 316).

Nuseibeh & Easterbrook (2000) list the following eliciting techniques in order to clarify possible requirements:

- *Traditional* techniques, such as interviews (see Paragraph 2.2.1.3), questionnaires (see Paragraphs 2.2.1.3 and 2.2.3.1), surveys or the evaluations of existing documentation
- *Group* techniques, such as focus groups and brainstorming
- *Model-driven* techniques, such as goal-based as well as scenario based methods (see Paragraphs 2.2.1.2 and 2.2.1.2)
- *Prototyping* (see Chapter 2.2.2)
- *Contextual* techniques, like field observations (see Paragraph 2.2.3.1)

Relationships of Requirements *"Capturing structural relationships between requirements is essential for effective visualization"* (Ozkaya, 2006, p. 3).

Since it is almost impossible to generate a complete list of requirements, another approach focuses on the relationships between the different requirements. Based on the questions below, a more complex list of requirements can be deduced:

- *"Which requirements can be derived from another?*
- *Which are directly affected by a change?*
- *Which are independent from the rest?*
- *How functional requirements and quality attributes are related to each other?*
- *What kinds of usage scenarios exist?"* (Ozkaya, 2006, p. 3).

Especially in the early phases of software creation, visualization techniques (for example UML) lead to good results, as these techniques make complex relationships of individual requirements more transparent and enable the developer to produce complex data structures.

The modelling language UML has been developed to specify, design and document the requirements of a software system. *"UML interactions, when used to model requirements, show the required behavior of several system components communicating towards a common goal"* (Araujo et al., 2004, p. 59).

Identification of Intended Users in the RE Process

Sharp et al. (1999, p. 387) refer to the following key stakeholder definition: *"In software engineering, stakeholders have been defined as: 'The people and organizations affected by the application'"*.

Accordingly, the above definition includes end-users, customers, sponsors, project managers and engineers who are responsible for the development of the system and its engineering design.

There is also another approach, which separates the stakeholders into two groups – *"into those who will use the system directly or indirectly, and those who will be involved in developing the system"* (Sharp et al., 1999, p. 388). In the RE process, the most important process is the identification of all relevant stakeholders as well as their needs for a specific system. Sharp et al. (1999) suggest using participatory techniques, such as contextual inquiry as used in the CoU analysis (see Chapter 2.2.1.1) to understand the working environment and the actual needs for a software program.

In general, techniques for stakeholder identification (see reference of Sharp et al., 1999) need to capture all involved parties. Analogical techniques (for example Personas, see Subchapter 2.2.1.3) support the target group in formulating and defining their requirements (Cheng & Atlee, 2007).

It is a matter of common sense that interactions take place between specific stakeholders, for instance the exchange of relevant data and information, directions or supporting activities (see Figure 14). These operations must be identified and captured, too (Sharp et al., 1999).

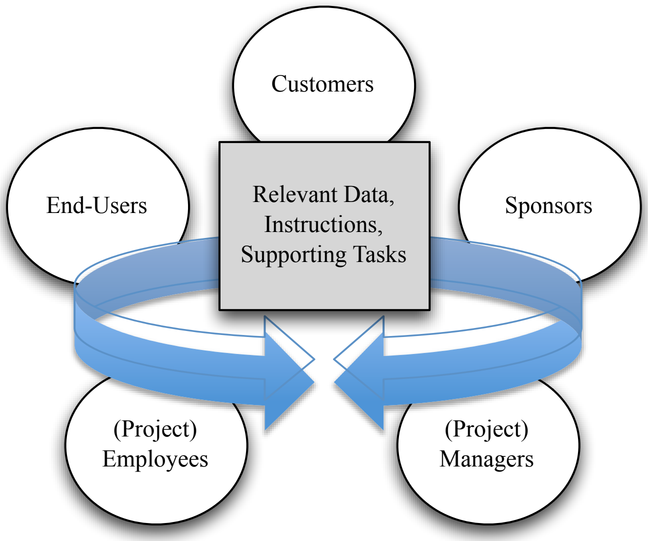


Figure 14: Information Exchange Between Different Stakeholders
[adapted from (Sharp et al., 1999)]

Since the needs and perceptions of individual stakeholders get more complex due to their interdependence of each other, the process described above might be quite difficult (Strohmaier et al., 2007).

Depending on the amount of stakeholders involved, difficulties may occur. Since it is quite difficult for customers or end-users to express their needs and expectations for the software, it is part of the requirement engineer's job description to understand the context in which the software will be used and come up with an ideal solution. In addition, it is significant to *"identify the most important goals of each participant, and ensure these goals are met"* (Nuseibeh & Easterbrook, 2000, p. 41). Therefore, it is necessary to use different requirements modeling techniques to handle the given and accessible information (Yu, 1997).

However, it is almost impossible to collect all the needs and perceptions of the identified stakeholders. There will always be some, which are unknown:

- resulting from suddenly occurring budget cuts for example or
- the development of a software system that is launched on the market through competitors and which is preferable to the customers.

Once the stakeholders' interests and needs have been identified, all collected information has to be prepared into a certain form to be used for structured analysis, communication and development (Nuseibeh & Easterbrook, 2000). For example, the software solution *HP Quality Center*[6] (see Figure 15) offers an opportunity to manage the requirements.

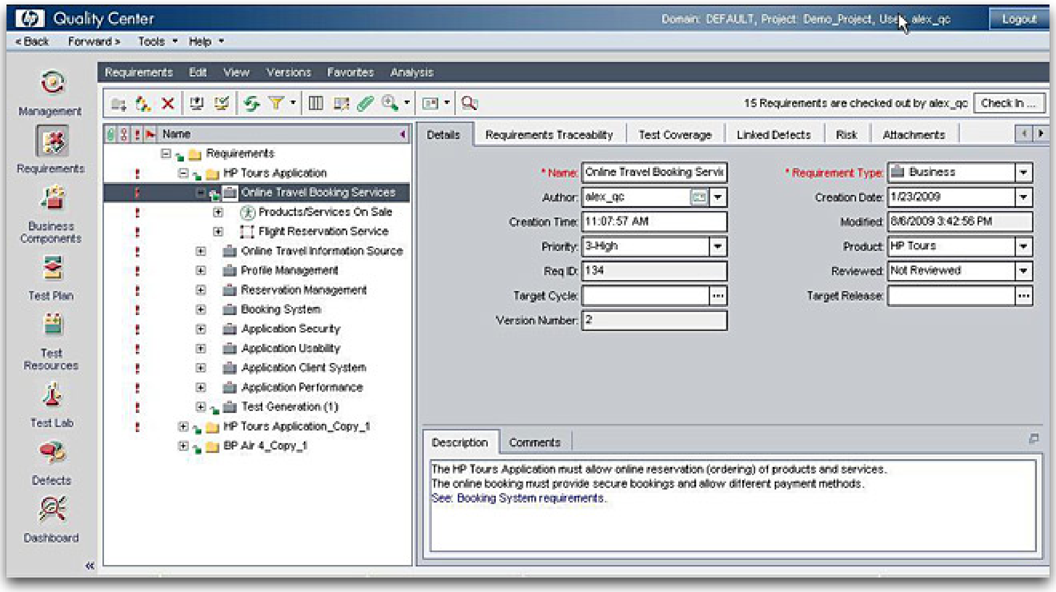


Figure 15: Requirements Management Module of HP Quality Center [from HP[6]]

End-user perspective A crucial point in the requirement analysis is the perspective of the end-users. Seffah et al. (2001) suggest that representative end-users and/or stakeholders should be invited to summarize the system from their perspective.

Table 1 contains a catalogue of questions that help the system engineer to lead the end-user in the right direction and get the answers needed for the software development.

[6] http://www.hp.com/

Question
What is the purpose of the system?
Why is this system necessary?
Who will use the system?
What will the users accomplish with the system?
Where will the system be used?
How will users learn to use the system?
How will the system be installed?
How will the system be maintained?

Table 1: Catalogue of Example Questions
[from (Seffah et al., 2001)]

Usability engineers analyze, or rather edit, this information and then create a complete system-summary. This summarization is the so-called *'roadmap'* for the developers and is the result of a common consensus between the end-users, other important stakeholders and the developers (Seffah et al., 2001).

2.2.1.3. Output of Analysis

<u>Usefulness and Results of Questionnaires</u>

"Questionnaire is a method for the elicitation, recording, and collecting of information [...]. It is a kind of an ordered survey, which provides feedback from the point of view of the system's user" (Drapala et al., 2010, p. 351). By filling out specially designed questionnaires, valuable data can be gained about the users themselves and their currently used software product and environment. Another advantage of using reliable questionnaires is the evaluation of the quality of software systems and their interfaces (Drapala et al., 2010). Various types of questionnaire are to be discussed in this work as follows:

Personal Questionnaires provide general information about the interviewees as well as their background knowledge of the software product and the computers the users are currently working with. Using this technique, the experience relating to the system usage (Drapala et al., 2010), including its support materials, can also be determined. The following is a brief example obtaining valuable information: A usability test is conducted on behalf of a company in order to verify the used software system. Depending on the analyzed answers, the evaluation can be that there is still a need for training of the employees or support materials.

Typically, this type of questionnaire is conducted at the beginning of a usability test. In Appendix A.2, there is an example of a prepared background questionnaire (customized for the AVL Software), which was used in the usability study conducted in this master's thesis.

Usability Questionnaires identify statistical information about the measurement of user satisfaction relating to system usability (Holzinger, 2005). Hornbæk (2006) notes the fact that there is a challenge *"to distinguish and empirically compare subjective and objective measures of usability"* (Hornbæk, 2006, p. 79). Chapter 3 presents related work of different usability questionnaires as well as providing a short explanation of each used evaluation form.

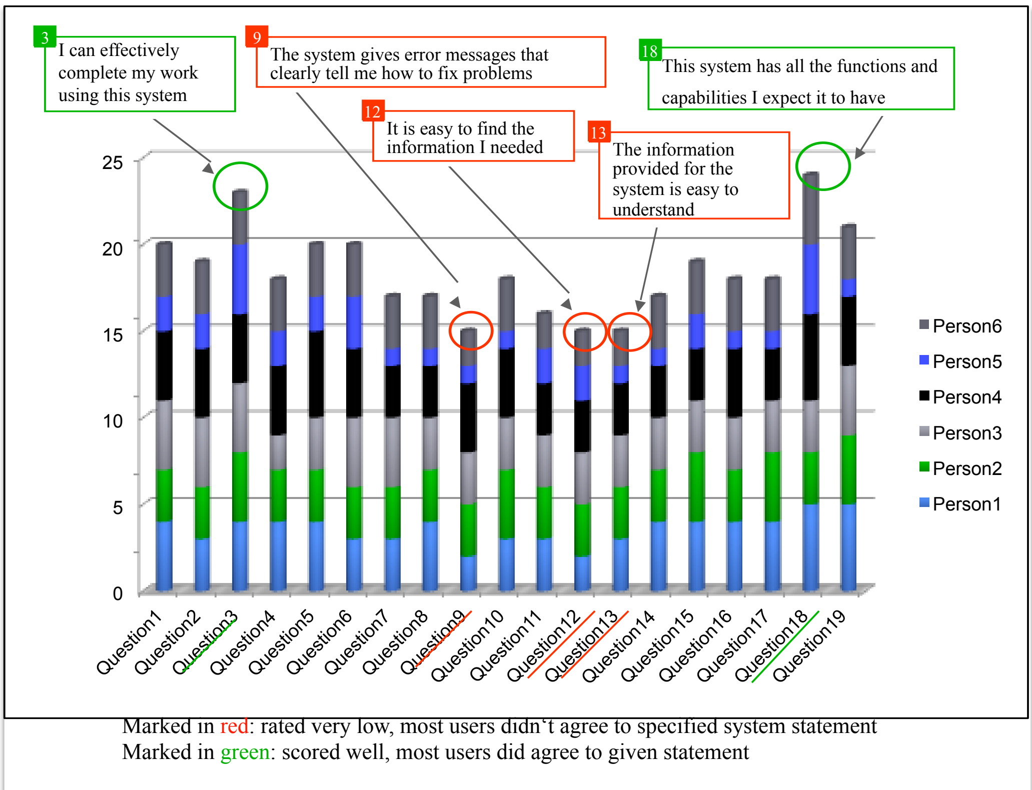


Figure 16: Example of Visual Presentation of Usability Questionnaire Results

Prepared in a clear and comprehensible way, statistical information is available for a presentation of the results (Holzinger, 2005). Figure 16 shows an example of the statistical interpretation of data taken from a questionnaire carried out in a usability study. This graphical representation indicates statements, which were rated very low by test participants, meaning that most users did not consent to this particular statement about the system. Other statements scored well, which roughly means that most users did agree to a given statement.

Of course, this result can be processed in a more detailed form relating to the individual rankings. Figure 17 displays an example of an individual score taken from the usability questionnaire of Figure 16: Opinions strongly vary concerning learnability and satisfaction. One user complains about the learnability of the system. Four users are satisfied with the overall system usability, one user voted neutral, one completely disagrees.

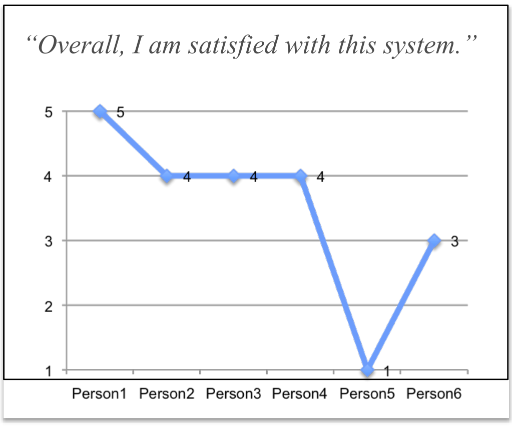


Figure 17: Example of Individual Ranking

Interview Another form of the questionnaire is the interview, which *"can be adjusted to respond to the user and encourage elaboration"* (Holzinger, 2005, p. 74).

Use Cases

The strategy of uses cases is probably *"one of the best known and most widely employed requirements elicitation techniques in the industry" (Lee et al., 1998, p. 1115)* and which is familiar in the object-oriented analysis. Use cases are used to describe the functionality of a system from the point of view of the users (Lee & Xue, 1999). In literature, this is often formulated as the representation of a conversion between system and the user (Ecklund et al., 1996). Based on a analysis of the context of use (see Chapter 2.2.1.1) and requirements eliciting (see Paragraph 2.2.1.2), valuable information about user tasks can be transformed into use cases by the requirements engineer, which *"can be used to describe the outwardly visible requirements of systems"* (Nuseibeh & Easterbrook, 2000, p. 39). The benefits of applying use cases are as follows: their description is not complicated and they are easily understandable. Moreover, use cases make the requirements for the creation of design and implementation more transparent (Lee et al., 1998).

Additionally, a graphical overview of the identified use cases is obtained with UML (see Paragraphs 2.2.1.2 and 2.2.1.2) . Lee & Xue (1999) support the issue of modeling use cases with the following statement: *"Use-case approaches are increasingly attracting attention in requirements engineering because the user-centered concept is valuable in eliciting, analyzing, and documenting requirements. One of the main goals of the requirements engineering process is to get agreement on the views of the involved users*" (Lee & Xue, 1999, p. 92).

Personas

The purpose of a *persona* is the presentation and creation of a user model with concrete characteristics and user behavior (see Figure 18, which shows an example of a persona).

A persona is similar to a user profile description and focuses primarily on the individual objectives (Blomkvist, 2002) of the user when using a system or product. Casas et al. (2008) refer to a positive statement of the inventor, such that real users are represented in the design process. Furthermore, this technique is based on the data gathering relating to the research of the intended users and the necessary information is provided through interviews, contextual analysis and other qualitative techniques (Junior & Filgueiras, 2005). However, there is a difference between a general description of a user profile and this specific model. Blomkvist (2002) describes personas as patterns, which are the representation of the behavior, goals and motives of the users. Junior & Filgueiras (2005) add that *"the persona composition can be based on imaginary information, demographic and biographical characteristics of the personality under modeling. Personas have names* *like real people and can be represented through [...] a picture, to add realism"* (Junior & Filgueiras, 2005, p. 278).

After the identification of a persona group, each can be categorized as a primary or secondary persona. The characteristic of a primary persona mirrors the main target group. In contrast, secondary personas can use the interfaces of the primary group with other additional requirements (Casas et al., 2008). *"In summary, personas are a valuable tool, particularly when used in scenarios where designers test and evaluate the system features for usability and effectiveness. Working with personas is one of the best ways to provide the developers with valuable insights and an efficient way of keeping the stakeholders in mind throughout the system design with the aim of making and simplifying design decisions"* (Casas et al., 2008, p. 116).

	Peter Thompson, 29	Professional (primary persona)
	Resident in:	Costa Mesa, California
	Profession:	Senior Engineer at Toyota Racing Development, California
	Family status:	Married to Nina (27 years old, teacher)
	Uses following AVL products:	PumaOpen 1.3.2, Puma 5.6, Concerto, Indicom
	Trainings/ Experiences:	Training courses for PO 1.3.2

GOALS

* Preparation of test runs and increase accuracy of tests
* Optimization of measurements
* Easy evaluation and comparison of measurement results
* Stability of the overall system to guarantee a trouble-free and efficient workload for operators and test cells
* Rather prefers a less graphical user interface than a graphical, complex and slow one
* Wants to find and understand errors to solve problems quickly

Figure 18: An Example of a Persona

<u>**Visualization of Task Workflow**</u>

By performing a CoU analysis (see Chapter 2.2.1.1), the current tasks of the intended users can be beneficially identified. This technique is based on interviews and observations in order to get information about the workflow of the users (Nielsen, 1992).

After evaluation, an activity diagram of the user tasks can be visualized *"with enough detail to clearly understand the different triggers, processes, and goals employed by different subtypes of users within the target user group"* (Kramer et al., 2000, p. 46).

Figure 19 shows an example of a flowchart of the determined tasks of two users. Such visualizations are highly recommended, because it is also possible to identify shared tasks or connecting points between the intended users. Not only do the tasks become transparent by performing a task analysis, but also problems can be pointed out, which are addressed by the users. Additionally, the labeling of problem positions can be graphically identified as shown in the accompanying figure.

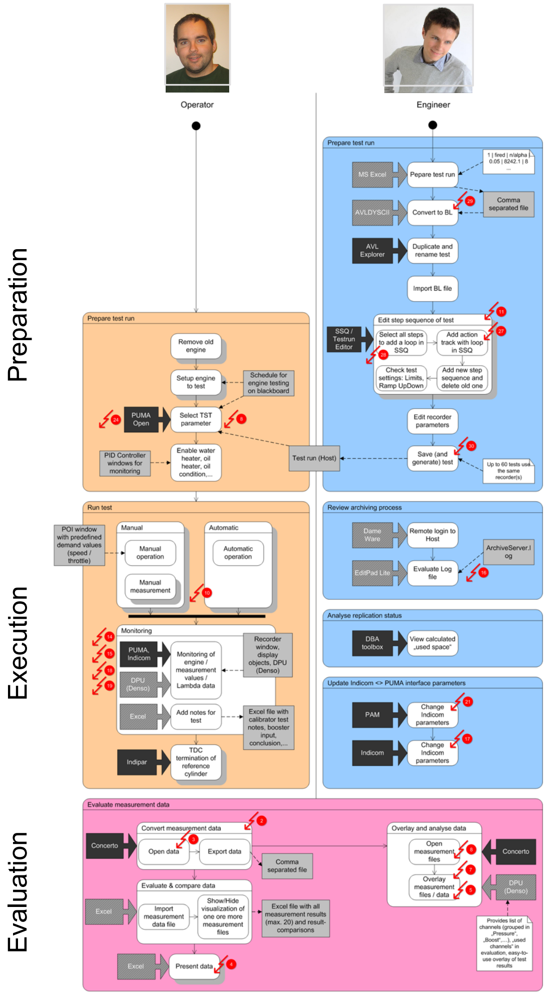


Figure 19: Flowchart of Tasks determined by CoU Analysis

2.2.2. Design and Prototyping

According to the *Analysis* stage, the creation of a design concept is an important matter, which reflects the requirements regarding the software application (see Figure 20). Therefore, the development and the specification of structure, content and design are necessary. Afterwards, the plan or idea of the design concept has to be implemented into a prototype, which shows the prospective layout and structure of the system and also supports developers trying out interfaces and dialogues of the specified software product.

Holzinger (2005) refers to the main principle *"that usability must be considered before prototyping takes place"* (Holzinger, 2005, p. 72).

Rapid prototyping is an effective method used to reduce risks and point out requirements that are relevant (Hsia et al., 1993). Prototypes, especially in the early phases of software development, provide the creation of an authentic UI for design and evaluation (Holzinger, 2004). Furthermore, the design engineer gets early feedback through the developed prototype. In general, some user experience (UX) teams have their own UX developers who are especially responsible for the development of prototypes and evaluation of the new designed GUI frameworks.

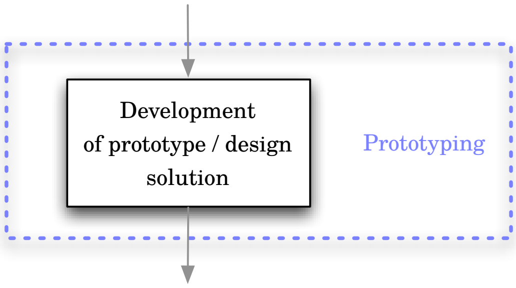


Figure 20: Third Activity: Prototyping [adapted from (Jokela et al., 2003)]

2.2.2.1. Classification of Prototypes

There are several techniques of prototyping, which allow stakeholders (refer to Subsection 2.2.1.2) to have an application visualized that has not yet been developed and helps them to get an idea of what the system will look like. In general, prototypes are differentiated according to their fidelity. The following describes the characteristics of low-fidelity (lo-fi) prototypes and high-fidelity (hi-fi) prototypes.

Figure 21: Example of a Paper Mockup [from (Brown & Holzinger, 2008)]

Lo-Fi Prototypes are used to validate first design proposals by users early in the software development. Such prototypes are created quickly and simply and are low in costs and easy to modify.

- Paper mockups[7] are drafted quickly and stakeholders get first impressions concerning the conceptual aspects from the design of the target system, which should be developed. (Casaday & Rainis, 1996; Holzinger, 2004; Brown & Holzinger, 2008).

[7] Mockup: Screen design and dialog elements, which are built for presentation purposes and testing.

As the name suggests, these prototypes are built on paper (see Figure 21), but they are also created using drawing software tools. Beyer & Holtzblatt (1999) conclude that *"Rough paper prototypes of the system design test the structure of a User Environment Design and initial user interface ideas before anything is committed to code"* (Beyer & Holtzblatt, 1999, p. 40).

- Another possibility is the usage of prototyping software in order to design a GUI with a slightly higher fidelity and which is closer to the final design. Such a tool is known as a GUI Builder or GUI Designer. The design tools *Qt Designer*[8], Apple's *Interface Builder*[9], *Adobe Photoshop*[10] or *Adobe Flash*[10] (compare with (Pleuss et al., 2007)) are examples of GUI mockup software. An alternative technique is the usage of, for example, HTML[11] for the creation of design concepts. This is also a quick method to prepare a presentation of the first scenarios of the planned software design (Holzinger, 2004). Typically, these prototypes are called "click dummies", which look like the target system, but do not provide any functionality. They show a conceptual user interface and provide some basic interaction possibilities for stakeholders.

Hi-Fi Prototypes already contain functionalities and are very close to the target system. Hi-fi prototypes provide stakeholders a more realistic impression of the software and in addition, these prototypes make it easier for them to make decisions on the software design (Rettig, 1994).

2.2.2.2. Requirement-Driven Design

Commonly, many design problems result from conflicting specifications or from requisites that are not easily identifiable to the designers such that they belong to the category of ill-defined problems. Ozkaya & Akin (2006) specify ill-defined problems in this manner as a situation in which all necessary steps to solve the problem as well as the problem itself are not sufficiently described. Furthermore, the authors explain that drawings, notes and diagrams, which are provided by the designers, support the provision of a solution related to the requirements specification. Design failures are often generated because designers produce quite an informative drawing of the design solution, but which does not actually met the given requirements. Therefore, it is important that design developers are able

[8] `http://qt.nokia.com/`
[9] `http://developer.apple.com/`
[10] `http://www.adobe.com/`
[11] **H**yper**T**ext **M**arkup **L**anguage

to detect potential failures during requirement specification – social interactions between programmers, clients and users are essential (Ozkaya & Akin, 2006).

2.2.2.3. User Interface Design

Wilding (1998) describes User Interface Design (UID) as a composition of art and science and refers to the following key criteria in software interface design: usability, functionality, visual communication and aesthetics. In this context, *"the Graphical User Interface is the dominant concept for building user interfaces. The main design paradigm in a GUI is a rich visual representation of the available information and functions and the possibility of direct manipulation"* (Jani & Schrepp, 2004, p. 53). Ferré et al. (2001) note that, in addition to the visual part, the user interface also includes interaction, which defines the interplay of information exchange and coordination between the system and user. This is an issue, which the UI designers and software developers have to consider during the development process. *"Customer satisfaction with regard to user interfaces becomes increasingly more important and is, eventually, decisive for the selection of systems within a competitive market. End-users [...] expect efficient and optimum support in their work with their interfaces"* (Holzinger & Brown, 2008, p. 217).

Guidelines of Design Designing a UI of software applications requires the need of predefined style guides and rules. *"Usability is high when all components work well together, producing the extra benefits of their synergy"* (Carter, 1999, p. 181). Yet in considering all software usability standards for each individual element of a software project, developers are overextended to observe every single design criterion of the composed instruction sets, which are provided in great variety and whose configuration is still quite complex.

Screen Design Designing usable GUI screens is becoming more important in software development because this is an essential part of an application to enable the (visual) communication between user and system. More precisely, screen design is not only the graphical aspect of a software application but also the guideline on how the user should interact with any program and what the system's reaction will be. The design of both is necessary to provide usability and communication. In addition to implying the advantage of user knowledge, the design process includes the handling of layout techniques, the elaborate utilization of colors, fonts, symbols, icons and controls as well the approach to creating easy-to-use software. The major difficulty is the decision in which innumerable screen elements can be combined as best. Each screen component with which the user interacts and accesses functionality should have a specified mode of operation, and redundant elements should be eliminated from the visual display (Wilding, 1998).

Screen design should be simple in design (Cooper et al., 2007). *"When multiple design elements (controls, panes, windows) are required for similar or related logical purpose, they should be visually rendered in a consistent fashion to take advantage of the concept of inheritance. Inheritance provides the opportunity for an understanding of one element to transfer to other elements that are similar"* (Cooper et al., 2007, p. 308).

Respective Function Icons Function icons provide additional support to the users when they click on them. Cooper et al. (2007) describe the function icon as a *butcon*, which is in fact a combination of a button and an icon. A function icon or *butcon* represents a GUI element, which is recognized quickly and enables a prompt classification for the user. However, this can only take place under the condition that meaningful symbols are created. In general, such function icons do not contain text labeling, because *"the problem is that using both text and images is very expensive in terms of pixels.* [...] *ToolTips provide an effective way to bridge the gap between these two classes of users."* (Cooper et al., 2007, p. 496).

2.2.3. Evaluation of the Design

The final part of the UCD process is the evaluation of the created design against specified requirements (see Figure 22). The idea of a usability evaluation is the estimation of a UID relating to its quality. The results of such tests support professionals in making their design decisions, and the outputs can be used for further improvement (Nielsen et al., 2005). During the design process, several design proposals for the UI are created. An evaluation of the usability of these versions is to obtain feedback so that effective reactions further influence the design process directly (see Figure 23). The form of feedback is a written report that contains the usability problems in detail.

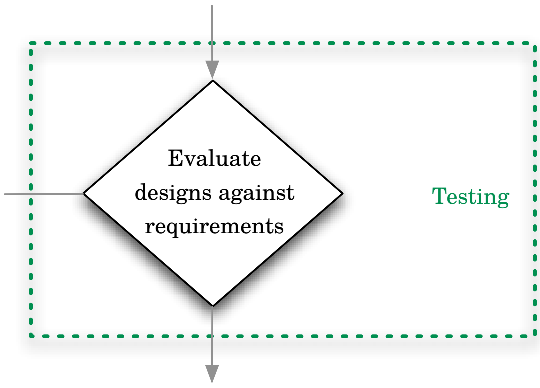


Figure 22: Fourth Activity - Evaluation of the Design

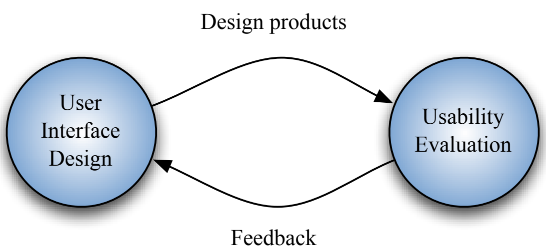


Figure 23: The Interplay between UID and Usability Evaluation
[adapted from (Nielsen et al., 2005)]

In their paper, Nielsen et al. (2005) discuss the different design products, which can be evaluated. Probably, the most common product is the software system itself. An evaluation of the end product is called a *validation test*. The most typical product of a usability test is a prototype (see Chapter 2.2.2) of the system, which does not have all functions at that moment, and such an evaluation is a so-called *assessment test*. Another technique is paper-prototyping, which is recommended in the very early stages of development and this evaluation is characterized as an *exploratory test*.

Constitutive usability evaluation establishes a database, which enables the improvement of effectiveness, efficiency or satisfaction of the product or system that will be developed. Niegemann (2008) considers the usability evaluation as one of the most essential parts in the entire usability engineering (UE) process. *"The most basic elements in the usability engineering model are empirical user testing and prototyping, combined with iterative design. Because it's nearly impossible to design a user interface right the first time"* (Nielsen, 1992, p. 13). Holzinger (2005) adds that alterations and reengineering of the UI can be costlier and more complex to set up in its implementation. If these aspects are taken into account and UCD is considered at an early stage in the development process, it will have quite the opposite effect.

In the next chapter, the classification and types of evaluation methods are explained in greater detail.

2.2.3.1. Usability Testing and Usability Inspection

"Usability testing is an important instrument of the usability engineers' toolkit to analyze an application and make suggestions for usability improvement" (Brinkman et al., 2008, p. 1143). Its difference to usability inspection methods is that usability tests involve representative end users in order to validate the software product. Usability specialists do the usability inspection themselves (Holzinger, 2005; Molich & Dumas, 2008). Inspection methods deliver results of usage assumptions relating to various users. Testing the system with potential users proves these assumptions in practice.

Figure 24 shows the classification of inspection and test methods as well as the associated and most common techniques with respect to their characteristics.

The usability inspection methods include heuristic evaluation (HE), a cognitive walkthrough (CW) and the action analysis. Thinking aloud (THA), field observation and questionnaires belong to the usability testing methods. The following paragraph deals with a short explanation of some of these different techniques as described in (Holzinger, 2005).

	Inspection Methods			Test Methods		
	Heuristic Evaluation	Cognitive Walkthrough	Action Analysis	Thinking Aloud	Field Observation	Questionnaires
Applicably in Phase	all	all	design	design	final testing	all
Required Time	low	medium	high	high	medium	low
Needed Users	none	none	none	3+	20+	30+
Required Evaluators	3+	3+	1-2	1	1+	1
Required Equipment	low	low	low	high	medium	low
Required Expertise	medium	high	high	medium	high	low
Intrusive	no	no	no	yes	yes	no
Comparison of Usability Evaluation Techniques						

Figure 24: Comparison of Usability Evaluation Techniques
[from (Holzinger, 2005)]

Inspection Methods (without Users)

Inspection methods are intended *"for identifying usability problems and improving the usability of an interface design by checking it against established standards"* (Holzinger, 2005, p. 72).

Heuristic Evaluation inspects the interface of a system via usability experts. Usually, three to five specialists are involved in such an HE, but each assessor initially validates the given UI alone. Therefore, the evaluator surveys the UI and compares various interactive components or dialogs with established usability principles, which are called *heuristics*. These usability heuristics, which should be followed by all UI designers, are based on the work of Nielsen & Molich (1990), the revised principles can be found in Nielsen (1993), inter alia.

Cognitive Walkthrough is a task-specific approach that determines the functionalities of the system by an analyst. Cognitive issues are examined and *"CW simulates step-by-step user behavior for a given task"* (Holzinger, 2005, p. 73).

Test Methods (with Users)

Usability test methods obtain information directly from end users concerning their specific problems with the relevant UI.

Thinking Aloud is probably the most common technique in testing usability (Law & Hvannberg, 2004). Holzinger (2006) points out the fact that THA a kind of royal discipline in the fields of UE.

One particular feature is that the involved end users have to think aloud while they are using the system and challenging the prepared tasks of the usability study. The advantage of this technique is the recording and retraceability of individual steps via the verbalization of the thoughts of the test users. Problems can be systematically located and identified. It takes quite some time, but it is a cheap solution for recognizing usability suggestions. Typically, THA is commonly used for testing prototypes, and was used in this master's thesis (see Chapter 4.4).

The tracking of generic facial expressions and gestures is another benefit to the THA technique. By means of video recording or behavioral observation software, this auxiliary method can provide conclusions about the subjective impressions of the test users and their behavior during the usability study (Holzinger, 2006; Holzinger & Brown, 2008).

Field Observation is the study and analysis of the users in their working environment (see Figure 25[12]). This kind of technique *"focuses on major usability catastrophes that tend to be so glaring they are obvious the first time they are observed and thus do not require repeated perusal of a recorded test session"* (Holzinger, 2005, p. 74). The usage of video recording can assist in evaluating the system. Additionally, the observation must not be disturbed; otherwise interferences can affect the results.

Figure 25: User Observation

Questionnaires are certainly an excellent method in determining the subjective satisfaction of end users. Furthermore, the usage of the system by the users, can allow their favorite features or possible concerns to be evaluated using this technique. See Chapter 2.2.1.3 for further information.

[12] `http://www.avl.com` (retrieved on December 20th, 2009)

2.3. Component-Based Software Development

Current developing methods focus on component-based software development (CBSD), which *"has become a significant aspect of improving quality and reducing the cost of the software development process"* (Arhippainen, 2003, p. 9). Crnkovic (2003) specifies this approach as a subcategory of Software Engineering. The extension to this technical term is third-party component-based software development, which involves the usage of external components for in-house development work (Arhippainen, 2003). This aspect is especially considered, because it relates to the thematic complex of this master's thesis. In this chapter, an explanation of a software component is given. The term 'third-party software component' is presented in short detail. Furthermore, the key concept, usage and evaluation of the usability of such components are described.

2.3.1. Definition of Software Component

Literature provides several definitions for the characterization of a software component. For example, a pre-implemented software module can be described as *"a physical packaging of executable software with a well-defined and published interface" (Hopkins, 2000, p. 27).* Another explanation for this term used in research articles is that a *"software component is a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture"* (Mari & Eila, 2003, p. 25).

Third-Party Software Component An extension of this technical term is known as a *third-party* software component.

In general, software systems are developed within a company. Third-party entities (the same applies to third-party software applications) are developed by external individuals or enterprises, and can be reused by the provider of the software system.

2.3.2. Concept and Usage of Third-Party Software Components

To reduce costs as well as the cycle time of developing software (Bertoa et al., 2006), great use is made of the acquisition of reused software components from external companies. Another reason for the use of third-party software components might be that the software system increases continuously in size and complexity and necessitates new technical solutions (Crnkovic, 2003).

In addition to previous statements about software reuse, Arhippainen (2003) refers to the division of two categories, namely developing software *for reuse* and *with reuse.*

Developing the former group, the software is designed and planned in a way such that its implementation may be used in and provided to different environments and contexts. With regard to the second category for developing software components, this type of software engineering involves the integration of reusable software components in order to develop a software system. Typically, additional components from third-party companies are built-in, which have to be adapted in their configurations as well as having to match the specific requirements of the major software system.

This also implies that the software development process needs to be modified in order to generate economic success through the advanced features of the third-party software component. Another benefit is that the companies who apply the usage of third-party software components can concentrate on their own core business (Arhippainen, 2003).

In Arhippainen's (2003) publication, the role-play between software developer and customer is reversed in relation to software engineering. Normally, developers implement software applications for their enterprises. In contrast, the CBSD promotes the switch of developer to a customer, who has to purchase the software component from an external company and expand the company's proprietary software through its integration.

"The use of third-party components elicits new risks to the integrator's software development projects. From the project management perspective, schedule estimation is difficult because of the new third-party component tasks and vendor dependence. The other risk is that a purchased component may turn out incapable. A component may not perform as well as expected (Arhippainen, 2003, p. 22). Moreover, the quality of the existing system cannot be guaranteed with the additional component. To avoid such difficulties, the third-party software component has to be verified relating to its functions and qualitative aspects. Furthermore, reliability is a main prerequisite for a sustainable and stable software system. Several evaluations and tests should be performed following the integration of the third-party application. Continuous contact with the third-party company is also generally recommended (Arhippainen, 2003).

As an executive summary, Figure 26 shows the concept of the third-party CBSD process, which has been explained in this section.

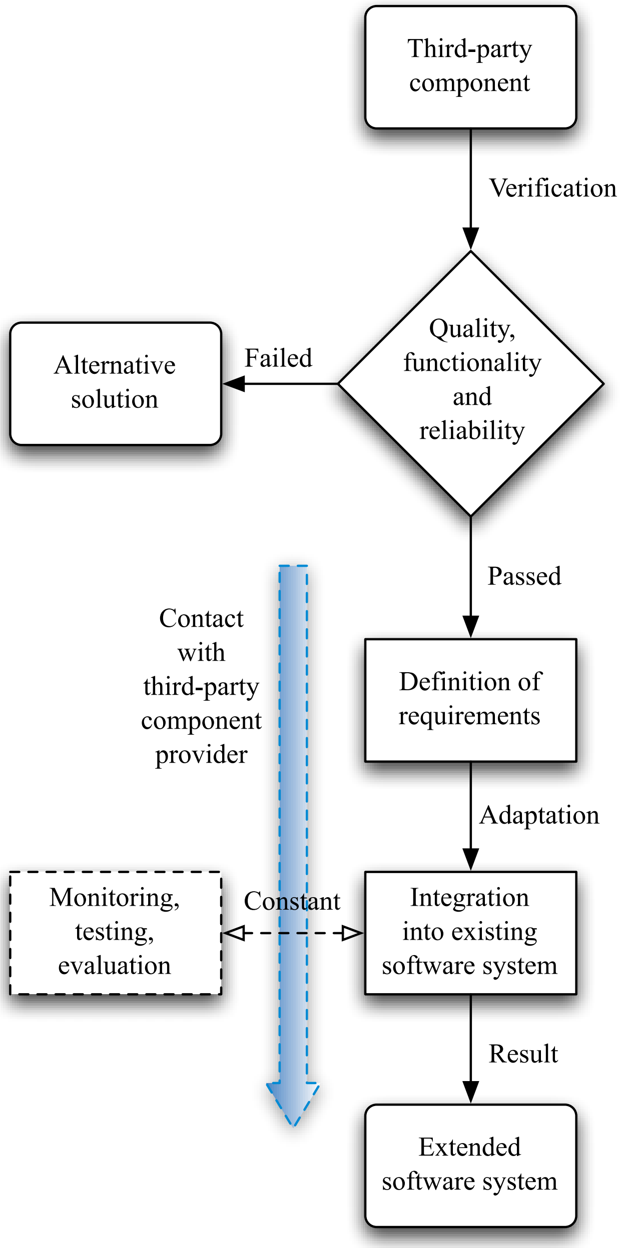


Figure 26: Process of Third-Party CBSD

2.3.3. Usability Evaluation of a Software Component

Only the observable constituent parts of a software component can be used to measure the usability. More specifically, these make up the technical documentation, which mostly consists of manuals, help systems or simply a demonstration, and a few functional elements of the component, which include interfaces, operations, events and also configurable parameters (see Figure 27) (Bertoa et al., 2006).

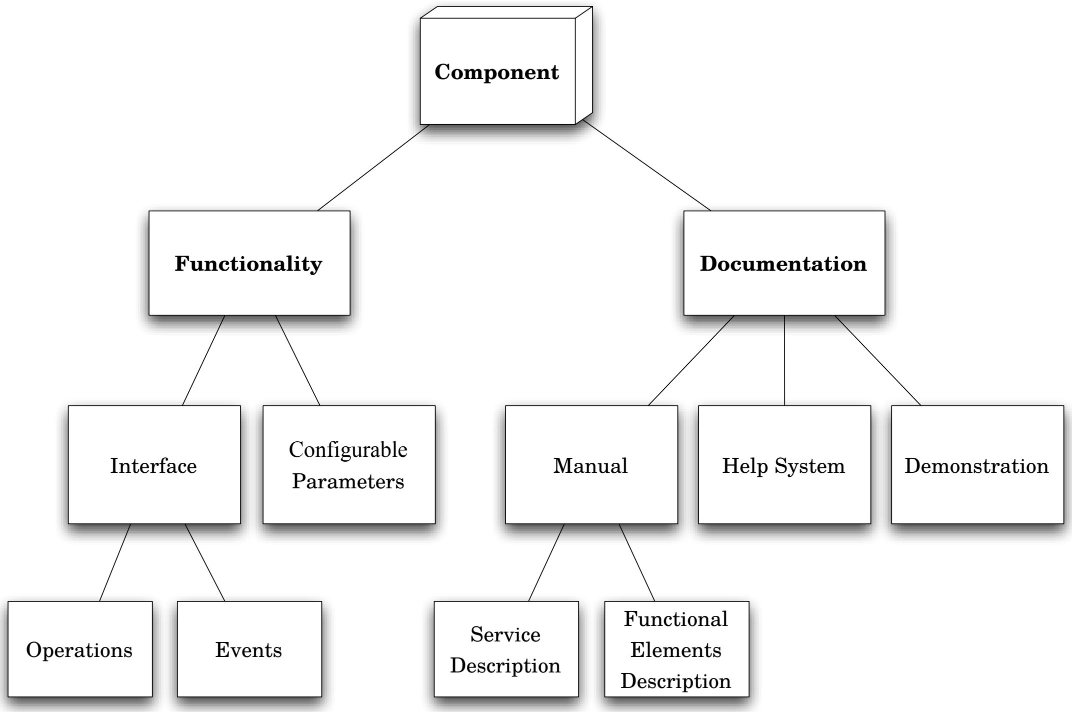


Figure 27: Relevant Information for Usability
[adapted from (Bertoa et al., 2006)]

3. Related Work

The UCD process (see in Chapter 2.2) important for the usability of a software system. In his paper, Bevan (1999) refers to a study, which *"found that 60 % of software defects arise from usability errors, while only 15 % [..] are related to functionality"* (Bevan, 1999, p. 94). According to this statement, the value of user-centered methods is also associated with the *"quality in use (the extent to which the software meets the needs of the user)"* (Bevan, 1999, p. 89). Van Veenendaal (1998) describes the relationship of the quality in use and product quality in his paper. Achieving quality in use depends on satisfying certain criteria for the quality of the product (see Figure 28). It is a fundamental principle that the quality in use links the approach to usability of human factors with UCD (Bevan, 1999).

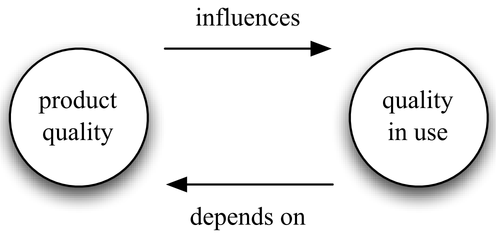


Figure 28: Relationship between Quality in Use and Product Quality [from (van Veenendaal, 1998)]

With respect to high quality UIs, Pleuss et al. (2007) emphasize the importance of prototype development in the very beginning of software projects. *"Creative development tasks - such as user interface design [...] - are usually performed using different tools optimized for the respective tasks"* (Pleuss et al., 2007, p. 241). The authors of this paper present an approach for integrating heterogeneous instruments (they used the tools Adobe Photoshop and Adobe Flash), which support the process of creative design, into model-centric development in order to ensure usability of the software product as well as the expected quality of the UI. Designed models of the target software *"are an excellent vehicle for integrating different stakeholders and different views on the system during the whole development process"* (Pleuss et al., 2007, p. 243).

Another important rule of the UCD is the empirical verification of the resulted software or prototypes, whereby usability test methods and various questionnaires are used to find out the rating of stakeholders relating to the new product (Hassenzahl et al., 2003). This is the main issue addressed in this master's thesis.

"Usability tests have been extensively applied in industry to evaluate a system's prototypes of different levels of fidelity. [...] The primary goal of a usability test is to derive a list of usability problems (UPs) from evaluators' observations and analyses of users' verbal as well as non-verbal behavior. Improvement requests are proposed to systems developers for correcting the UPs thus identified" (Law & Hvannberg, 2004, p. 9). Here it appears useful to add that (Law & Hvannberg, 2002) define a UP *"as a flaw in the design of a system that makes the attainment of a particular goal with the use of the system ineffective and/or inefficient, and thus lowers the user's level of satisfaction with its usage"* (Law & Hvannberg, 2004, p. 71). Related work is also done on this subject by Law & Hvannberg (2002) who examined the comparison of the effectiveness of HE and usability test. In each test, their investigation focuses on the following: HE examined quantitative and qualitative measurements, in contrast, the usability test considered the performance and subjective measures. Referring to this, Figure 29 provides a more detailed overview of this content, which have been presented by the authors.

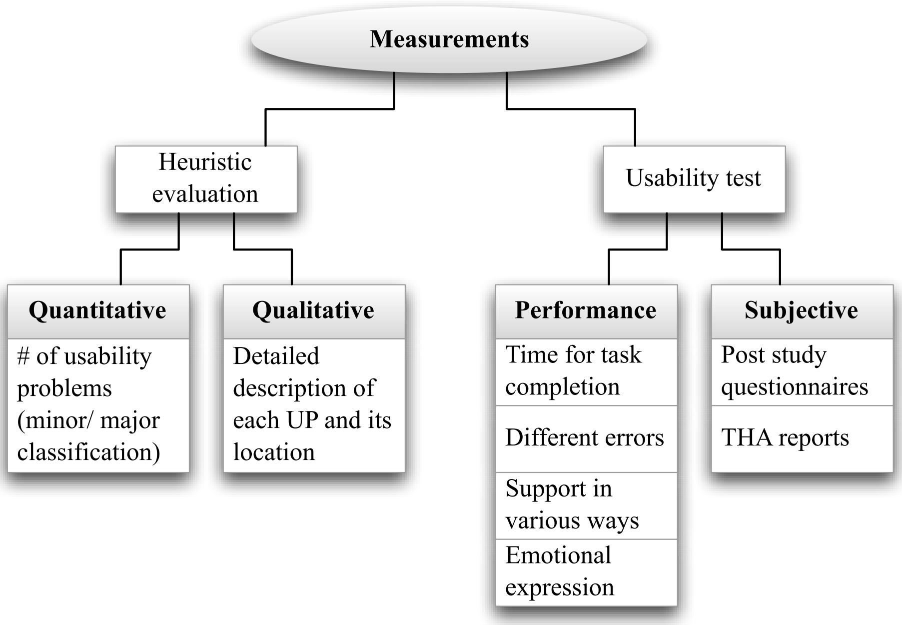


Figure 29: Different Measurement Capabilities
[extracted from (Law & Hvannberg, 2002)]

Furthermore, Law & Hvannberg (2004) describe the binominal probability formula $1 - (1-p)^n$, which was presented in Nielsen & Landauer's (1993) publication. By the use of this formula, the optimality of sample size can be calculated with respect to conduct a usability test, whereas n is the number of test users or reviewers and p is the probability of finding the average UP when using a single, average user (Law & Hvannberg, 2004).

Regarding of the number of participants of a usability test, Nielsen (1993) states that three to five end-users are enough for the identification of 80-90 % of all UPs (Ebner et al., 2006; Law & Hvannberg, 2004; Holzinger, 2005) using the test method THA. Besides, Law & Hvannberg (2004) attempted to investigate the correlation between time-on-task and number of usability problems in their study, because they states following assumption: *"When time-on-task of a specific task is longer than the corresponding benchmarked value, it typically implies the existence of UP"* (Law & Hvannberg, 2004, p. 14). This aspect is also considered in this master's thesis.

With regards to the mentioned attributes of usability (see Chapter 2.1.2), Sauro & Kindlund (2005) describe the measurement of each dimension as follows: *"Effectiveness includes measures for completion rates and errors, efficiency is measured from time on task and satisfaction is summarized using any of a number of standardized satisfaction questionnaires"* (Sauro & Kindlund, 2005, p. 401). Figure 30 shows their quantitative model of usability, which uses four metrics derived from their previously citations, to represent the three dimensions. The measurement scope of these three dimensions also was the subject of the research focused in this master's thesis.

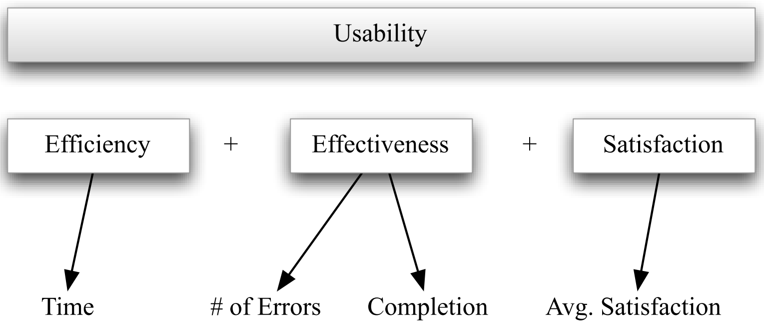


Figure 30: Quantitative Model of Usability
[adapted from (Sauro & Kindlund, 2005)]

Moreover, in this section, this thesis discusses literature (compare (Sauro & Kindlund, 2005), for example) which deals with using questionnaires in order to gain valuable knowledge of the user and his software system usage. Schrepp et al. (2006) conducted empirical research regarding the usage of questionnaires. By means of a study based on the questionnaire AttrakDiff™ (compare Chapter 2.1.2.3), the authors examined whether or not *"existing results related to hedonic aspects of a user interface apply also to business management software"* (Schrepp et al., 2006, p. 1056). Therefore, it is understood that this business software is utilized to support the users in completing their daily tasks. The test participants of this study had to evaluate three variants of user interfaces with the same functionality concerning their attractivity. The results could be confirmed: Both pragmatic and hedonic qualities influence the attractiveness of a user interface. The experimental results also provided evidence that the preferences of users increase, the higher the attractivity of a user interface is. In this context it must also be mentioned that the AttrakDiff™ questionnaire was used in this usability study of this work.

Lewis (1995) also formulates some subjective measurement instruments for usability, which are relevant for and used in this thesis. The IBM Post-Study System Usability Questionnaire (PSSUQ, refer Appendix A.5.1) and Computer System Usability Questionnaire (CSUQ), to quote only two, are both developed and validated by Lewis (1995). The completion of the PSSUQ presents an overall evaluation of the used software system relating to the tasks and scenarios of a performed usability test. The answering of the PSSUQ is usually filled out after the task completion within a study. The CSUQ is similar to the PSSUQ, but there is a minor modification. Instead of using the terms *"tasks and scenarios"*, the CSUQ contains the wording *"my work"*. If the usability study is not a laboratory experiment and, for example, it is carried out in a company to evaluate the user satisfaction of the used software system, the application of the CSUQ is recommended (Lewis, 1995). As a result, four scores are generated through the CSUQ and PSSUQ units, which are shown in Table 2.

No.	**Scores**	**Abbreviations**	**Appropriate Items**
1.	Overall Satisfaction	OVERALL	Questions 1 - 19
2.	System Usefulness	SYSUSE	Questions 1 - 8
3.	Information Quality	INFOQUAL	Questions 9 - 15
4.	Interface Quality	INTERQUAL	Questions 16 - 18

Table 2: Different Scores of the PSSUQ/ CSUQ
[adapted from (Lewis, 1995)]

Several other studies deal with the development of methods for measuring user satisfaction. The SUMI method (refer to Paragraph 2.1.2.2) is another subjective solution applied to gather user satisfaction relating to the software system. The SUMI questionnaire *"provides a valid and reliable method for the comparison of (competing) products* *and differing versions of the same product, as well as providing diagnostic information for future developments"* (van Veenendaal, 1998, p. 2). Furthermore, in addition to the introduction of the SUMI technique, van Veenendaal's (1998) paper presents three practical applications concerning the usage of SUMI for testing the usability.

Lin et al. (1997) propose the Purdue Usability Testing Questionnaire (PUTQ) that is a technique for comparing the relative usability of different software systems. The research of this study qualifies eight human factor principles based on the theory of human information processing. These specified considerations (compatibility, consistency, flexibility, learnability, minimal action, minimal memory load, perceptual limitation and user guidance) are relevant to the usability of software. The questionnaire is composed of 100 items. The calculation of an intelligence index for the usability of an interface is provided, which is based on the score received from the evaluation, and which is set according to the highest possible score. The authors also state the limitation, that the *"items in PUTQ are mainly focused on conventional graphical user interface software which requires visual display, keyboard, and mouse"* (Lin et al., 1997, p. 274-275).

In Ryan & Gonsalves' (2005) empirical study, the effect of context in mobile applications is tested. It is obvious that such content of mobile usability in this paper is not directly comparable to the usability of software components as referred to in this work. Nevertheless, by means of within-subjects design, four different configurations have been tested. The researchers examined the effects of context and types of use relating to the objective as well as subjective usability attributes. They concluded that, in contrast to the results of the other tested application types, the mobile web based implementation performed comparatively poorly because it *"was unable to take advantage of location context or client-side application code"* (Ryan & Gonsalves, 2005, p. 115). This master's thesis also deals with such similar content, which regards testing two software components relating to their subjective usability attributes (compare Chapter 2.1). Beside the focus on the subjective aspects of usability, the attractiveness of each software system and the users' level of satisfaction are measured in this work. *"Recent studies suggest that the weighting of both aspects in forming an overall evaluation of an interactive product heavily depends on features of the actual situation, such as whether an individual has to perform a specific task or not"* (Hassenzahl et al., 2008, p. 473).

Nielsen (1993) refers to a study, which concerns evaluating the usability of several icon designs. The focus was placed on gaining user feedback on the icon designers. In this usability test, four different set of icons for a GUI were provided, all of which were designed by different design experts. The requirement was to find a suitable icon set for the specified software system. The proposed icon sets were analyzed relating to the usability attributes' ease of learning, efficiency of use and the subjective user satisfaction (see Chapter 2.1) by performing several tests. After completion of the assessment, it was possible to compare the four different sets of icons by considering the results of the evaluation. Finally, another set of icons was designed for the system. It was based on the set, which had obtained the best results in the usability test.

A conclusion of research carried out by Carter (1999) is that various usability guidances have been designed to improve the usability of the different components, such as windows, dialogs or menus. For example, the seven general design principles are named (see Table 3), which are specified in *ISO 9241 (Part 10)* to achieve usable systems.

1.	Suitability for the task
2.	Self-descriptiveness
3.	Controllability
4.	Conformity with user expectations
5.	Error tolerance
6.	Suitability for individualization
7.	Suitability for learning

Table 3: Seven Design Principles
[extracted from (Carter, 1999)]

Molich & Nielsen (1990) classified various usability heuristics, for example the principle of a *Simple and Natural Dialogue: "Dialogues should not contain irrelevant or rarely needed information. Every extraneous unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility. All information should appear in a natural and logical order"* (Molich & Nielsen, 1990, p. 339).

Carter (1999) describes the *Usability First* methodology, which supports the preparation of usable guidelines and the improvement of the development process. This technique can be combined with other software development procedures and methods. In addition, this approach includes permanent evaluations in the life cycle relating to the usability of the methodologies for developers as well as of applications, designs and developed systems for the users. In addition, the *Usability First* approach specifies several activities for developers, such as using task analysis methods, the identification of guidelines and standards, creating use models in order to transform requirements into design and evaluating the design on its standardized norm.

4. Materials and Methods

Within this thesis, an empirical research study is presented relating to the comparison of two software systems in order to determine whether the integration of a third-party software component into a larger system improves its usability for users familiar with the existing system (see Figure 31).

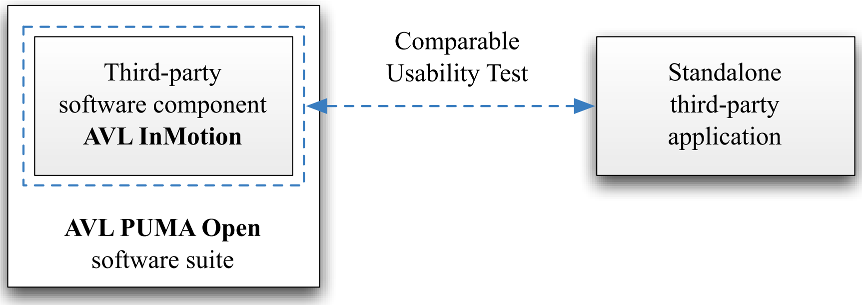


Figure 31: Comparable Usability Test

This chapter focuses on the design of the target component software (see Chapter 4.2.1) and the usability test (see Chapter 4.4) in order to obtain reproducible results concerning the usefulness of both software systems. The usability test took a closer look at details, especially at the attractiveness and subjective attributes of learnability, efficiency, memorability, error tolerability and satisfaction of the two tested software systems.

4.1. Analysis of the Third-Party Software Component

In the following section, the integration of the third-party software component is considered from the viewpoint of the UCD. Such integration is often particularly associated with alterations to the design.

4.1.1. Customization and Design

The integration of an additional and externally developed software component into an existing software system, which has often a complex structure, is usually associated with problems. Multiple causes can be named, such as incompatible system architectures and frameworks as well as communication problems or various views on goals between developers (Larsson et al., 2004). *"The target is to integrate components into a product and to ensure that the product works appropriately so that it can be delivered to customers"* (Larsson et al., 2004, p. 431).

In the majority of cases, it is practically impossible to integrate a third-party software component without making modifications to suit the particular needs. It is also a prerequisite for AVL to adapt the delivered standalone third-party application for the integration into their proprietary existing business software suite, especially considering that the corporate design of the third-party software component must be uniform to AVL software products.

To ensure good usability, Dellarocas (1997) notes that the designers themselves are usually responsible in addition to producers, consumers and third parties. Moreover, the usability requirements have to be fixed preferentially at the design time, because *"the exact meaning and range of usability considerations varies with each kind of resource"* (Dellarocas, 1997, p. 6).

4.1.2. Target Application

The target system is the software component AVL InMotion that supports hardware-in-the-loop (HIL) tests of hybrid vehicles at early stages of the development process. This software solution simulates virtual maneuver and event based testing at the test bed, as shown by the Figure 32[13]. Additionally, results are delivered at the very beginning of the development process by analyzing different elements, for example combustion engines or hybrid systems.

[13] `http://www.avl.com` (retrieved on December 20th, 2009)

Figure 32: Maneuver and Event based Testing via AVL InMotion

At the beginning, the three-dimensional simulation application was represented by an external software component, and AVL integrated it into their existing business software suite named AVL PUMA Open, which is a solution for testing engines, transmissions and power trains.

The next section provides a brief description of the functionality, or rather the key elements, of the software component AVL InMotion, which was developed in combination with the support of a lo-fi prototype (see Chapters 2.2.2.1 and 4.2).

4.1.2.1. Maneuver Based Sequencer

A *Maneuver Based Sequencer* (MBS) for AVL InMotion represents a test block in which settings such as road, driver and maneuver of the used environment can be defined. Furthermore, the MBS requires only a few parameters for a realistic simulation. The MBS-data set is typically used as a test parameter in AVL's PUMA Operator Interface for the parameterization of a test run.

The Implemented Use Case replaced the *AVL ISAC Driver* with the *IPGDriver Standard*. This special IPGDriver Standard is aware of the road geometry and thus can perform maneuvers like avoiding obstacles, cutting corners or (reverse) parking.

Road specific Settings An extended functionality is the parameterization of the road for the simulation of a test run. The resultant construction is a three-dimensional model of the road, where the trajectory of road as well as its width and friction can be defined.

Moreover, road segments can be added to the MBS, which can be modified in their geometric properties, such as straight lines, curves and so on. It is also possible to regulate the starting point of the road for the vehicle as well as the driving lane or distance.

Needless to say, it is possible to verify the specified street definition, which has been modified, either through the *3D Movie Preview* of the road (watching the road characteristics in detail, especially the longitudinal and lateral slopes) or using the *Bird's Eye View of Road*, to get a geometrical illustration in a two-dimensional view.

Maneuver specific Settings Additionally, the creation of maneuvers is possible in the *Road Definition* dialog of AVL InMotion for both analyzing the behavior of the hybrid vehicles by driver activities (for example braking) and adding some system interferences or failures.

Driver specific Settings Maneuvers are performed by a *Driver*, which is a model that acts exactly the way in which a real driver would. The virtual driver consists of a control in order to follow a course and a speed controller.

Basically, there are different types of driver next to *AVL's ISAC Driver*, which reuses the existing trigger criteria and switch curves: The *IPGDriver Standard* and the *IPG-Driver Racing*. There is a difference between these models. The settings for each parameter of the IPGDriver Standard can be parameterized individually (for example, this driver can decide when it is better to shift sooner to get more power at the end of a curve), as opposed to the second (IPGDriver Racing), which calculates the physical limits of the driver after a learning phase. Multiple parameter settings can be managed in the *Driver GUI*, which will be described in the next paragraph.

The maximum cruising speed of the driver can be specifically adjusted in the "General Parameters" section to keep the vehicle on track. Furthermore, the *corner-cutting coefficient* can be parameterized to provide a check on how the virtual driver has to cut the curves. The *maximal longitudinal* and *lateral acceleration* of the driver can be set and finally, it is also possible to adjust the *engine speed* at which the driver shifts up or down.

Movie The three-dimensional view shows an animation movie of the current simulation (see Figure 33), which can also be stored. The usage of the 3D movie in the offline mode represents only a preview of the created road. For example, in contrast to the online simulation, this road preview does not display a car model as shown in Figure 34. In addition, the 3D road preview does not show any changes in speed and the vehicle does not stop, even if a stop sign is configured on the current track. During the simulation, the virtual world is displayed as is precisely specified. Furthermore, the view of the camera perspective is changeable. Additionally, the tire forces of the vehicle can also be displayed via colored bars (see Figures 35 or 36), which can be validated with or without the vehicle body. The simulation speed can be aligned, in real-time or slower/ faster than in real-time. Information on the distance is also displayed in the movie window.

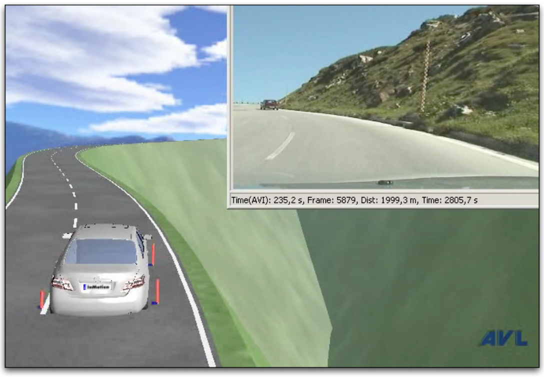


Figure 33: 3D View of Digitized Test Track

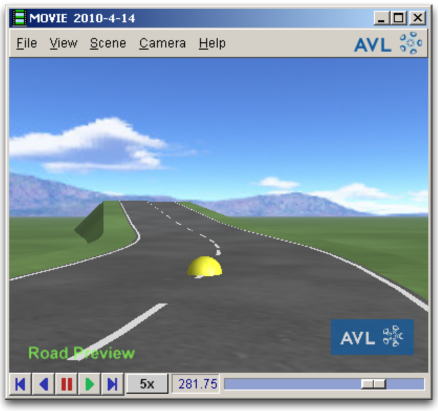


Figure 34: 3D Road Preview, Offline Mode

Figure 35: Tire Forces with Vehicle Body

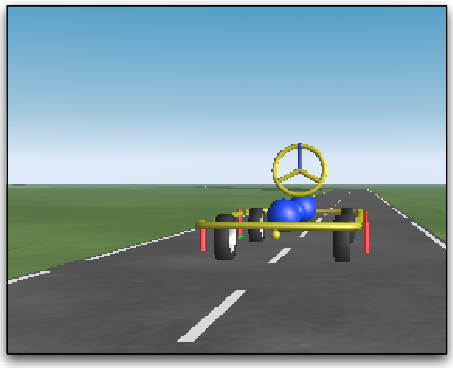


Figure 36: Tire Forces without Vehicle Body

Online Instruments The most useful data are displayed graphically. Amongst others, a tachometer indicates the mode of operation in the online environment (see Figure 37).

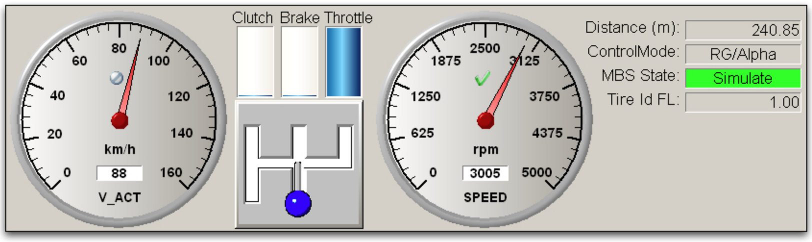


Figure 37: Useful Data are displayed in Online Panel

4.2. Prototype Development of AVL InMotion

The main objective for the construction of a prototype for the software component AVL InMotion was the development of a user-friendly graphical user interface.

The following Chapter 4.2.1 provides numerous design suggestions for the GUI of AVL InMotion. These first resulting design proposals of the prototype were the representation of a design idea of the software component, which was built on the basis of the standalone third-party software IPG CarMaker[14] in order to get the main features and components for the intended UI. See Figure 38[15], which displays the simulation environment of the standalone third-party application IPG CarMaker.

Beyond reducing and avoiding having many individual windows as well as confusing menu entries, however, the question that arises is how the creation of a less complex UI for AVL InMotion can be achieved. Based on usability heuristics, the standalone third-party software has been reviewed in the first step. Furthermore, the identified requirements determined the required parameters, which are essential in order to develop a realistic model for the design of AVL InMotion.

The final prototype, a click dummy (compare Chapter 2.2.2), was developed with HTML in combination with a GUI-Builder software tool and represented the planned software design including an interaction with the main features required from the target system. It enabled the navigation through the different main settings and their submenus by tabs. This lo-fi prototype served as a guideline for the implementation of the software component AVL InMotion.

[14] http://www.ipg.de/

[15] The colored circles in red indicate some problem areas.

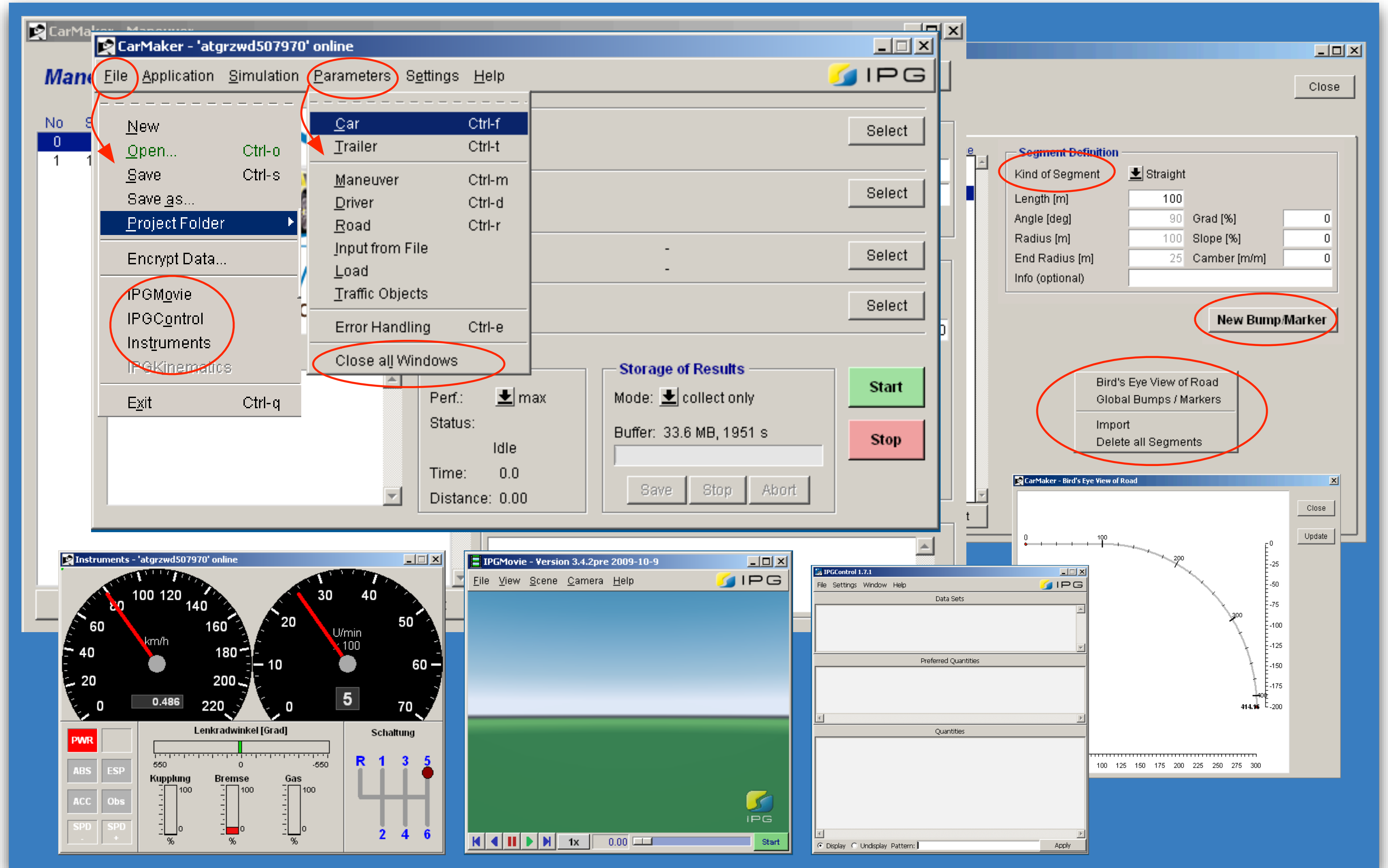


Figure 38: A few Windows of the Third-Party Simulation Software IPG CarMaker

4.2.1. Design of Graphical User Interface

The simulation software of IPG CarMaker was the technical basis for the prototype of AVL InMotion, as mentioned previously. On closer inspection of the existing software design, it was not easy to get a design structure immediately. This standalone third-party application is composed of individual windows, which are only accessible by using the context menu of the software application.

The difficulty was to distinguish the most important and main elements of the application as well as to inspect each window in order to create the new graphical user interface for the application AVL InMotion, which should contain all key parameters and collect all functions in one common frame.

It is obvious that several design variants could be possible. For example, the dialog could consist of subtabs or be designed in wizard mode (see Figure 39).

Finally, the idea for the design of AVL InMotion was to divide the entire dialog into four main categories, which include the *road*, *driver*, *maneuver* and *simulation* settings (see Figure 40). Each of the specified categories form a register on the left-hand side of the dialog, and, furthermore, the appending parameter settings should be easily controllable in the corresponding register.

Next, several drafts of the different registers will be presented, which helped the developers to produce the first layout of the software component AVL InMotion (compare Chapter 4.3).

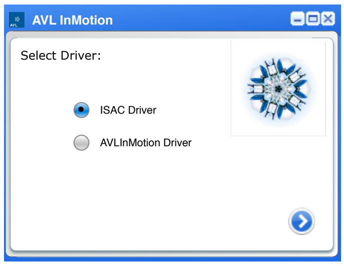


Figure 39: Possible Design, Wizard Mode

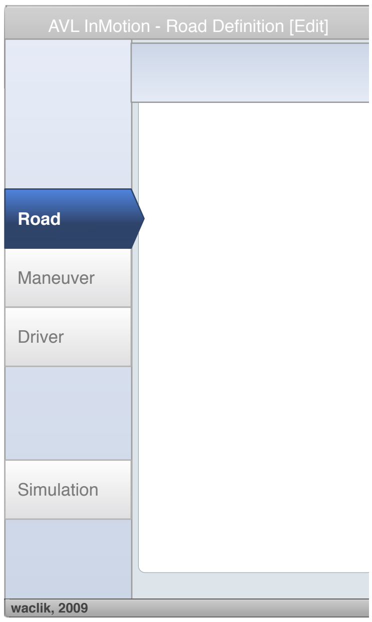


Figure 40: Main Categories of the Dialog

4.2.1.1. First Drafts of AVL InMotion

Design Solutions for the Roads Specific Settings

In this section, mockups are provided for the possible design of the road settings.

The register of the road settings is sub-divided into three categories, namely *General Parameters* (see Figure 41), *Segments* (see Figures 42, 43, 44, 45, 45, 46, 47 and 48) and *Movie Interface* (see Figure 49). The main focus was on the second tab *Segments*, for which eight design recommendations were produced. Mainly, the list view of segments was modified several times as well as the arrangement of each parameter sets.

In the text below, each mockup is explained in greater detail.

<u>Draft of Road Settings - Tab *General Settings*</u>

Figure 41 represents the first tab of the register of the road settings in order to set the general settings for the MBS, such as driving lane or track width, for example. It should be mentioned that there is no essential difference to the original content.

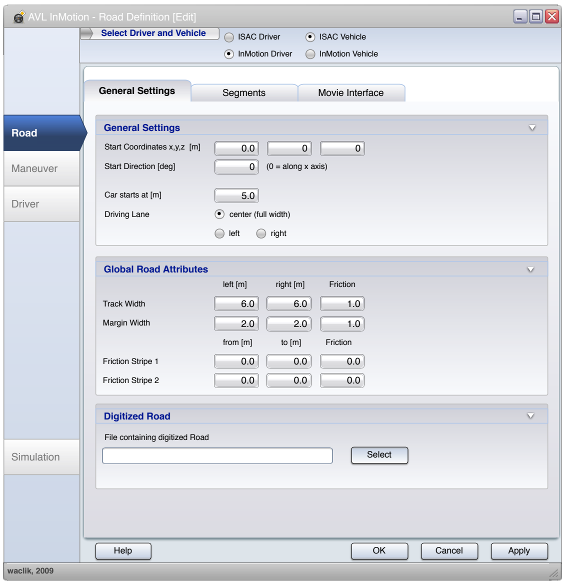


Figure 41: Draft of Tab General Settings

<u>Draft of Road Settings - Tab *Segments* #1</u>

The second tab *Segments* defines the route sections or overrides specified segment attributes (see Figure 42). In this and the following design ideas, the arrangement of all required definition fields and also the Bird's Eye View of Road are clearly visible at once. Note that the new label *"Type"* has replaced the label *"Kind"* in the drop-down list. For better recognizability, icons have been used instead of the control buttons (such as `Copy`, `Paste`, `Delete` and so on), because this is more well-known and the common style for these named operations.

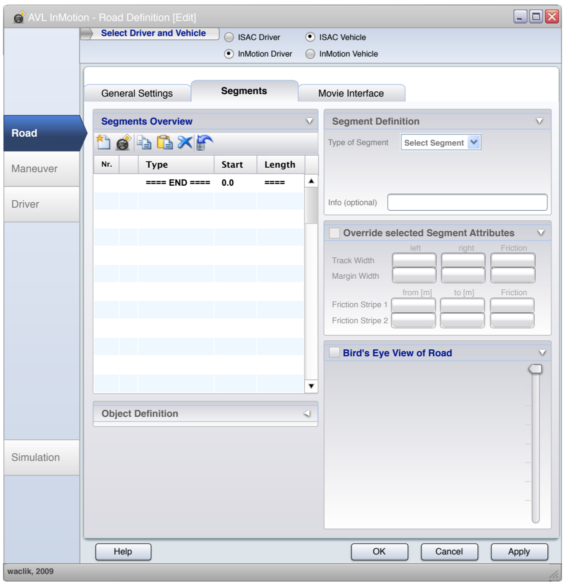


Figure 42: Draft of Tab Segments #1

The focus of the following figures is on the design of the segments list, which is placed to the left of the dialog content. The information on the selected road segment is displayed at the top right of the dialog; the settings for an added marker or obstacle are positioned to the left at the bottom. Furthermore, the *Bird's Eye View of Road* is updated automatically after adding or changing the segments list if the user has marked the checkbox for displaying the geometric view of road.

<u>Draft of Road Settings - Tab *Segments* #2</u>
In this draft (see Figure 43), each added segment is sequentially positioned and is clearly visible, while the added markers and obstacles are nested within the corresponding segment. Markers, such as road signs, traffic signs and obstacles, can be integrated when setting up the road in order to simulate the traffic situations imaginable. For example, wind effects can be simulated or a stop sign can be added to the road as shown in Figure 43.

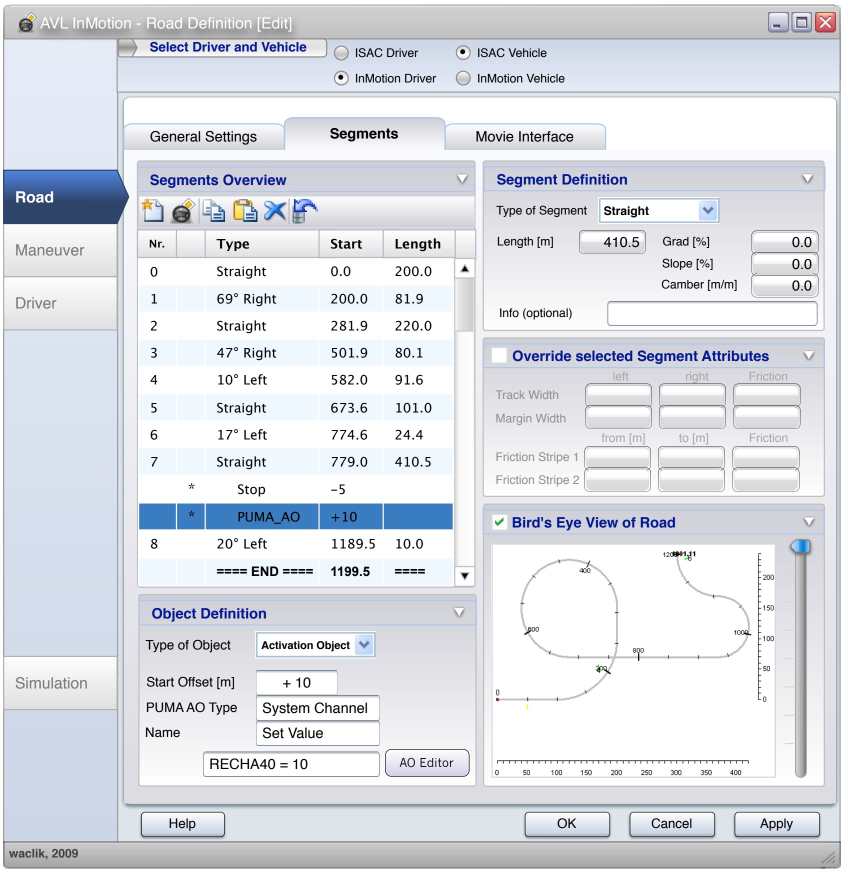


Figure 43: Draft of Tab Segments #2

Draft of Road Settings - Tab *Segments* #3

Another approach using the segments view is the usage of nested sets (see Figure 44); to view the road markers and obstacles of these road segments, click on the arrow icon to unfold the details in a tree structure. Additionally, all shown arrows that are highlighted in the color blue contain object entities, which are interleaved with the marked segment.

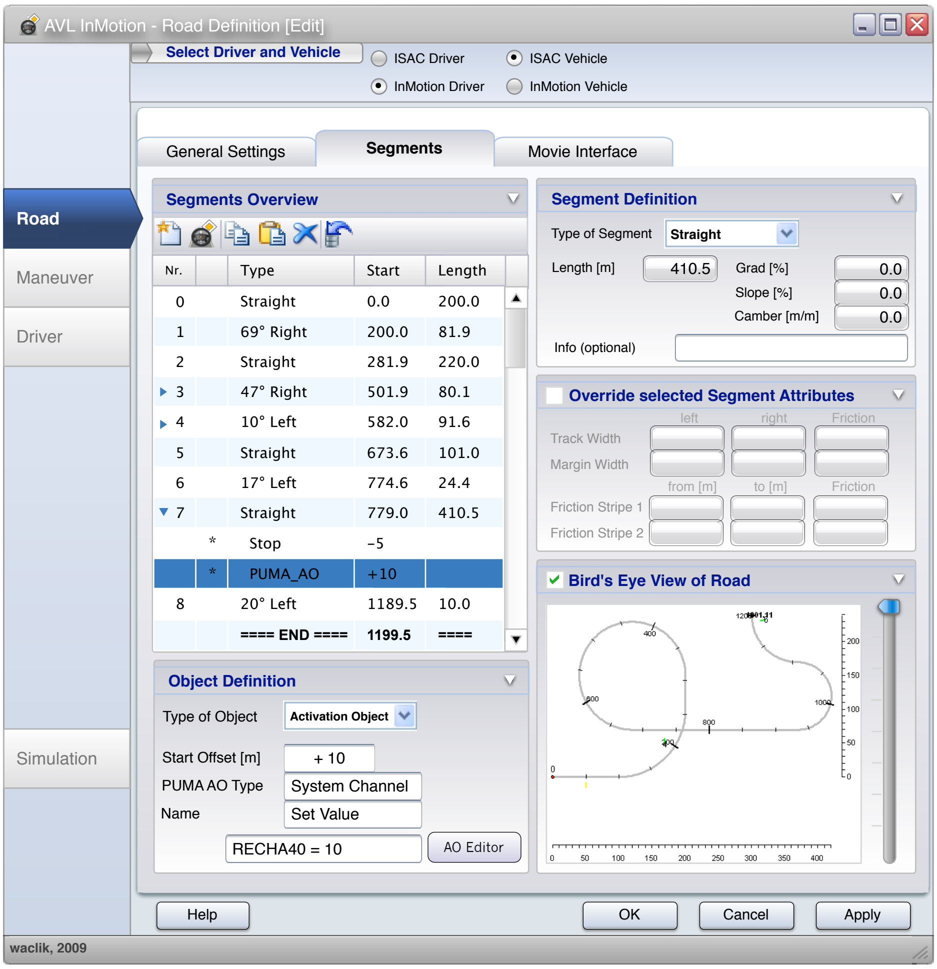


Figure 44: Draft of Tab Segments #3

<u>Draft of Road Settings - Tab *Segments #4*</u>

Another idea for the design of the list view of segments is to emphasize the selected entity in terms of color (see the gray and blue highlighted rows in Figure 45). Additionally, symbols can be used for particular elements, according to the segment, marker or obstacle.

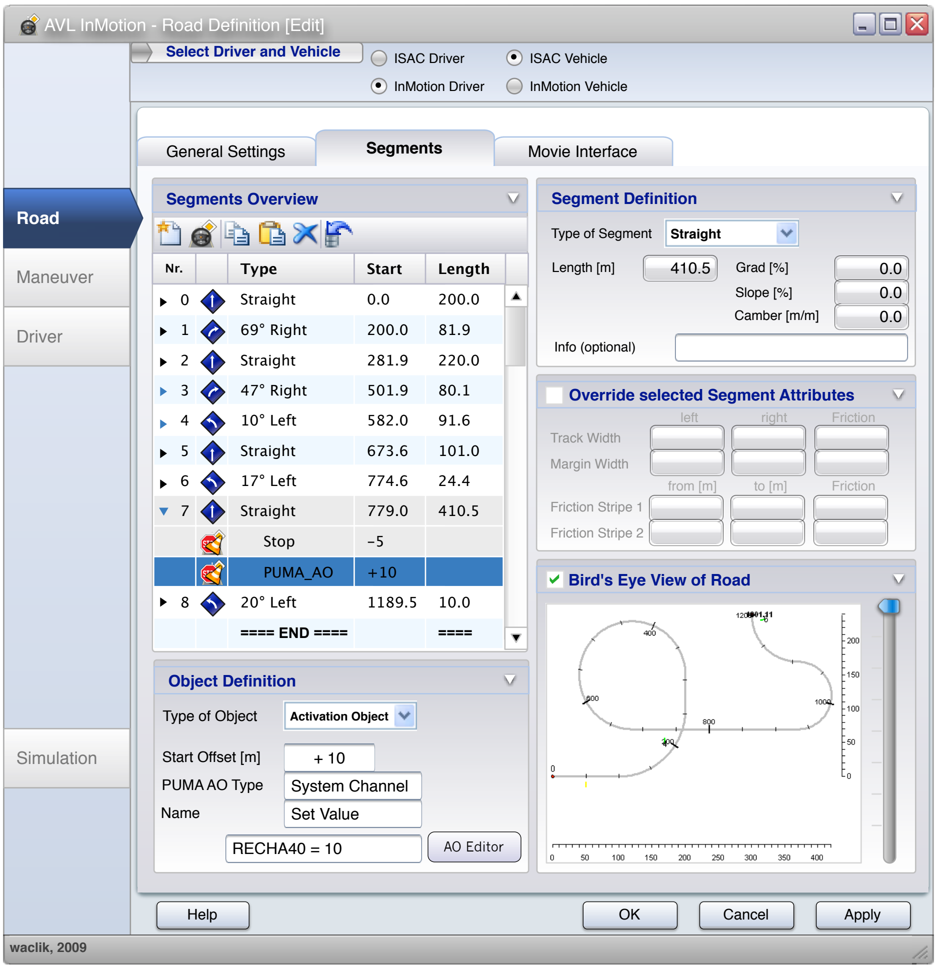


Figure 45: Draft of Tab Segments #4

<u>Draft of Road Settings -Tab *Segments* #5</u>

The next approach to represent the segments is the usage of the tree hierarchy (see Figure 46). There are no additional definition fields for the segments, markers or obstacles settings. The modifications are made within the tree view (see Figures 47 or 48) in order to change the length or gradient of the segment, for example.

A further variant designing the nested sets is that each node element is preceded with a minus or plus sign, which, when clicked on, expands and collapses the road segments immediately (see also Figure 48).

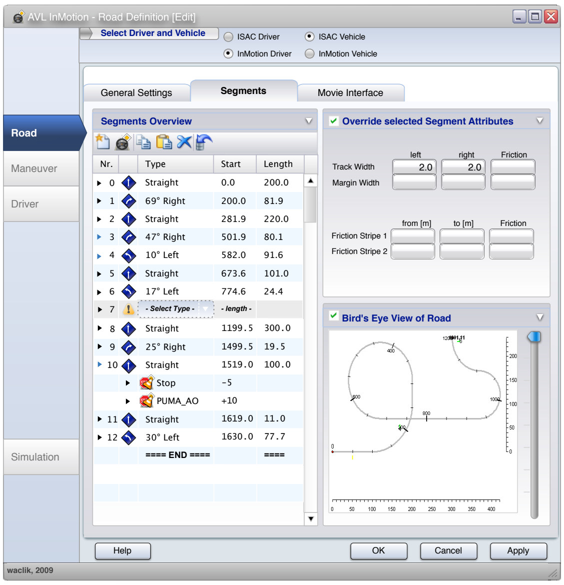


Figure 46: Draft of Tab Segments #5

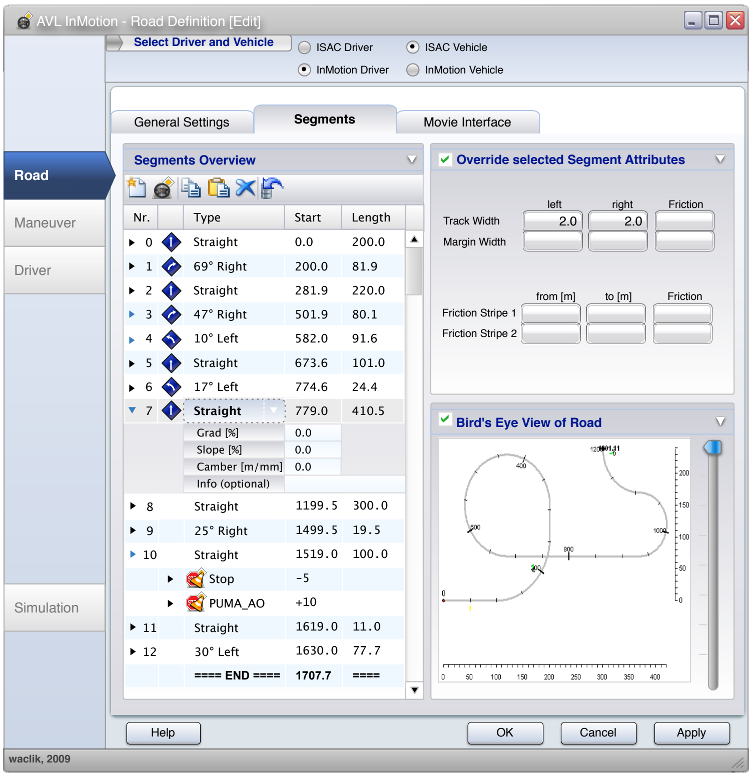


Figure 47: Draft of Tab Segments #6

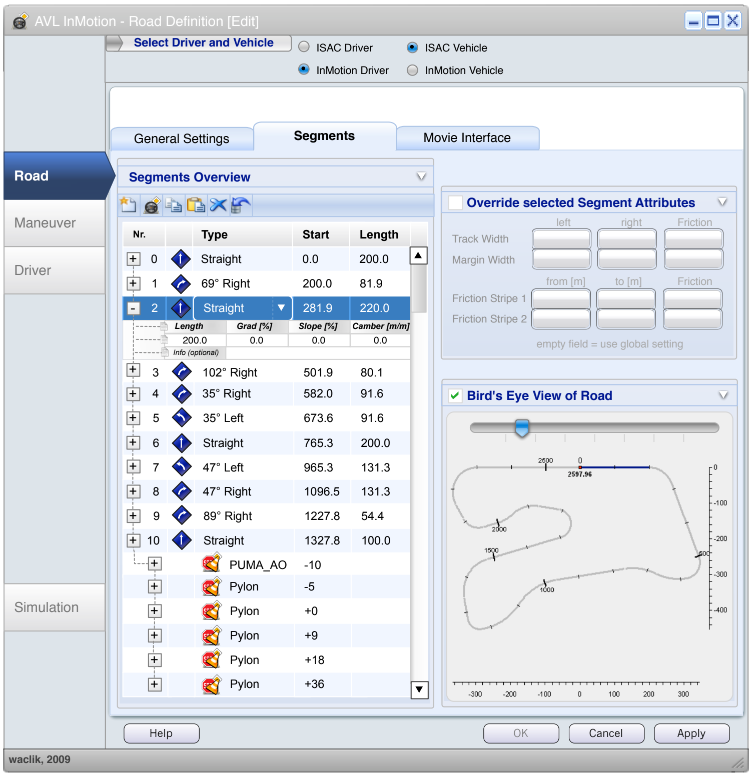


Figure 48: Draft of Tab Segments #7

<u>Draft of Road Settings - Tab *Movie Interface*</u>

The third tab of the *Road* register contains the settings for the movie playback (see Figure 49).

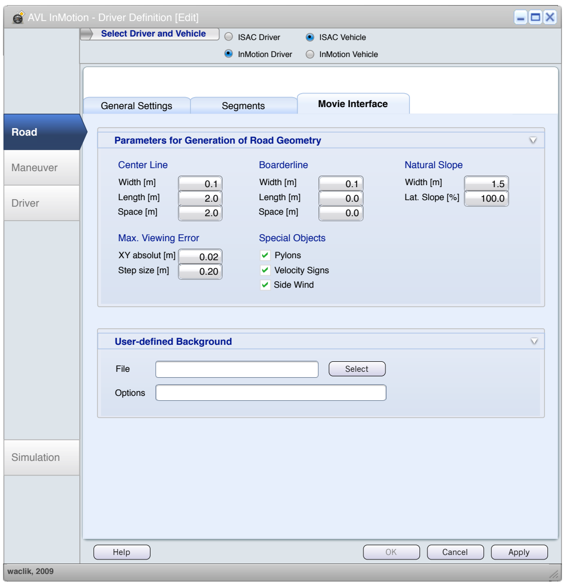


Figure 49: Draft of Tab Movie Interface

Design Solution for the Maneuvers Specific Settings Events for individual route sections can be scheduled in this *Maneuver* register (see Figure 50). The list of the maneuver steps is similar to the design and functionality of the road segments list. All definition fields are also well-arranged in this design concept. The control buttons (such as `Copy`, `Paste`, `Delete` and so on) have been transformed into icons, because this is more well-known and is the general style for these named operations as already described before for the *Road* register.

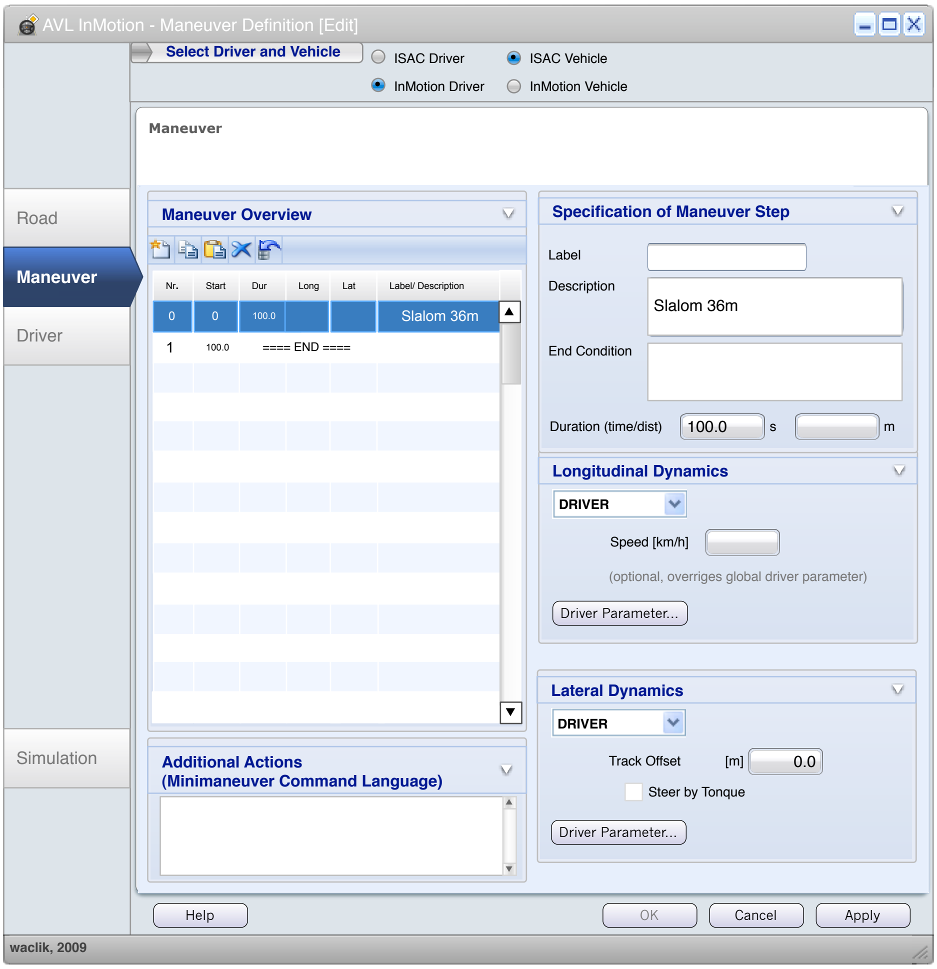


Figure 50: Draft of Maneuver

Design Solutions for the Driver's Specific Settings First of all, the driver model must be selected in the MBS dialog at the top. Depending on the selection, the appropriate edit fields are enabled in the *Driver* register.

The general parameters for the driver model can be regulated in the first tab *Standard Parameters* of the *Driver* register (see Figure 51). Figure 52 (racing parameters) and Figure 53 (adding extra parameters optionally) display the design of the fields for the various settings, which are also well-arranged in these design concepts.

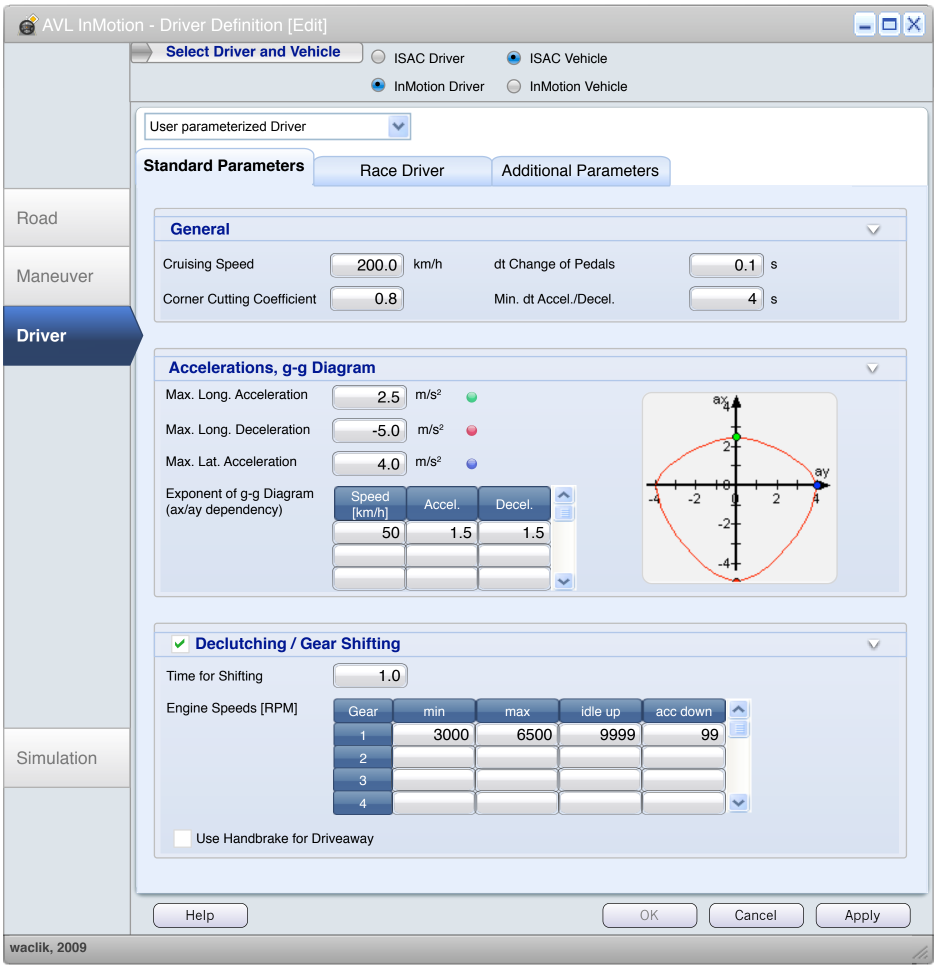


Figure 51: Draft of Tab Standard Parameters

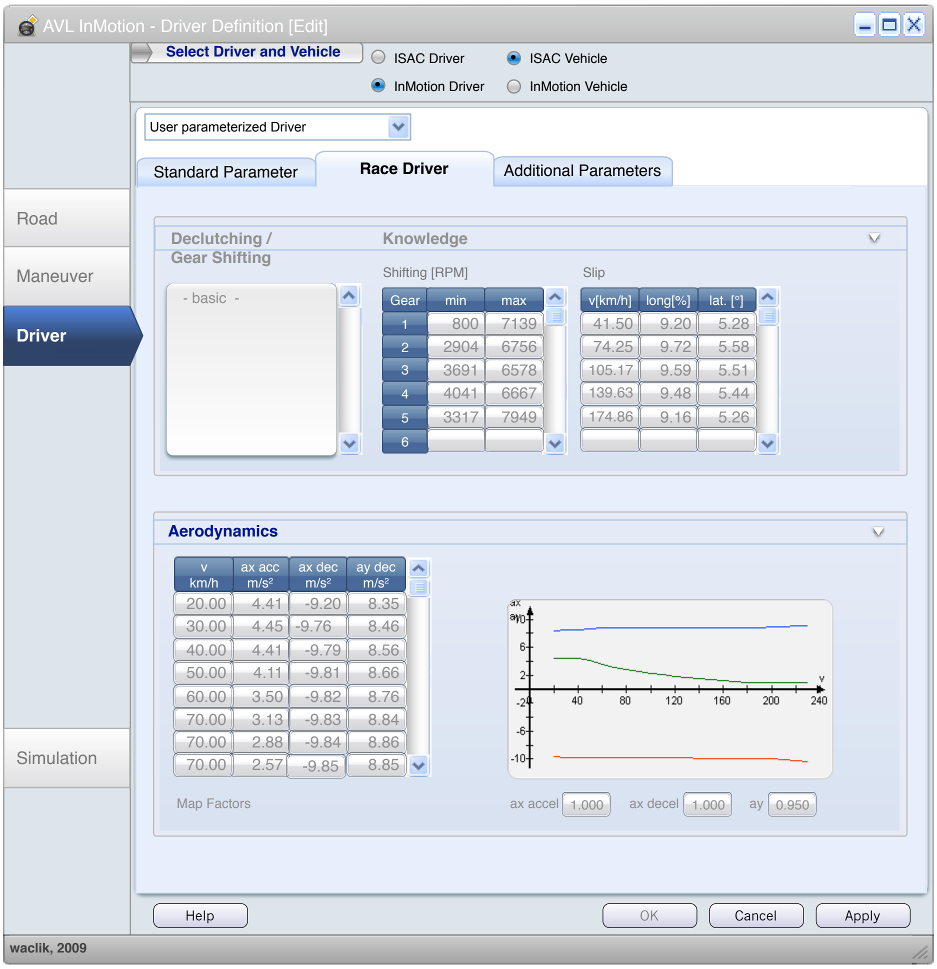


Figure 52: Draft of Tab Race Driver

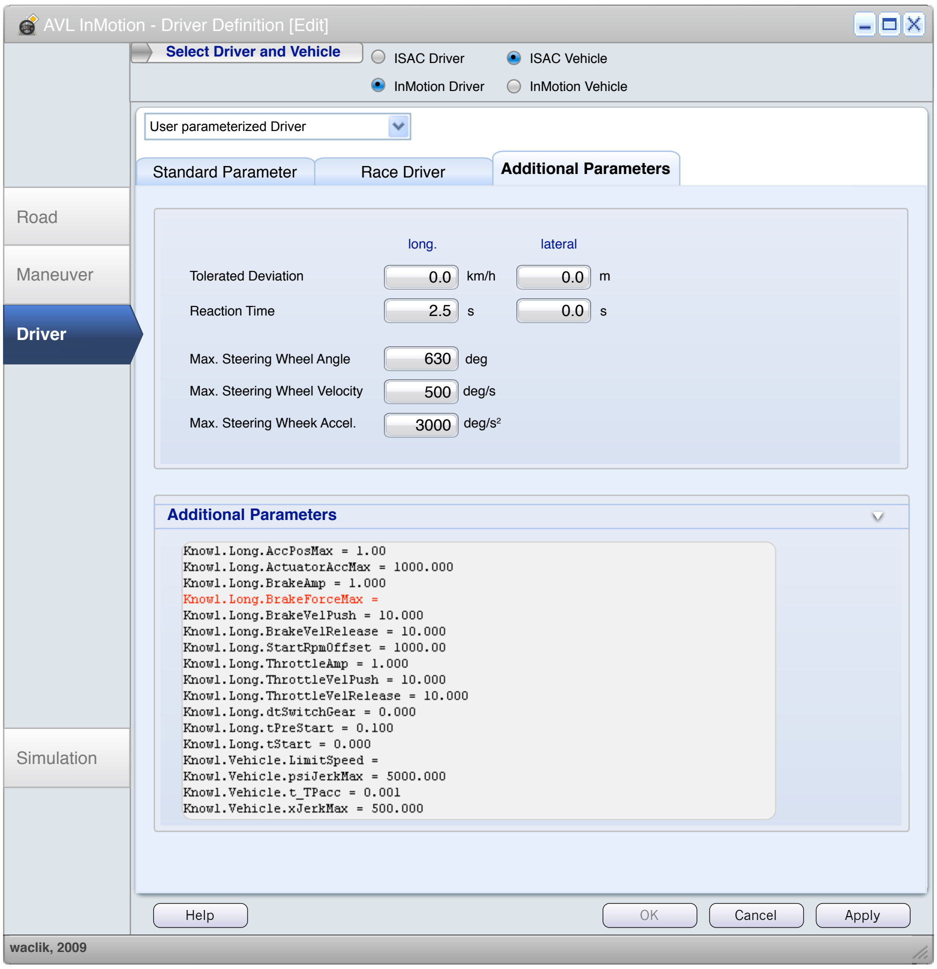


Figure 53: Draft of Tab Additional Parameters

Design Solution for the Simulation View

In the layout design of the *Simulation* register (see Figure 54), the main settings for the simulation are summarized. Details about the selected car model as well as the performance are displayed in this view. The green `Start` button starts and the red `Stop` button stops the three-dimensional simulation of the road.

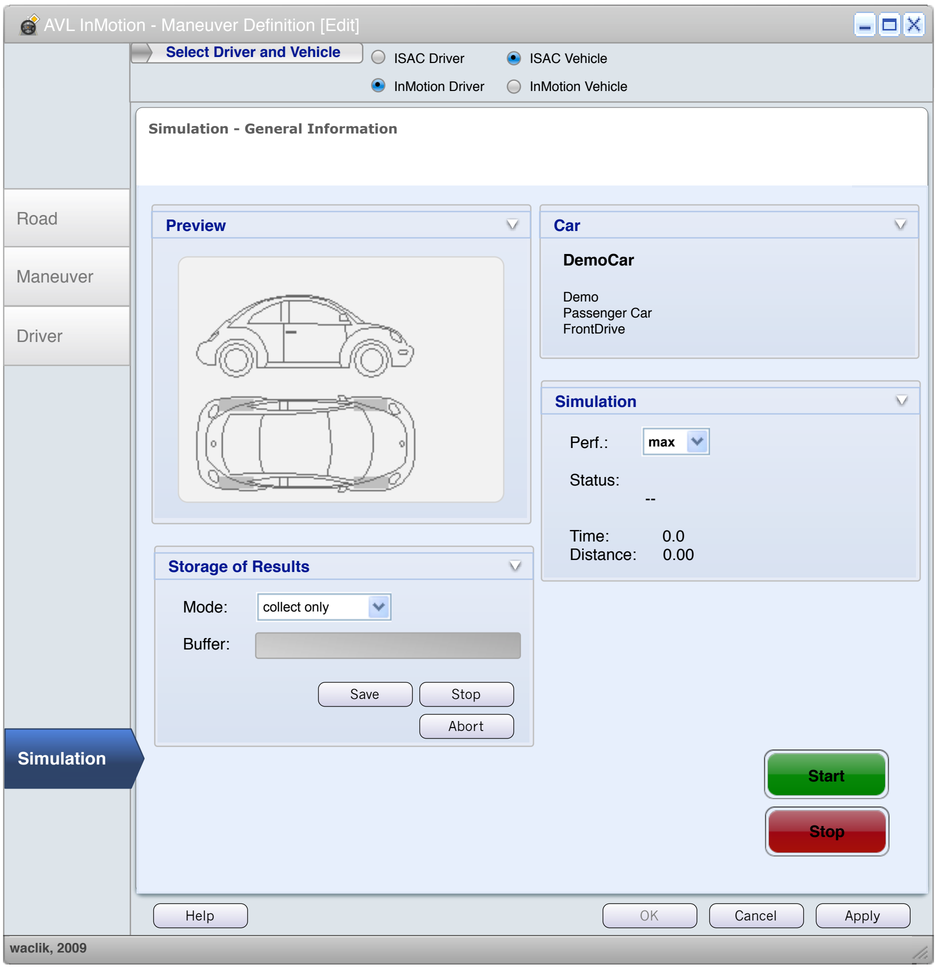


Figure 54: Draft of Simulation Settings

4.3. Integration of the Developed Design

The user interface of AVL InMotion is specifically designed in accordance with the design suggestions presented in Chapter 4.2. Engineers from the third-party company recreated the UI while developers at AVL List performed the integration of the redesigned third-party software component.

However, it is necessary to conduct several iterations until the design fits. Figure 55 shows the UID of AVL Motion after the first iteration of design.

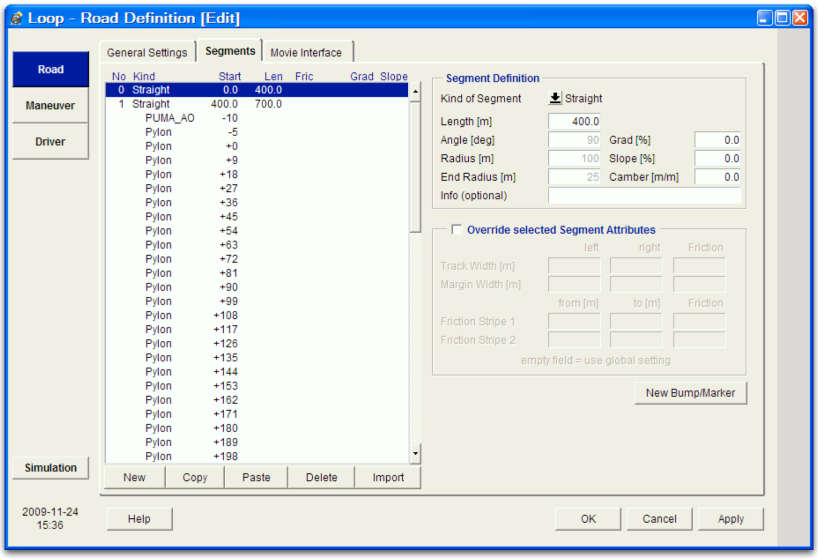


Figure 55: AVL InMotion: First Iteration, Visualization of Road Register

After the second iteration, the following visible changes and main improvements were made:

- The graphical representation is adapted to the layout of AVL products.
- Every register includes a symbol relating to its functionality.
- In *Maneuver* register and tab *Settings* of *Road* register:
  - The buttons of listviews were extended to contain an image.
- The placement of the two registers *Road* and *Maneuver* was changed.

After the redesign, an overview of the latest, partly unfinished build of the software component is presented below. Figure 56 shows the register for the road settings, Figure

57 displays the register for the driver settings and Figure 58 presents the visualization for the maneuver settings.

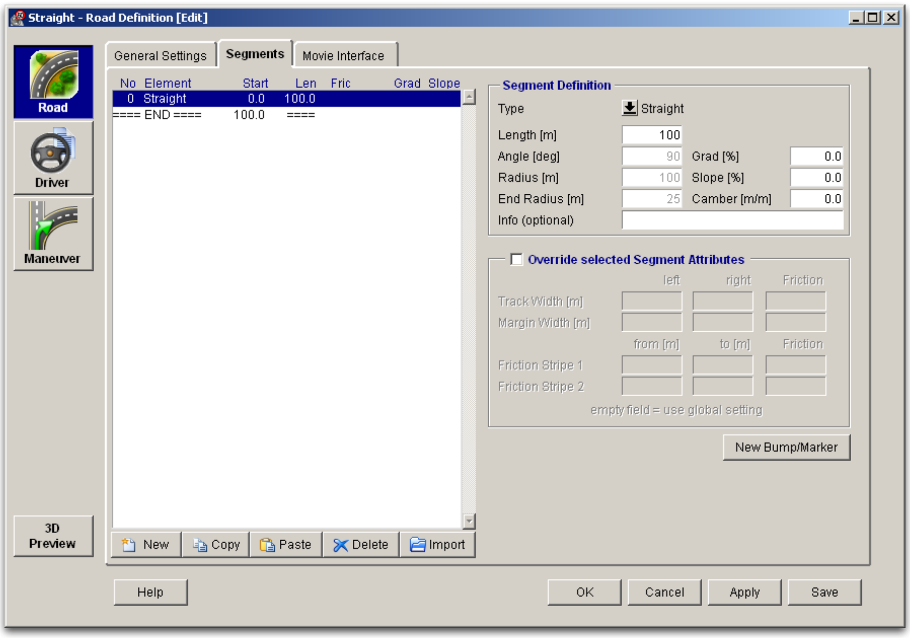


Figure 56: AVL InMotion: Second Iteration, Visualization of Road Register

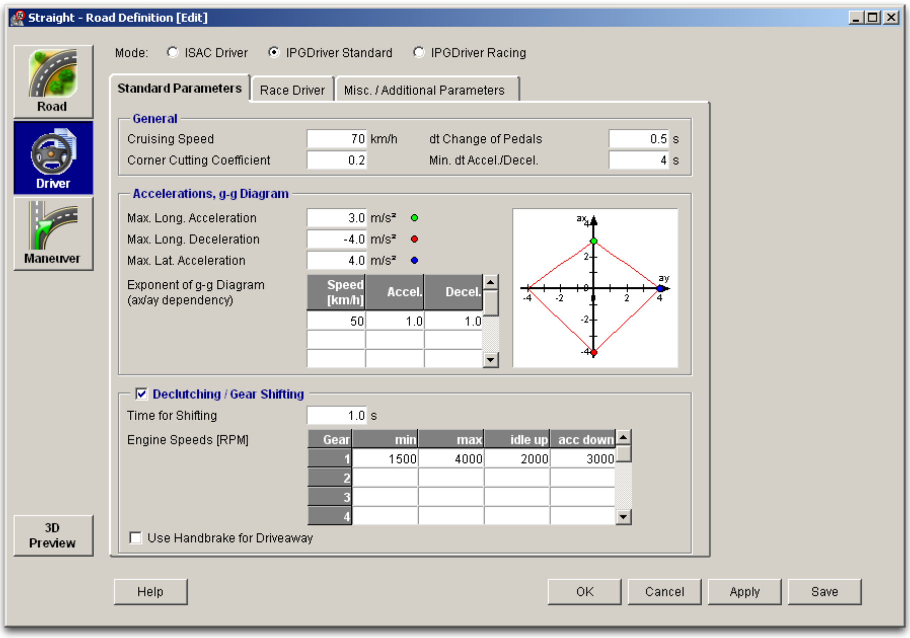


Figure 57: AVL InMotion: Second Iteration, Visualization of Driver Register

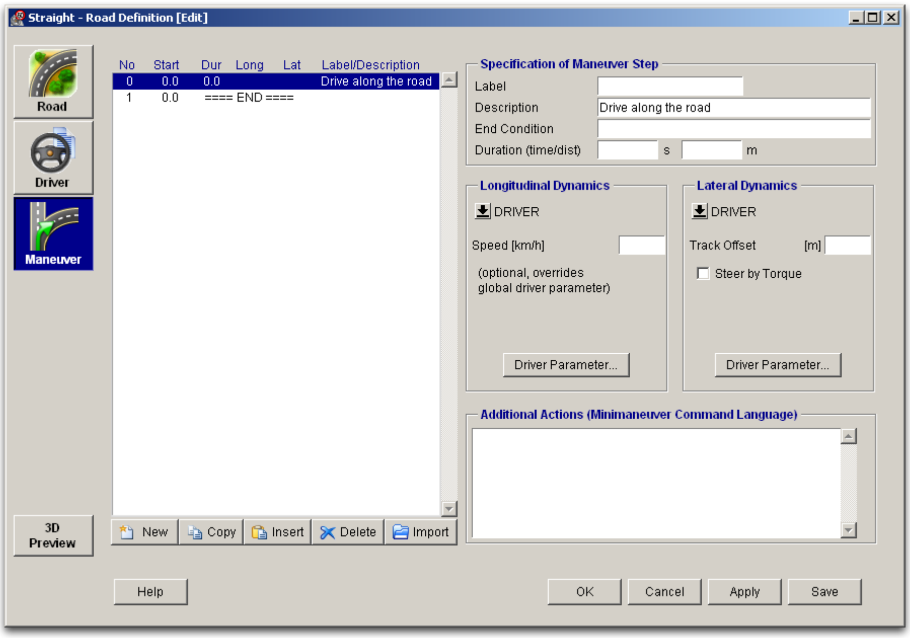


Figure 58: AVL InMotion: Second Iteration, Visualization of Maneuver Register

4.4. Test Specification

4.4.1. Original and Actual Approach

It should be noted that originally the comparable usability test with the developed lo-fi prototype (see Chapter 4.2) of the software component AVL InMotion should be carried out. The goal was the evaluation of the proposed GUI design, the identification of the stakeholders' needs and the collection of the weaknesses as well as strengths of the software component AVL InMotion. Based on the results, AVL InMotion should be developed and contain not only new features, but also improvements. However, once the lo-fi prototype was presented to AVL's stakeholders, the preliminary version of the GUI was implemented immediately (see Chapter 4.3). No test with representative users was taken into account before development.

For this reason, this usability test was conducted with a hi-fi prototype (refer to Chapter 2.2.2.1), which was the latest build of AVL InMotion (see Figure 31).

Figure 59 shows the originally planned approach and actual process concerning AVL InMotion.

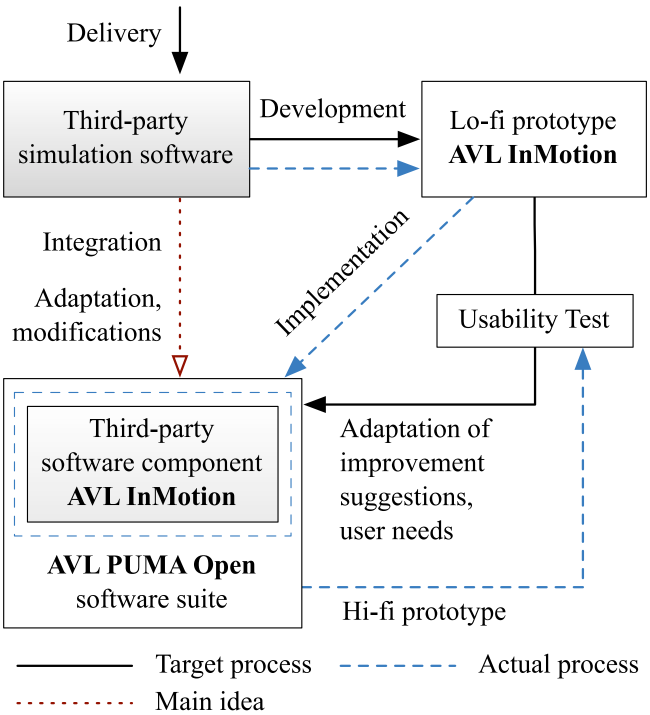


Figure 59: Original Approach versus Actual Process

4.4.2. Aim of the Usability Test

The purpose of this usability test was to gain an evaluation of the standalone program IPG CarMaker and the software component AVL InMotion that is integrated into an existing software suite (refer to Figure 31) regarding the usability and simplicity, but without testing the skills of the test participants. The UI proposals should be reviewed and all findings and resulted improvement suggestions based on them should influence the further development of AVL InMotion and possibly other AVL software products, too.

Additionally, the handling of the test participants with the UID was observed, and at the same time the test users were asked to articulate all their thoughts and actions during the tasks aloud (see Chapter 2.2.3.1).

In the following section, both the test procedure and the practical resources are clearly represented and described in detail, which were required to conduct a correct usability test for this master thesis.

4.4.3. Experimental Procedure

To insure routine in executing the usability experiment, a test scenario had to be prepared containing a test plan with all the necessary steps for carrying out the test procedure.

In the following paragraph, this is explained more in detail and, an overview of the identified phases of the usability test is clearly represented in Figure 60.

First of all, the usability test started with a friendly welcome speech in order to thank the participants for having sufficient time and patience. In the orientation discourse, it was also important to highlight that the test focus concentrated on the system, and not on the performance of the user. The explanation of the test procedure followed and also the videotaping was mentioned. For a more detailed orientation speech, please refer to Appendix A.1.

A standardized questionnaire (see Appendix A.2) had to be completed in order to acquire the necessary background knowledge for the further course of the analysis. Moreover, all candidates were required to sign a consent form (see Appendix A.3) in order to get their authorization for videotaping of their test sessions and also to use these recordings for analyses and research purposes.

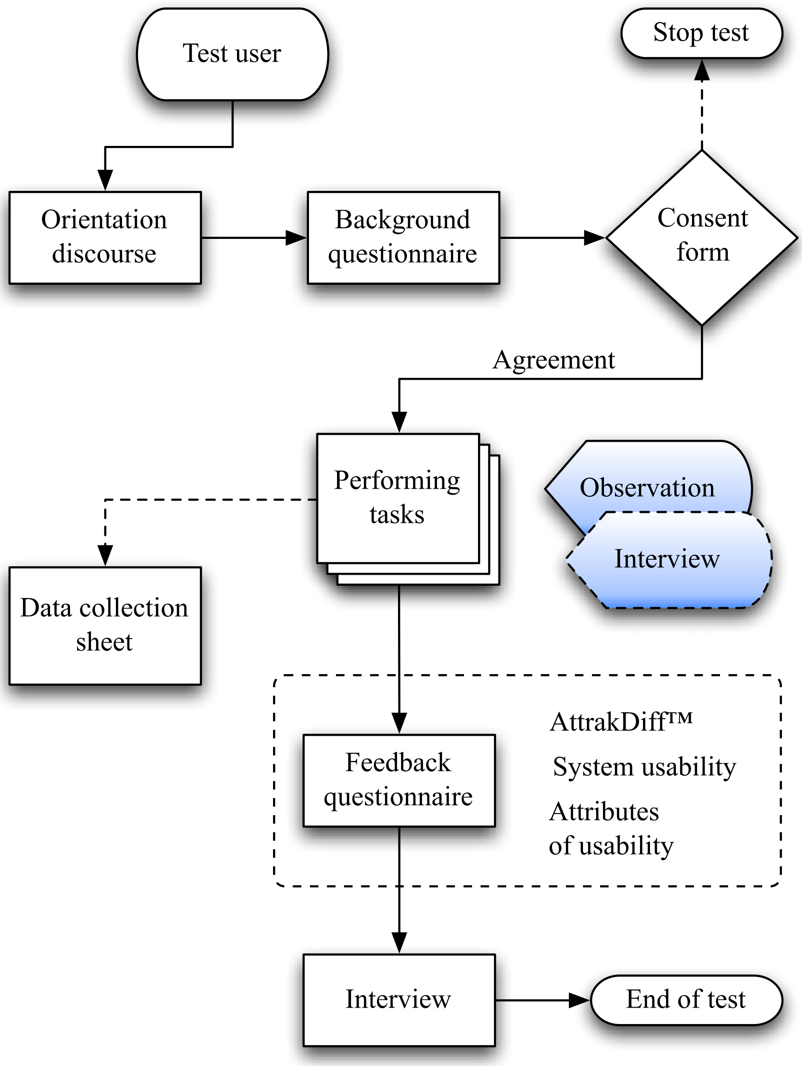


Figure 60: Stages of the Usability Test

After completing all the formalities (including approval from the test users), the main part of the actual usability test could be started.

In this context, nineteen realistic tasks scenarios (see Chapter 4.4.5) were prepared for testing both applications. The tasks were designed to be as simple as possible, so that the test subjects could solve the tasks in a feasible period of time. The users were not given the entire list of tasks at once; they got only the task description (see Chapter 4.4.5.2) and one instruction each on a separate piece of paper at a certain time (see Figure 61 for an example of such task description). The test users were asked to complete each task as best possible, and concurrently, the test participants were requested to *think-aloud* (see

Paragraph 2.2.3.1) while they solved the tasks.

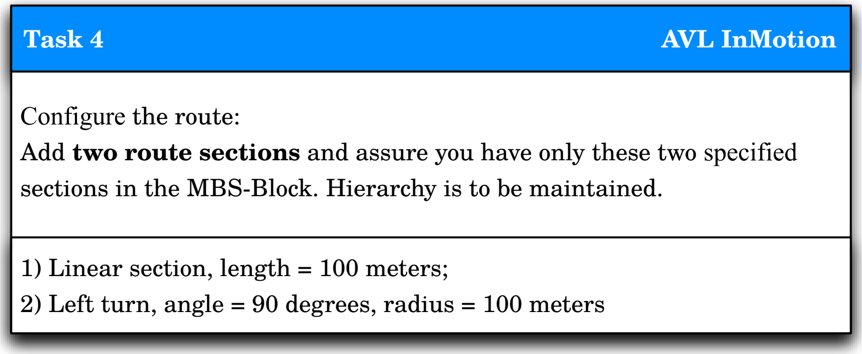


Figure 61: Example of a Task Card given to the User

During the test session, any questions, reactions and other findings made by the test candidates were listed in a data collection sheet (see Appendix A.4) in order to generate a benefit for analyses about usability afterwards. To make an analysis of task completion, the duration each test user needed for a task was also stopped.

After finishing the main usability test part, the participants had to fill out several post-experience questionnaires relating to the system usability using a scale of 1 to 5 (see Appendix A.5.1), the attributes of usability based also on a scale from 1 to 5 (see Appendix A.5.2) and they had to evaluate the application of its hedonic and pragmatic quality by means of AttrakDiff™ questionnaire (refer to Chapter 2.1.2.3 and Appendix A.5.3).

At the end, the test users were interviewed regarding their impressions of the user interface. It is worth mentioning that each test session lasted about two hours.

4.4.4. Test Participants

Experimental Design

First, it should be noted that a *randomized control group allocation without pre-test* (Holzinger, 2010) was used within the experimental plan in this usability study. *"By use of random allocation of the test subjects to the groups [...], almost all interference factors are controlled, with the exception of socially based sources of error"* (Holzinger, 2010, p. 64).

First of all, the test subjects were randomly divided into two test groups:

- Users who performed the usability test for the new and integrated software component AVL InMotion (User Group **A**).
- Users who tested the external third-party software (User Group **B**).

Due to the fact that the software component AVL InMotion was in the development stage at that time, the representative test users were AVL employees who had no experience of using either the integrated software component AVL InMotion or the standalone third-party application. Some of the test users were even software developers of the previously defined business software suite AVL PUMA Open and they were purposely selected in order to be able to valuate the usage of their developed software in particular.

It should be emphasized that this combination of the selected test participants and their know-how perfectly matched the ideal user profile for this test.

Additionally, a pilot test was conducted in order to check the course of the test, for example, whether or not the tasks were well-formulated and comprehensible.

During the test phase, it turned out that five test subjects for each test group were sufficient to gain meaningful results. Questionnaires were also planned, but in gathering reasonable usability recommendations five THA tests in each test group provided meaningful results. For this reason, no further questionnaires or inquiries were conducted.

Table 4 and Table 5 represent the test users by their user group as well as alias numbers, which were used in the result report. The respective information is formulated from the collected background information of the executed usability test.

Test User Group A

Test User	**T1**	**T3**	**T4**	**T6**	**T10**
Date of Test	May 10th, 2010	May 10th, 2010	May 10th, 2010	May 11th, 2010	May 12th, 2010
Time of Test	08:30-10:30	13:00-15:00	15:00-17:00	10:00-12:00	13:00-15:00
General Information					
Gender	Male	Male	Male	Male	Male
Age	50 - 59 years	40 - 49 years	30 - 39 years	30 - 39 years	30 - 39 years
Education	University Degree, Computer Science/ Mathematics	University Degree, Computer Science	General Qualification for University Entrance	General Qualification for University Entrance	General Qualification for University Entrance
Foreign Language Competence	English	English	English	English	English
Experience with Computers					
Using a Computer for	More than 6 years	More than 6 years	More than 6 years	More than 6 years	More than 6 years
Using a Computer on average per day	More than 6 hours	More than 6 hours	2-6 hours	More than 6 hours	More than 6 hours
Experience Level of Computational Skills	Advanced User	Professional	Advanced User	Professional	Professional
Used Operating System	Microsoft Windows	Microsoft Windows	Microsoft Windows, Apple Macintosh	Microsoft Windows	Microsoft Windows

Participation in a Usability Study before (As a Test User)	Yes	Yes	No	No	No
Experience with AVL Product ***AVL PUMA Open***					
Use of the Product	Yes	Yes	Yes	Yes	Yes
Use of Subcomponent PUMA online	Yes	Yes	Yes	Yes	Yes
Using the product	More than 6 Years	More than 6 Years	More than 6 Years	More than 6 Years	1 - 3 years
Experience Level of Using the Product	Advanced User	Professional	Professional	Advanced User	Advanced User
Used Support Materials	Online Documentation	Online Documentation	User Manual, Online Documentation	Reviews	Product Description, User Manual, Online Documentation
Language Setting	English	English	English	English	English

Table 4: Test User Group A

Test User Group B

Test User	**T2**	**T5**	**T7**	**T8**	**T9**
Date of Test	May 10th, 2010	May 11th, 2010	May 11th, 2010	May 11th, 2010	May 12th, 2010
Time of Test	10:30-12:30	08:00-10:00	13:00-15:00	15:00-17:00	09:00-11:00
General Information					
Gender	Male	Male	Male	Male	Male
Age	20 - 29 years	20 - 29 years	20 - 29 years	20 - 29 years	20 - 29 years
Education	University Degree, Telematics	University Degree, Software Engineering and Management	University Degree, Telematics	University Degree, Software Engineering	University Degree, Sports-Equipment Technology
Foreign Language Competence	English	English	English, Italian	English, French	English
Experience with Computers					
Using a Computer for	More than 6 years	More than 6 years	More than 6 years	More than 6 years	More than 6 years
Using a Computer on average per day	2-6 hours	More than 6 hours	More than 6 hours	More than 6 hours	More than 6 hours
Experience Level of Computational Skills	Professional	Professional	Amateur	Professional	Advanced User
Used Operating System	Microsoft Windows	Microsoft Windows, Apple Macintosh	Microsoft Windows	Microsoft Windows	Microsoft Windows

Participation in a Usability Study before (As a Test User)	Yes	Yes	Yes	No	No
Experience with AVL Product ***AVL PUMA Open***					
Use of the Product	No	No	Yes	No	Yes
Use of Subcomponent PUMA online	-	-	Yes	-	Yes
Using the product	-	-	1 - 3 years	-	3 - 6 years
Experience Level of Using the Product	-	-	Amateur	-	Advanced User
Used Support Materials	-	-	Online Documentation	-	Online Documentation, Colleagues
Language Setting	-	-	English, German	-	English

Table 5: Test User Group B

4.4.5. Test Tasks

To begin with, the internal task lists are presented, which contain the description, prerequisites, completion criteria and average processing time as well as the possible solution path for the particular task of each user group. The second part only lists the instructions for each user group, which are given to the users during the test.

4.4.5.1. Internal Task Lists

Various internal tasks are listed below for testing both the integrated software component (see Table 6) and the standalone third-party application (see Table 7). The main focus regarding the creation of the tasks was the identification of the main and critical points that had to be tested. The tasks were identified through a detailed and precise study of the third-party software component AVL InMotion and the external third-party application. The modeling of tasks had to be as realistic as possible, the instructions had to be clear, simple and understandable for the potential test users.

The Internal Task List for Group A

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
1	Launch AVL Explorer via AVL PUMA Open Application Desktop and add a new Test. [Rename it into MBS_Test _X].	AVL PUMA Open Application Desktop is open.	New test named MBS_Test_X was added as specified.	1 min	Click **AVL Explorer** on **AVL PUMA Open Application Desktop** to open application. Navigate to treeview Local→Projects→ AVL→Projects Data and right click to access a contextual menu. Follow select menu New | Test and rename this test into **MBS_Test_X** afterwards.
2	Open the previously added test MBS_Test _X in order to create a new MBS-Block. [Type: Straight].	AVL Explorer is open and previous added test is available.	In opened test editing window in tab Library a new Straight (MBS-Block) was added.	2 mins	Double click on previous added test **MBS_Test_X** (or open the contextual menu by right-clicking and select Edit) to open the test-editing window. Navigate to tab **Library** and open the contextual menu by right clicking on it in order to create a new MBS-Block of type **Straight**. Select contextual menu New | Road and Maneuvers | Straight.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
3	Open MBS-Editor and spend a few minutes looking around the application.	Predefined Test is open and a MBS-Block was added before.	MBS-Editor was opened and the users indicate they have finished looking around, or 3 minutes have elapsed.	3 mins	After 3 minutes, ask the user: 1. What does this editor represent? 2. Who is MBS-Editor intended for? 3. What does this MBS-Editor offer or, better, what are you able to do?
4	Configure the route: Add two route sections and assure you have only these two specified sections in the MBS-Block. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 100 meters; 2) Left turn, angle = 90 degrees, radius = 100 meters]	Predefined test is open, an MBS-Block was added and is opened.	Both segments as per instruction folder were added to road configuration and were displayed in segment view.	3 mins	Select the **Road** register on the left hand side in MBS-Editor and navigate to tab **Segments**. The first segment of type Straight is already predefined; there is nothing more to do. To add the second segment (after segment one!) select the last entry of segment definition view and click the **New** button. In the field **Segment Definition** select the required parameter **Type** and modify the settings as per instruction (Modification of parameters **Angle [deg]** and **Radius [m]** are necessary).

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
5	Verify the current settings of route via a three-dimensional view.	Predefined test is open, an MBS-Block was added and is opened. Two road segments are added into MBS-Editor.	No error message appeared by performing the preview simulation. The MOVIE window was opened and the 3D preview of route was displayed as per instruction folder.	2 mins	Press the **3DPreview** button to open the MOVIE window. Click the **green arrow** button to start 3D preview simulation. (Other possibility to open the preview window is to right click in segment view to access a contextual menu in order to select 3DPreview.
6	Add several route sections to MBS-Block as defined in instruction folder. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 50 meters, grade = 10 percent; 2) Left turn, angle = 180 degrees, radius = 40 meters; 3) Linear section, length = 50 meters, grade = -10 percent] 4) Linear section, length = 30 meters]	Predefined test is open, an MBS-Block was added and is opened. Two road segments are added into MBS-Editor.	Two straights and one turn left - both modified with required settings - were added to existing MBS-Block.	6 mins	Select the **Road** register on the left hand side in MBS-Editor and navigate to tab **Segments**. To place the required segment to right position select the last entry of segments and click the **New** button. In the field **Segment Definition** select the required parameter **Type** and modify the settings as per instruction. Repeat previous action as long as all required segments are added to road. (Modification of parameters **Length [m], Angle [deg]**, **Radius [m]** and **Grade [%]** are necessary).

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
7	Validate the route from "above" if the road has been created as defined in instruction folder.	Predefined test is open, an MBS-Block was added and is opened. Five road segments are added into MBS-Editor.	The *Bird's Eye View* was opened and it has displayed the actual route as defined in instruction folder.	3 mins	Select the **Road** register on the left side in MBS-Editor and navigate to tab **Segments**. To activate the *Bird's Eye View* open the contextual menu with right-click and select Bird's Eye View of Road) to open required view.
8	Place a billboard 50 meters after starting point of route and move the label to the right hand side of road.	Predefined test is open, an MBS-Block was added and is opened. Five road segments are added into MBS-Editor.	The sign plate was placed to the right hand side and 50 meters after starting point.	4 mins	Select the **Road** register on the left hand side in MBS-Editor and navigate to tab **Segments**. To place the required sign plate to correct position select the first segment and click **New Bump/Marker** button. In the field **Bump/ Marker** select the required **Sign Plate** from drop-down box of parameter **Type**. Set also the parameters **Start Offset [m]** and **y Offset [m]** to get the sign plate in the correct position.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
9	Add a stand still five meters before the road downhill ends and make sure that the car is waiting for 3 seconds.	Predefined test is open, an MBS-Block was added and is opened. Six road segments are added into MBS-Editor.	The required stop was added to road.	4 mins	Select the **Road** register on the left hand side in MBS-Editor and navigate to tab **Segments**. To place the required stop to correct position, select the road segment where the vehicle coasts and then click **New Bump/Marker** button. In the field **Bump/ Marker** select the required **Stop** from drop-down box of parameter **Type**. Set also the parameters **Start Offset [m]** and **Duration [s]** to get the stop on right position as well as to define the time of waiting.
10	Apply modifications and close the MBS Editor. Add the MBS-Block to the test run on the right hand side. Afterwards, an engine command has to be added to finally start this test run. Last, close, save and generate the modified test.	Predefined test is open, an MBS-Block was added and is opened. Several road segments are added into MBS-Editor.	MBS-Block and a command to start the engine were added to test run. The modifications were saved and the test was generated.	1 min	Click the **OK** button to close the MBS-Editor. Navigate to tab **Library** and via drag & drop add the previous created and modified MBS-Block to test run on the right window side between **MBS_Test_X** and **Interrupt Testrun**. Switch to tab **Toolbox**. Select UUT Control | Engine Command to add a start command to test run before MBS-Block on the right side. Close window and click **Save & Generate** button.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
11	Launch PUMA via AVL PUMA Open Application Desktop and open test parameter MBS_Test_X. In AUTOMATIC mode, run the previously created test (in combination with standard PUMA parameters) and validate the vehicle performance by simulation movie.	AVL PUMA Open Application Desktop is open and MBS_Test_X was created.	PUMA was opened, the specified test MBS_Test_X was selected and state AUTOMATIC was reached. The MOVIE window was opened and the user had started the simulation movie to validate vehicle performance.	6 mins	Click **Start PUMA** on **AVL PUMA Open Application Desktop** to open PUMA and wait till state MONITOR is reached. In field **Test** click **Open file** button in order to select test parameter **MBS_Text_X** in **Parameter-Dialog**. Afterwards, click **Reload** button and go to state MANUAL. In **Vehicle Status** dialog, select **Start Engine** button and click **AUTOMATIC** button in PUMA. Open the MOVIE to monitor the maneuver-based testing of vehicle by clicking **Start Movie** button in **Vehicle Status** dialog.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
12	Modify test MBS_Test_X again and add several route sections to road definition in order to reproduce the road as defined in instruction folder. [Specification for this task: see Figure 62]	Predefined test is open, an MBS-Block was added and is opened. Six road segments are added into MBS-Editor.	A right hand bend, a straight and Tree Strips were added to existing MBS-Editor, thus the route was specified as per instruction folder.	4 mins	Open MBS-Editor as described in Tasks 2 and 3, select **Road** register and navigate to tab **Segments**. To add the right hand bend select the last entry of segments and click **New** button. In the field **Segment Definition** select the required parameter **Type** (**Turn right**) and modify the settings [Angle = 90 degrees, radius = 20 meters]. To add a straight, select the last entry of segments and click **New** button. In the field **Segment Definition**, select the required parameter **Type** (**Straight**) and modify the settings [Length = 500 meters]. To add two trees vis-à-vis towards the end of route, select the previously created straight segment and then click **New Bump/Marker** button. In the field **Bump/ Marker,** select the required **Tree Strip(s)** from drop-down box of parameter **Type**. Set also the parameters **Start Offset [m]** to 450 meters and select the option **both** of parameter **Road side**.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
13	Change the length of the first route section to 500 meters and set the cruising speed to 100 km/h for the first 500 meters.	Predefined test is open, an MBS-Block was added and is opened. Eight road segments are added into MBS-Editor.	The first road segment was set to 500 meters and a new maneuver was defined, which defined that the vehicle had a cruising speed of 100 km/h for the first 500 meters.	5 mins	Select the first segment of road definition and set the parameter **Length** to 500 meters in field **Segment Definition**. Select the **Maneuver** register on the left hand side in MBS-Editor. Then select the first, existing maneuver (No. 0) and click **New** button in order to add a new maneuver before existing maneuver. In field **Specification of Maneuver Step** set the parameter of **Duration (time/dis)** for distance to 500 meters. Further, in field **Longitudinal Dynamics** select **DRIVER** from drop-down box and set parameter **Speed [km/h]** to 100.
14	Where can you monitor the actual velocity of test run when executing the three-dimensional simulation?	Main window is enabled and visible.	Menu or rather the window for monitoring the velocity was found.	2 mins	**PUMA**: in **Vehicle Status** dialog.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
15	Set the cruising speed to 130 km/h for approximately the last 500 meters.	Predefined test is open, an MBS-Block was added and is opened. Eight road segments are added into MBS-Editor and a maneuver was also added.	The cruising speed of driver was set to 130 km/h for the last ~500 meters by adding new maneuvers to MBS-Editor.	6 mins	(Assuming, that the length of route is approximately 1400 meters, which can be found in road register in tab segments.) Select the **Maneuver** register on the left hand side in MBS-Editor. Then select the existing, added by default maneuver (No. 1), and click **New** button in order to add a new maneuver before existing maneuver. In field **Specification of Maneuver Step** set the parameter of **Duration (time/dist)** for distance up to 400 meters. Next step is to add a new maneuver afterwards previous created maneuver in order to set the distance to 500 meters. Further, in field **Longitudinal Dynamics** select **DRIVER** from drop-down box and set parameter **Speed [km/h]** to 130.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
16	Modify the settings so that the vehicle drives in the right hand lane. Configure the road in such a way that the vehicle drives in a two-meter lane for the first 500 meters.	Predefined test is open, an MBS-Block was added and is opened. Eight road segments are added into MBS-Editor.	The driving lane of vehicle was set to right hand side and also the track width was overridden to two meters for the first 500 meters.	5 mins	Select **Road** register and navigate to tab **General Settings**. In field **General Settings** set the parameter **Driving Lane** to **right.** Afterwards navigate to tab **Segments** and select the first segment to override settings. Set checkbox **Override selected Segment Attributes** in order to configure the parameter **track width** for the left and right hand side as specified in instruction.
17	Please modify the ground speed to 80 km/h.	Predefined test is open, an MBS-Block was added and is opened.	Cruising speed was set to 80 km/h.	2 mins	Select the **Driver** register on the left hand side in MBS-Editor and navigate to tab **Standard Parameters**. In the field **General**, enter the required velocity of Parameter **Cruising Speed**.
18	Where would you make modifications to have a Bird's Eye View in three-dimensional view?	Main window is enabled and visible.	Menu was found for doing such modifica-tion.	2 mins	Press the **3DPreview** button to open the **MOVIE** window. Select Camera | Bird's Eye View to make certain settings.
19	Where can you find the instruction to delete all route sections at once?	Main window is enabled and visible.	Menu was found for performing such a command.	2 mins	Select **Road** register and navigate to tab **General Settings**. Then right click to access a contextual menu in order to select Delete all Segments.

Table 6: The Internal Task List for Group A

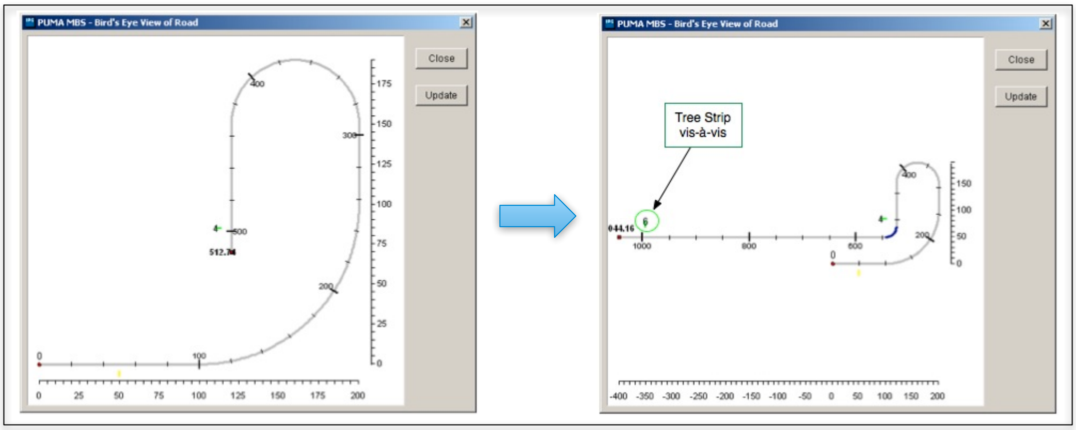


Figure 62: Recreation of the Bird's Eye View of Road

The Internal Task List for Group B

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
1	Launch IPG CarMaker and spend a few minutes looking around the program. [Menu, User Interface]	-	IPG CarMaker was opened and the users indicate they have finished looking around, or three minutes have elapsed.	3 mins	After 3 minutes, ask the user: 1. What does this application represent? 2. Who is IPG CarMaker intended for? 3. What does IPG CarMaker offer or, better, what are you able to do?
2	Create a new project folder. [Specification for this task: project folder name = TestuserX, project path: D:\MBS-UsabilityTest\]	IPG CarMaker is open.	Project Folder was created and it was displayed in caption of IPG CarMaker.	2 mins	Select menu File | Project Folder | Create/update project... . Afterwards, create a new project folder. For example, D:/MBS-UsabilityTest/ TestuserX.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
3	Add tires to the test run. [Specification for this task: Model of tires = RT_185 _55R15]	Predefined project is opened. Main window is enabled and visible.	In the main window of test run, the tires were registered.	2 mins	Click the **Select** button near section *Tires* on the main window and select the required tires in data/tire dialog.
4	Configure the route: Add two route sections and assure you have only these two specified sections in the road definition. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 100 meters; 2) Left turn, angle = 90 degrees, radius = 100 meters]	Predefined project is opened. Main window is enabled and visible.	Both segments as per instruction folder were added to road configuration and were displayed in segment view.	3 mins	Select menu Parameter | Road. Navigate to tab *Segments* and add a new road segment via **New** button. In the field **Segment Definition** select the required parameter **Type** (Straight) and modify the settings as per instruction (Modification of parameters **Length [m]**). To add the second segment (after segment one!) select the last entry of segment definition view and click the **New** button. In the field **Segment Definition**, select the required parameter **Type** (Left turn) and modify the settings as per instruction (Modification of parameters **Angle [deg]** and **Radius [m]** are necessary). Close the dialog again.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
5	Modify the project in order to be able to drive *along the street*.	Predefined project is opened. Main window is enabled and visible.	Maneuver was added to project and it was displayed in maneuver-section, including a description.	6 mins	Select menu Parameter | Maneuver (or simply click in maneuver-section on the main window) in order to open the maneuver dialog to define a new maneuver. Click the **New** button in order to add a new maneuver. In field Specification of Maneuver Step set the parameter of Description as per instruction folder. Afterwards click the **Close** button.
6	Verify the current settings of route via a three-dimensional view.	Main window is enabled and visible. Predefined test run is opened, a maneuver and two road segments are added, and the tires are selected.	No error message appeared by performing the preview simulation. The IPG Movie window was opened and the 3D simulation of route was displayed as per instruction folder.	2 mins	Open the IPG Movie dialog via menu File | IPG-Movie. (Selection of real time in performance-drop-down box in main window in the block simulation to assure to simulate in real-time.) To execute the test run, click the green **Start** button in the main window.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
7	Add further route sections contiguously to road as defined in instruction folder. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 50 meters, grade = 10 %; 2) Left turn, angle = 180 degrees, radius = 40 meters; 3) Linear section, length = 50 meters, grade = -10 %; 4) Linear section, length = 30 meters]	Main window is enabled and visible. Predefined test run is open, a maneuver and two road segments are added.	Two straights and one turn left, both are modified with required settings, were added to existing test run.	6 mins	Select menu Parameter | Road and navigate to tab **Segments**. To place the required segment to correct position, select the last entry of segments and click the **New** button. In the field **Segment Definition**, select the required parameter **Type** and modify the settings as per instruction. Repeat previous action as long as all required segments are added to road. (Modification of parameters **Length [m], Angle [deg]**, **Radius [m]** and **Grade [%]** are necessary).

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
8	Validate the route from “above” if the road has been created as defined in instruction folder.	Main window is enabled and visible. Predefined test run is opened and six road segments are added.	The *Bird’s Eye View* was opened and it has displayed the actual route as defined in instruction folder.	3 mins	Select menu Parameter | Road and navigate to tab **Segments**. To activate the *Bird’s Eye View* open the contextual menu with right-click and select Bird’s Eye View of Road) to open required view.
9	Place a billboard 50 meters after starting point of route and move the label to the right hand side of road.	Main window is enabled and visible. Predefined test run is open and six road segments are added.	The sign plate was placed to the right hand side and 50 meters after starting point.	4 mins	Select menu Parameter | Road and navigate to tab **Segments**. To place the required sign plate to correct position, select the first segment and click **New Bump/Marker** button. In the field **Bump/ Marker**, select the required **Sign Plate** from drop-down box of parameter **Type**. Set also the parameters **Start Offset [m]** and **y Offset [m]** to get the sign plate in the correct position.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
10	Add a stand still five meters before the road downhill ends and make sure that the car is waiting for 3 seconds.	Main window is enabled and visible. Predefined test run is open and six road segments are added.	The required stop was added to road.	4 mins	Select menu Parameter | Road and navigate to tab **Segments**. To place the required stop to correct position select the road segment where the vehicle coasts and then click the **New Bump/Marker** button. In the field **Bump/ Marker**, select the required **Stop** from drop-down box of parameter **Type**. Set also the parameters **Start Offset [m]** and **Duration [s]** to get the stop on correct position as well as to define the time of waiting.
11	Run the previously created test run via the three-dimensional simulation, and validate the modified settings and vehicle performance by simulation movie.	Main window is enabled and visible. Predefined test run is open, a maneuver and six road segments are added, and tires are selected.	The IPG Movie window was opened and the users had started the simulation movie to validate vehicle performance.	3 mins	Open the IPG Movie window via menu File | IPG-Movie. (Selection of real time in performance-drop-down box in main window in the block simulation to assure to simulate in real-time.) To execute the test run, click the green **Start** button in main window.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
12	Modify test run and add several route sections to road definition in order to reproduce the route as defined in instruction folder. [Specification for this task: see Figure 62]	Main window is enabled and visible. Predefined test run is open, a maneuver and six road segments are added, and tires are selected.	A right hand bend, a straight and Tree Strips were added to existing test run, thus the route was specified as per instruction folder.	4 mins	Select menu Parameter | Road and navigate to tab **Segments**. To add the right hand bend select the last entry of segments and click **New** button. In the field **Segment Definition** select the required parameter **Type** (**Turn right**) and modify the settings [Angle = 90 degrees, radius = 20 meters]. To add a straight, select the last entry of segments and click **New** button. In the field **Segment Definition**, select the required parameter **Type** (**Straight**) and modify the settings [Length = 500 meters]. To add two trees vis-à-vis towards the end of route select the previously created straight segment and then click **New Bump/Marker** button. In the field **Bump/ Marker**, select the required **Tree Strip(s)** from drop-down box of parameter **Type**. Set also the parameters **Start Offset [m]** to 450 meters and select the option **both** of parameter **Road side**. Verify the settings Bird's Eye View.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
13	Change the length of the first route section to 500 meters and set the cruising speed to 100 km/h for the first 500 meters.	Main window is enabled and visible. Predefined test run is open, a maneuver and eight road segments are added.	The first road segment was set to 500 meters and a new maneuver was defined, which defined that the vehicle had a cruising speed of 100 km/h for the first 500 meters.	5 mins	Select menu Parameter | Road and navigate to tab **Segments**. Select the first segment of road definition and set the parameter **Length** to 500 meters in field **Segment Definition**. Select menu Parameter | Maneuver (or simply click in maneuver-section on main window) in order to open the maneuver dialog to define a new maneuver. Then select the first, existing maneuver (No. 0) and click **New** button in order to add a new maneuver before existing maneuver. In field **Specification of Maneuver Step**, set the parameter of **Duration (time/dist)** for distance to 500 meters. Further, in field **Longitudinal Dynamics** select **DRIVER** from drop-down box and set parameter **Speed [km/h]** to 100.
14	Where can you monitor the actual velocity of test run when executing the three-dimensional simulation?	Main window is enabled and visible.	Menu or rather the window for monitoring the velocity was found.	2 mins	Open the Instruments window via menu File | Instruments.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
15	Set the cruising speed to 130 km/h for approximately the last 500 meters.	Main window is enabled and visible. Predefined test run is open, two maneuvers and eight road segments are added.	The cruising speed of driver was set to 130 km/h for the last ~500 meters by adding new maneuvers to list.	6 mins	(Assuming, that the length of route is ~1400 meters, which can be found in road dialog in tab segments.) Select menu Parameter | Maneuver (or simply click in maneuver-section on main window) in order to open the maneuver dialog to define a new maneuver. Then select the second and existing maneuver (No. 1), and click **New** button in order to add a new maneuver before that maneuver. In field **Specification of Maneuver Step** set the parameter of **Duration (time/dist)** for distance up to ~400 meters. Next step is to add a new maneuver afterwards previous created maneuver in order to set the distance to ~500 meters. Further, in field **Longitudinal Dynamics** select **DRIVER** from drop-down box and set parameter **Speed [km/h]** to 130.

Task No.	Description	Pre-requisites	Completion Criteria	Max. Time	Possible Solution Path
16	Modify the settings so that the vehicle drives in the right hand lane. Configure the road in such a way that the vehicle drives in a two-meter lane for the first 500 meters.	Main window is enabled and visible. Predefined test run is open and eight road segments are added.	The driving lane of vehicle was set to right hand side and also the track width was overridden to two meters for the first 500 meters.	6 mins	Select menu Parameter | Road and navigate to tab **General Settings**. In field **General Settings** set the parameter **Driving Lane** to **right.** Afterwards navigate to tab **Segments** and select the first segment to override settings. Set checkbox **Override selected Segment Attributes** in order to configure the parameter **track width** for the left and right hand side as specified in instruction.
17	Please modify the ground speed to 80 km/h.	Main window is enabled and visible. Predefined test run is opened.	Cruising speed was set to 80 km/h.	2 mins	Select menu Parameter | Driver and navigate to tab **Standard Parameters**. In the field **General**, enter the required velocity of Parameter **Cruising Speed**.
18	Where would you make modifications to have a Bird's Eye View in three-dimensional view?	Main window is enabled and visible.	Menu for doing such modification was found.	2 mins	Open the IPG Movie dialog via menu File | IPG-Movie. Select Camera | Bird's Eye View to make certain settings.
19	Where can you find the instruction to delete all route sections at once?	Main window is enabled and visible.	Menu was found for performing such a command.	2 mins	Select menu Parameter | Road and navigate to tab **Segments**. Then right click to access a contextual menu in order to select Delete all Segments.

Table 7: The Internal Task List for Group B

4.4.5.2. Task Descriptions

Tables 9 and 8 list the tasks descriptions, which were given to the test users by separate task cards (see Figure 61). The tasks contain only the test number and their description.

Descriptions of the Tasks - Group A

Task No.	Description
T1	Launch AVL Explorer via AVL PUMA Open Application Desktop and add a new Test. [Rename it into MBS_Test _X].
T2	Open the previously added test MBS_Test _X in order to create a new MBS-Block. [Type: Straight].
T3	Open MBS-Editor and spend a few minutes looking around.
T4	Configure the route: Add two route sections and assure you have only these two specified sections in the MBS-Block. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 100 meters; 2) Left turn, angle = 90 degrees, radius = 100 meters]
T5	Verify the current settings of route via a three-dimensional view.
T6	Add several route sections to MBS-Block as defined in instruction folder. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 50 meters, grade = 10 percent; 2) Left turn, angle = 180 degrees, radius = 40 meters; 3) Linear section, length = 50 meters, grade = -10 percent] 4) Linear section, length = 30 meters]
T7	Validate the route from “above” if the road has been created as defined in instruction folder.
T8	Place a billboard 50 meters after starting point of route and move the label to the right hand side of road.
T9	Add a stand still five meters before the road downhill ends and make sure that the car is waiting for 3 seconds.
T10	Apply modifications and close the MBS Editor. Add the MBS-Block to the test run on the right hand side. Afterwards, an engine command has to be added to finally start this test run. Last, close, save and generate the modified test.

Task No.	Description
T11	Launch PUMA via AVL PUMA Open Application Desktop and open test parameter MBS_Test_X. In AUTOMATIC mode, run the previously created test (in combination with standard PUMA parameters) and validate the vehicle performance by simulation movie.
T12	Modify test MBS_Test_X again and add several route sections to road definition in order to reproduce the road as defined in instruction folder. [Specification for this task: see Figure 62]
T13	Change the length of the first route section to 500 meters and set the cruising speed to 100 km/h for the first 500 meters.
T14	Where can you monitor the actual velocity of test run when executing the three-dimensional simulation?
T15	Set the cruising speed to 130 km/h for approximately the last 500 meters.
T16	Modify the settings so that the vehicle drives in the right hand lane. Configure the road in such a way that the vehicle drives in a two-meter lane for the first 500 meters.
T17	Please modify the ground speed to 80 km/h.
T18	Where would you make modifications to have a Bird's Eye View in three-dimensional view?
T19	Where can you find the instruction to delete all route sections at once?

Table 8: Descriptions of the Tasks - Group A

Descriptions of the Tasks - Group B

Task No.	**Description**
T1	Launch IPG CarMaker and spend a few minutes looking around. [Menu, User Interface]
T2	Create a new project folder. [Specification for this task: project folder name = TestuserX, project path: D:\MBS-UsabilityTest\]
T3	Add tires to test run. [Specification for this task: Model of tires = RT_185 _55R15]
T4	Configure the route: Add two route sections and assure you have only these two specified sections in the road definition. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 100 meters; 2) Left turn, angle = 90 degrees, radius = 100 meters]

Task No.	**Description**
T5	Modify the project in order to be able to drive *along the street*.
T6	Verify the current settings of route via a three-dimensional view.
T7	Add further route sections contiguously to road as defined in instruction folder. Hierarchy is to be maintained. [Specification for this task: 1) Linear section, length = 50 meters, grade = 10 %; 2) Left turn, angle = 180 degrees, radius = 40 meters; 3) Linear section, length = 50 meters, grade = -10 %; 4) Linear section, length = 30 meters]
T8	Validate the route from “above” if the road has been created as defined in instruction folder.
T9	Place a billboard 50 meters after starting point of route and move the label to the right hand side of road.
T10	Add a stand still five meters before the road downhill ends and make sure that the car is waiting for 3 seconds.
T11	Run the previously created test run via three-dimensional simulation and validate the modified settings and vehicle performance by simulation movie.
T12	Modify test run and add several route sections to road definition in order to reproduce the route as defined in instruction folder. [Specification for this task: see Figure 62]
T13	Change the length of the first route section to 500 meters and set the cruising speed to 100 km/h for the first 500 meters.
T14	Where can you monitor the actual velocity of test run when executing the three-dimensional simulation?
T15	Set the cruising speed to 130 km/h for approximately the last 500 meters.
T16	Modify the settings so that the vehicle drives in the right hand lane. Configure the road in such a way that the vehicle drives in a two-meter lane for the first 500 meters.
T17	Please modify the ground speed to 80 km/h.
T18	Where would you make modifications to have a Bird’s Eye View in three-dimensional view?
T19	Where can you find the instruction to delete all route sections at once?

Table 9: Descriptions of the Tasks - Group B

4.4.6. Test Equipment

The usability test was conducted in a familiar test environment, or rather in a realistic setting. A quiet room was provided at AVL and a personal computer and a monitor were also made available where all necessary software components were already installed. Next to the monitor for capturing the users' facial expressions, a mirror was placed. Moreover, video and audio recording equipment was also used for testing, which included a digital camcorder (hard disk recorder), a tripod and a microphone. Figures 63, 64 and 65 show the usability lab in order to get an idea of the test settings.

Figure 63: Usability Lab, View 1

Figure 64: Usability Lab, View 2

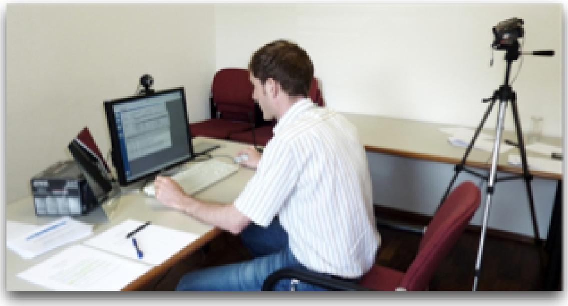


Figure 65: Usability Lab, View 3

4.5. Hypothesis Regarding the Usability of Software

In this section, a hypothesis is described in particular referring to user friendliness. Investigation is necessary to see whether the preexisting software suite with the integration of AVL InMotion, whose design is proposed in this master thesis, is, as expected, user-friendlier than the standalone third-party simulation application. In less technical terms, the extended software solution of AVL should achieve a better result than the externally developed software component:

Hypothesis 1 (H1): The integration of a third-party software component into a larger system improves its usability for users familiar with the existing system.

The comparative analysis of AVL InMotion and the standalone third-party application is possible, because the structure of both software systems according to their main functionalities are quite similar.

Based on the according usability methods, it shall be verified whether this hypothesis is significantly proven, disproven, or if it is neither true nor false considering the existing data (Holzinger, 2010). H1 also relates to the subjective attributes of usability. All associated measurement criteria have already been described in Section 2.1.

5. Results

This chapter includes the evaluation of the results found by the usability test. It provides the preparation of the test tasks of the two tested applications, namely AVL InMotion and the standalone third-party application, followed by the comparison of task completion, categorization of the identified problems (see Chapter 5.2.1) and suggested recommendations (see Chapter 5.2.2). Furthermore, a proposal regarding to a redesigned user interface for AVL InMotion (see Chapter 5.3) as well as the results of the used questionnaires are presented (see Chapter 5.4).

5.1. Task Completion

With regard to the comparison of the task completion, measurements of the performance are described below. These results are based on the observation of the experimenter and are also taken from the data collection sheet, which has been filled out during the test.

5.1.1. Preparation of Relevant Test Tasks

The preparation of the test tasks deals with the summary of the relevant tasks (see Chapter 4.4.5). Table 10 represents a list with all relevant tasks of the two tested applications. Only comparable tasks are practically useful for further analysis in order to ensure comparable results from these tests.

Task No.	Description
T1	Open *CarMaker/ MBS-Editor* and spend a few minutes looking around.
T2	Configure the route: Add two route sections and assure you have only these two specified sections in this *test run*. Hierarchy is to be maintained. [1) Linear section, length=100 meters; 2) Left turn, angle=90 degrees, radius=100 meters]
T3	Verify the current settings of route via a three-dimensional view.
T4	Add several route sections to *test run* as defined in the instruction folder. Hierarchy is to be maintained. [1) Linear section, length=50 meters, grade=10

	percent; 2) Left turn, angle=180 degrees, radius=40 meters; 3) Linear section, length=50 meters, grade=-10 percent] 4) Linear section, length=30 meters]
T5	Validate the route from “above” if the road has been created as specified before.
T6	Place a billboard 50 meters after starting point of route and move the label to the right hand side of road.
T7	Add a stand still five meters before the road downhill ends and make sure that the car is waiting for 3 seconds.
T8	Run the previously created test run via three-dimensional simulation (*online*) and validate the modified settings and vehicle performance by movie.
T9	Modify *test run/ test* again and add several route sections to road definition in order to reproduce the road as defined in instruction folder [see Figure 62].
T10	Change the length of the first route section to 500 meters and set the cruising speed to 100 km/h for the first 500 meters.
T11	Where can you monitor the actual velocity of test run when executing the three-dimensional simulation?
T12	Set the cruising speed to 130 km/h for approximately the last 500 meters.
T13	Modify the settings so that the vehicle drives in the right hand lane. Configure the road in a such way that the vehicle drives in a two-meter lane for the first 500 meters.
T14	Please modify the ground speed to 80 km/h.
T15	Where would you make modifications to have a Bird's Eye View in three-dimensional view?
T16	Where can you find the instruction to delete all route sections at once?

Table 10: Unified Task List

5.1.2. Statistic of the Task Completion

The test participants had to perform typical tasks with the applications. According to their task accomplishment, the results are evaluated in the following section. Note that only the comparable tasks (see Subchapter 5.1.1) are used for these analyses.

The task completion is divided into three completion criteria, namely *Passed (P), Failed (F)* and *with Assistance (A)*.

- P: The user completed the task without help.

- F: Despite assistance, the user could not complete the task.
- A: The user needed support (given by the experimenter) in order to complete the task.

5.1.2.1. Evaluation of AVL InMotion (Group A)

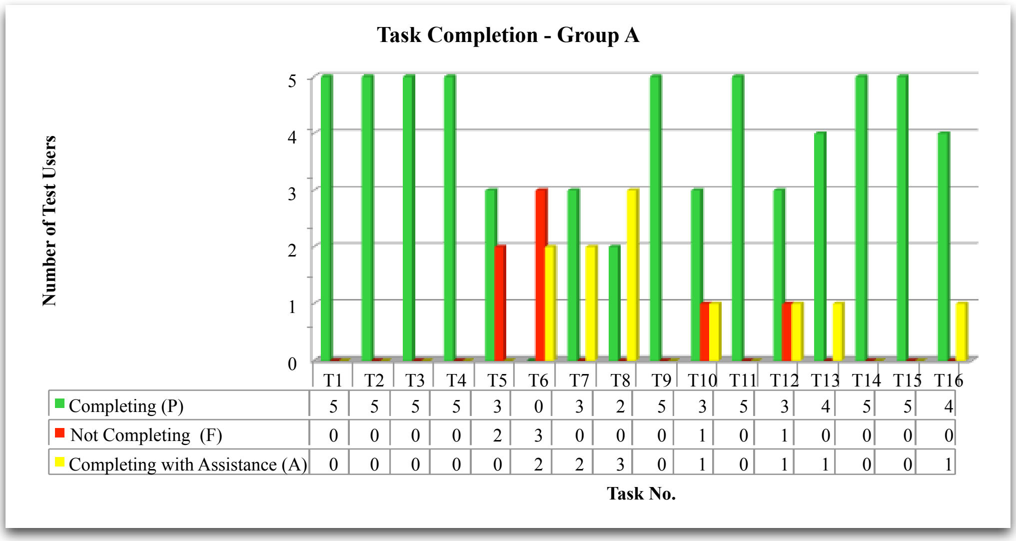
This subsection describes the evaluation of the tested software component AVL InMotion relating to the tasks' level of completion by test participants of user group **A** (TP1, TP3, TP4, TP6 and TP10).

Table 11 shows an overview of the individual results of task completion. Figure 66 lists the statistics of the task completion per test user.

Task Completion - Group A

User \ Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
TP1	P	P	P	P	F	F	P	P	P	P	P	P	P	P	P	P
TP3	P	P	P	P	P	A	P	A	P	A	P	P	P	P	P	P
TP4	P	P	P	P	F	F	P	A	P	F	P	A	A	P	P	A
TP6	P	P	P	P	P	A	A	P	P	P	P	F	P	P	P	P
TP10	P	P	P	P	P	F	A	A	P	P	P	P	P	P	P	P

Table 11: Group A - Overview of Task Completion



	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
Completing (P)	5	5	5	5	3	0	3	2	5	3	5	3	4	5	5	4
Not Completing (F)	0	0	0	0	2	3	0	0	0	1	0	1	0	0	0	0
Completing with Assistance (A)	0	0	0	0	0	2	2	3	0	1	0	1	1	0	0	1

Figure 66: Group A - Task Completion Statistics

In total, the test participants of group **A** completed 77 % of the tasks without help, 14 % of the tasks were completed with assistance and 9% of the tasks could not be completed by some test users (see Figure 67).

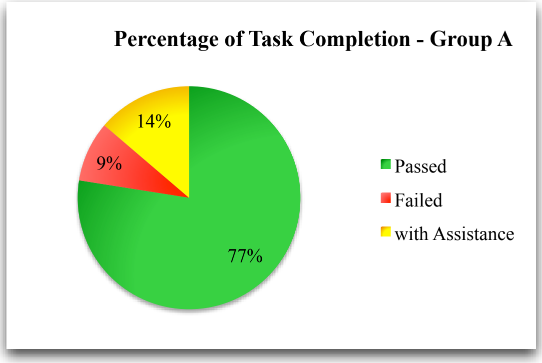


Figure 67: Group A - Percentage of Task Completion

Table 12 shows the percentage of the task completion rate.

- In total, the users finished eight tasks [T1, T2, T3, T4, T9, T11, T14 and T15] without assistance to 100 %.
- Four tasks were completed partly with help and the following percentages show the task completion rate with assistance in these tasks:
  - T7: 60%[P] : 40%[A]
  - T8: 40%[P] : 60%[A]
  - T13: 80%[P] : 20%[A]
  - T16: 80%[P] : 20%[A]
- There were difficulties with four tasks, which were not completed by all users:
  - T5: 60%[P] : 40%[F]
  - T6: 40%[A] : 60%[F]
  - T10: 60%[P] : 20%[A] : 20%[F]
  - T12: 60%[P] : 20%[A] : 20%[F]

Task No.	Passed (P)	with Assistance (A)	Failed (F)
T1	100%	0%	0%
T2	100%	0%	0%
T3	100%	0%	0%
T4	100%	0%	0%
T5	60%	0%	40%
T6	0%	40%	60%
T7	60%	40%	0%
T8	40%	60%	0%
T9	100%	0%	0%
T10	60%	20%	20%
T11	100%	0%	0%
T12	60%	20%	20%
T13	80%	20%	0%
T14	100%	0%	0%
T15	100%	0%	0%
T16	80%	20%	0%

Table 12: Group A - Task Completion Rate in Percent

The users, who did not manage these tasks, had the biggest problems on performing T5 and T6 (see Table 10) and they also did not accomplish the tasks with assistance.

Time Schedule for Task Completion

Figure 68 shows the time taken for the task completion of each test subject in test group **A**.

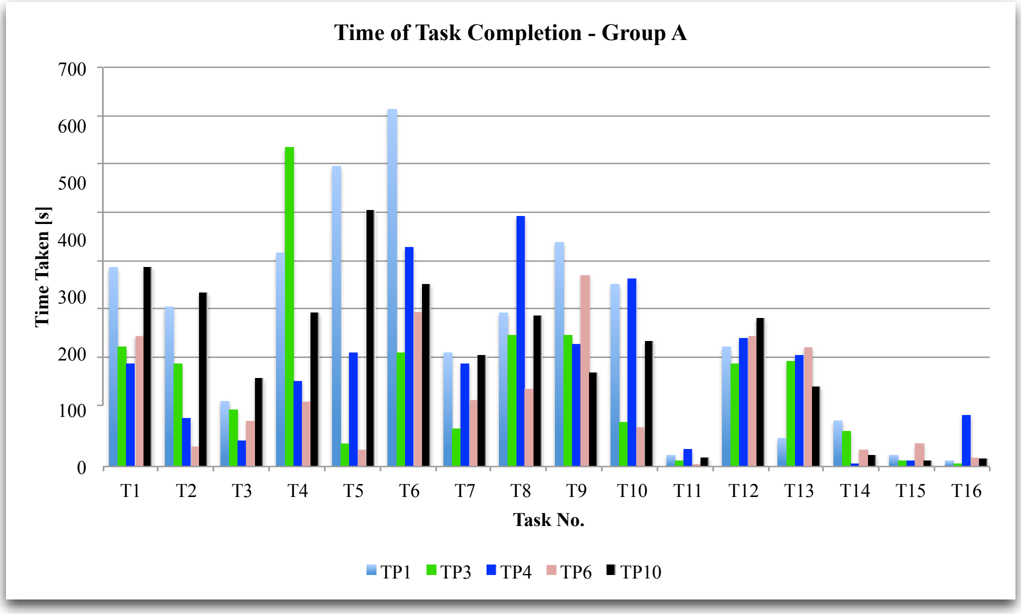


Figure 68: Group A - Time of Task Completion Rate

Table 13 presents the respective data table of the needed time for each test user of this group.

Time of Task Completeness - Group A

User \ Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
TP1	350	280	115	375	527	627	200	270	394	320	20	210	50	80	20	10
TP3	210	180	100	560	40	200	66	230	230	78	10	180	185	62	10	5
TP4	180	85	45	150	200	385	180	439	215	330	30	225	195	5	10	90
TP6	230	35	79	115	30	270	118	135	337	70	5	230	210	30	40	15
TP10	350	305	155	270	450	320	195	265	165	220	15	260	140	20	10	14
Time Scheduled	180	180	120	360	180	240	240	360	240	300	120	360	300	120	120	120

Table 13: Group A - Time of Task Completion

In Table 13, the row *Time Scheduled* indicates the given time limit for the test subjects in order to solve the tasks. The fields that are marked in the colour blue identify those users who needed more time to handle the specified task. T1, which is marked in gray in the table, is not considered for this analysis, because the background for this task was to get a first and quick overview of AVL InMotion.

From these results, one can conclude that some users had problems in carrying out the following tasks: T2, T3, T4, T5, T6, T8, T9 and T10. The majority of the users in group **A** took longer to accomplish tasks T5 and T6.

5.1.2.2. Evaluation of Standalone Third-Party Application (Group B)

In this subsection, the evaluation of the tested standalone third-party application is described relating to the tasks' level of completion by test participants of user group **B** (TP2, TP5, TP7, TP8 and TP9).

An overview of the individual results of task completion is displayed in Table 14. Figure 70 shows the statistics of task completion per test user.

Task Completion - Group B

User \ Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
TP2	P	P	A	P	F	P	P	P	P	P	A	F	P	P	P	P
TP5	P	P	P	P	F	A	F	P	P	F	P	P	P	P	P	P
TP7	P	A	A	A	A	F	F	P	P	A	A	F	A	P	P	P
TP8	P	P	A	P	F	P	A	P	P	P	F	F	P	F	P	P
TP9	P	A	A	P	F	A	A	P	P	P	F	P	A	P	P	P

Table 14: Group B - Overview of Task Completion

In total, the test participants in group **B** completed 61 % of the tasks without help, 21 % of the tasks were completed with assistance and 18 % of the tasks could not be completed by some test users (see Figure 69).

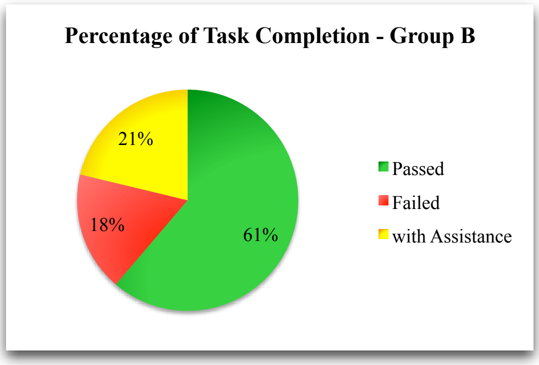


Figure 69: Group B - Completed Tasks

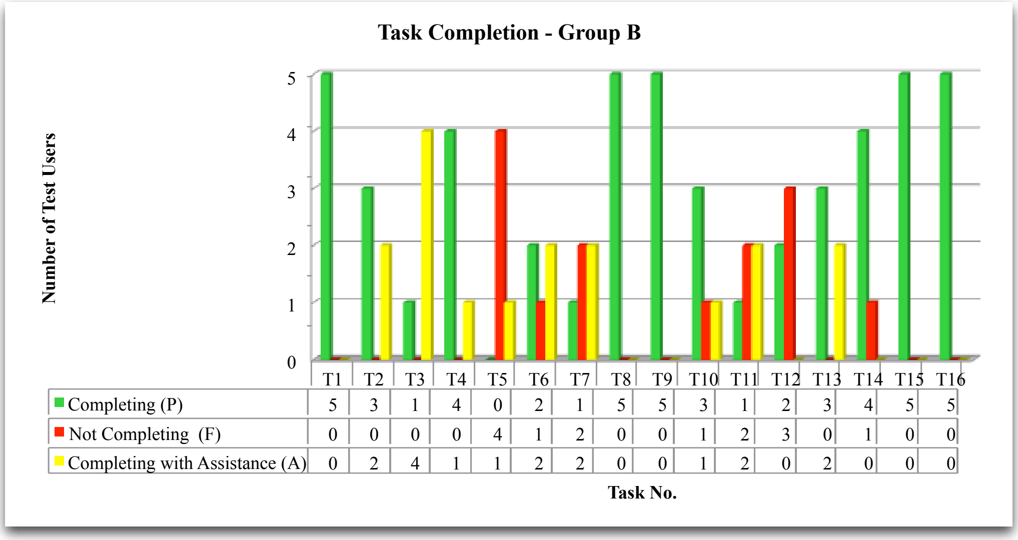


Figure 70: Group B - Task Completion Statistics

Table 15 shows the task completion rate in percent.

- In total, the users finished five tasks [T1, T8, T9, T15 and T16] without assistance to 100 %.
- Four tasks were completed partly with help and the following percentages show the task completion rate with assistance:
  - T2: 60%[P] : 40%[A]
  - T3: 20%[P] : 80%[A]
  - T4: 80%[P] : 20%[A]
  - T13: 60%[P] : 40%[A]
- There were difficulties with seven tasks, which were not completed by all users:
  - T5: 20%[A] : 80%[F]
  - T6: 40%[P] : 40%[A] : 20%[F]
  - T7: 20%[P] : 40%[A] : 40%[F]

Task No.	Passed (P)	with Assistance (A)	Failed (F)
T1	100%	0%	0%
T2	60%	40%	0%
T3	20%	80%	0%
T4	80%	20%	0%
T5	0%	20%	80%
T6	40%	40%	20%
T7	20%	40%	40%
T8	100%	0%	0%
T9	100%	0%	0%
T10	60%	20%	20%
T11	20%	40%	40%
T12	40%	0%	60%
T13	60%	40%	0%
T14	80%	0%	20%
T15	100%	0%	0%
T16	100%	0%	0%

Table 15: Group B - Task Completion Rate

- T10: 60%[P] : 20%[A] : 20%[F]
- T11: 20%[P] : 40%[A] : 40%[F]
- T12: 40%[P] : 60%[F]
- T14: 80%[P] : 20%[F]

The users who did not manage the tasks had the biggest problems performing T5, T7, T11 and T12 (see Table 10). They also did not accomplish the tasks with assistance.

Time Schedule for Task Completion

Figure 71 shows the time taken for the task completion of each test user in test group **B**.

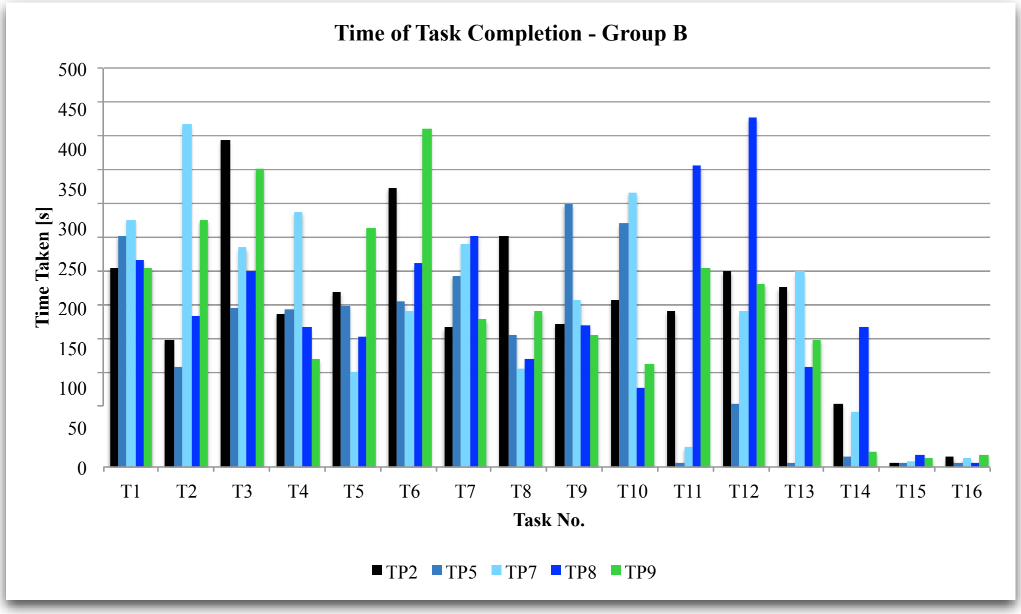


Figure 71: Group B - Time of Task Completion Rate

Table 16 presents the respective data table of the needed time of each test user in this group.

The row *Time Scheduled* in Table 16 indicates the given time limit for the test subjects in order to solve the tasks. The fields that are marked in the colour blue identify those users who needed more time to handle the specified task. T1, which is marked in gray in the table, is not considered for this analysis, because the background for this task was to get a first and quick overview of the standalone third-party application.

Task / User	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
TP2	250	160	410	192	220	350	175	290	180	210	195	245	225	80	5	13
TP5	290	125	200	198	202	208	240	165	330	305	5	80	5	14	5	5
TP7	310	430	275	320	120	195	280	123	210	345	25	195	245	70	7	12
TP8	260	190	245	175	163	255	290	136	178	100	378	438	125	175	15	5
TP9	250	310	375	135	300	425	185	195	165	130	250	230	160	20	12	15
Time Scheduled	180	180	120	360	180	240	240	360	240	300	120	360	300	120	120	120

Time of Task Completeness - Group B

Table 16: Group B - Time of Task Completion

From these results, one can conclude that some users had problems carrying out the following tasks: T2, T3, T5, T6, T7, T9, T10, T11, T12 and T14. The majority of the users in group **B** took longer to accomplish tasks T2, T3, T5, T6 and T11. The major time problem in this test group was in solving task T3, because all users needed more time than scheduled.

5.1.2.3. Comparison of Task Completion

In this subchapter, the comparison of task completion is presented relating to the two tested software systems AVL InMotion and the standalone third-party application.

The side-by-side charts provide an overview of the different task completion rates, which were calculated in the previous chapters 5.1.2.1 and 5.1.2.2. Each comparison indicates the system, which performed better with respect to the various task completion.

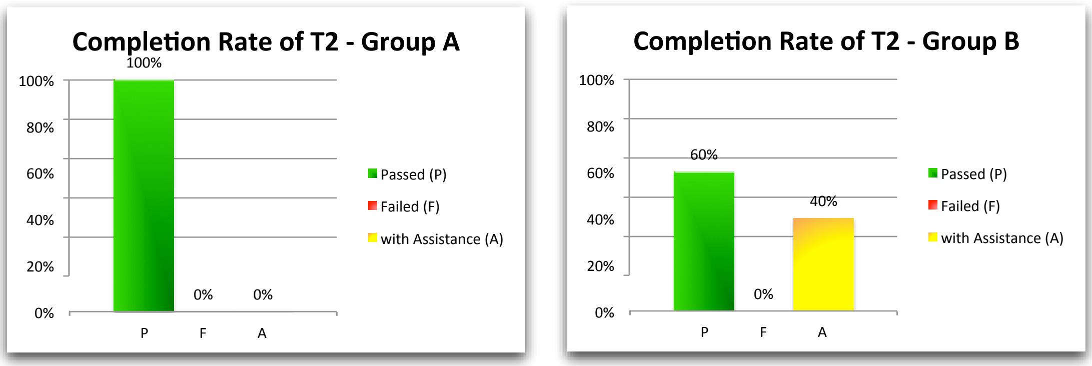


Figure 72: Comparison of Task Completion - T2

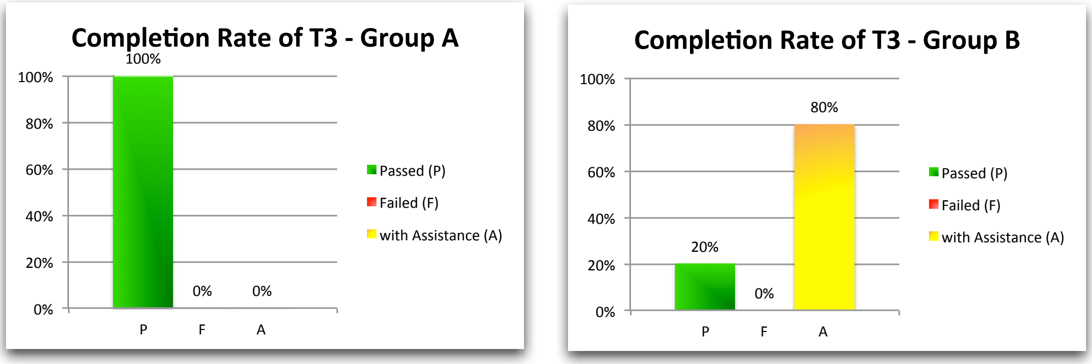


Figure 73: Comparison of Task Completion - T3

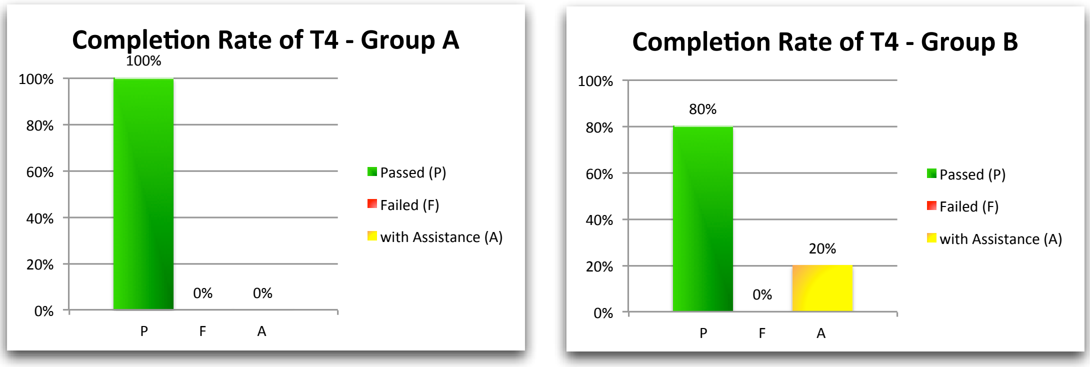


Figure 74: Comparison of Task Completion - T4

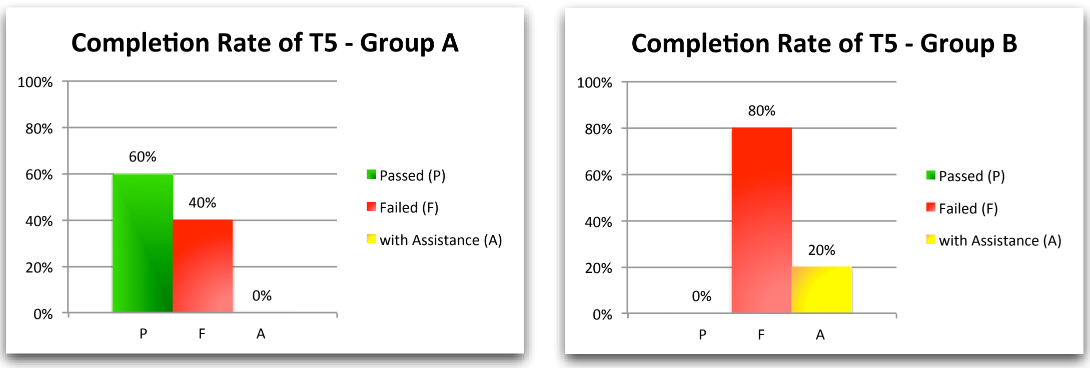
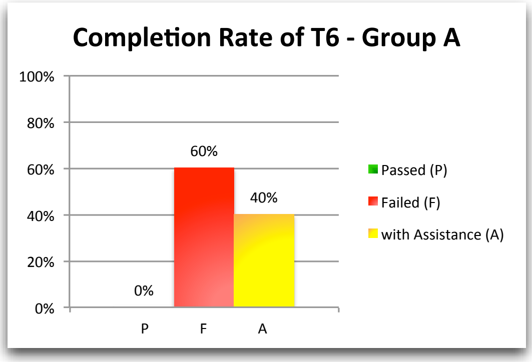


Figure 75: Comparison of Task Completion - T5



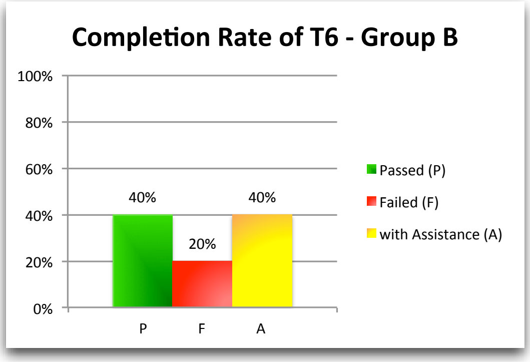
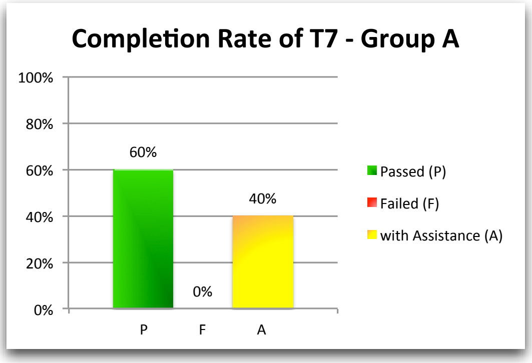


Figure 76: Comparison of Task Completion - T6



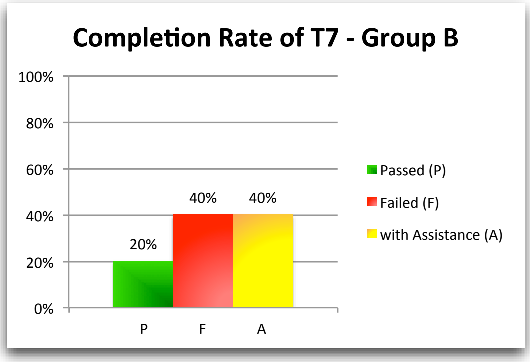
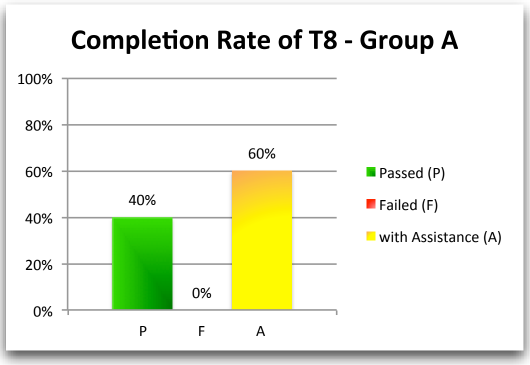


Figure 77: Comparison of Task Completion - T7



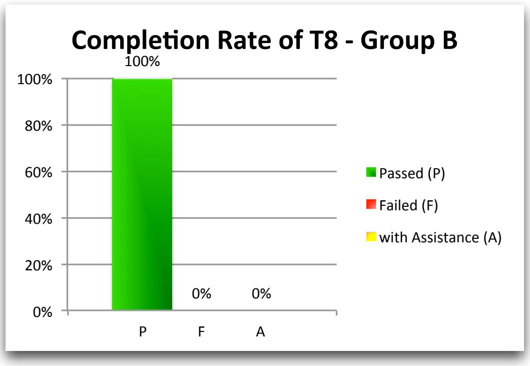


Figure 78: Comparison of Task Completion - T8

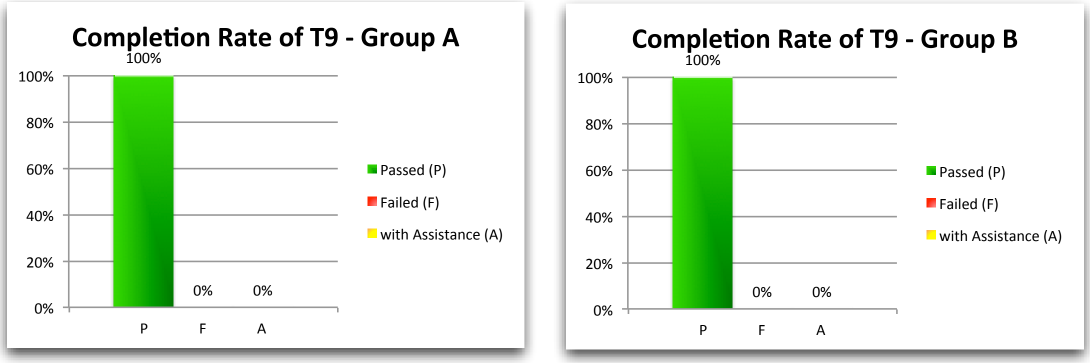


Figure 79: Comparison of Task Completion - T9

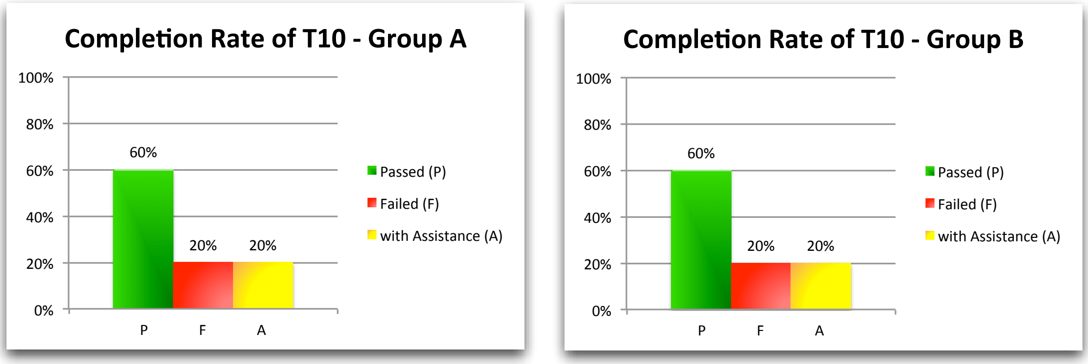


Figure 80: Comparison of Task Completion - T10

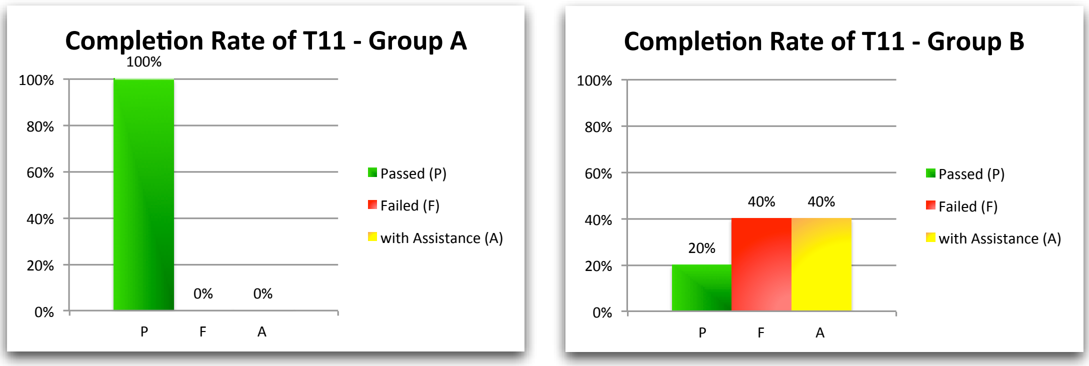


Figure 81: Comparison of Task Completion - T11

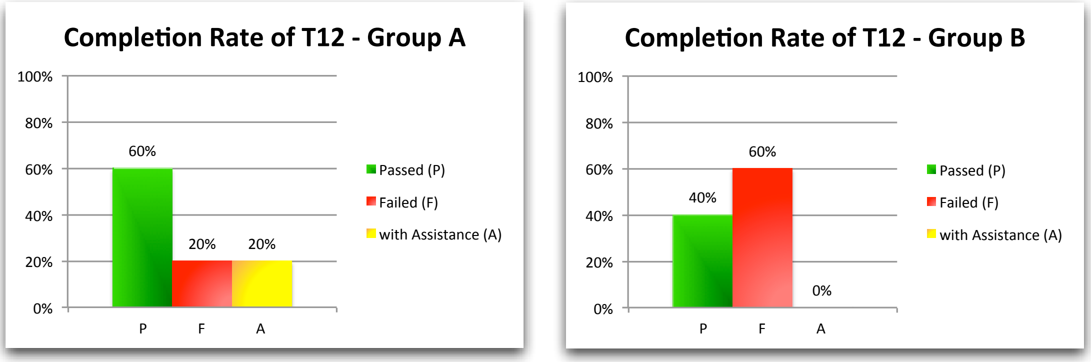


Figure 82: Comparison of Task Completion - T12

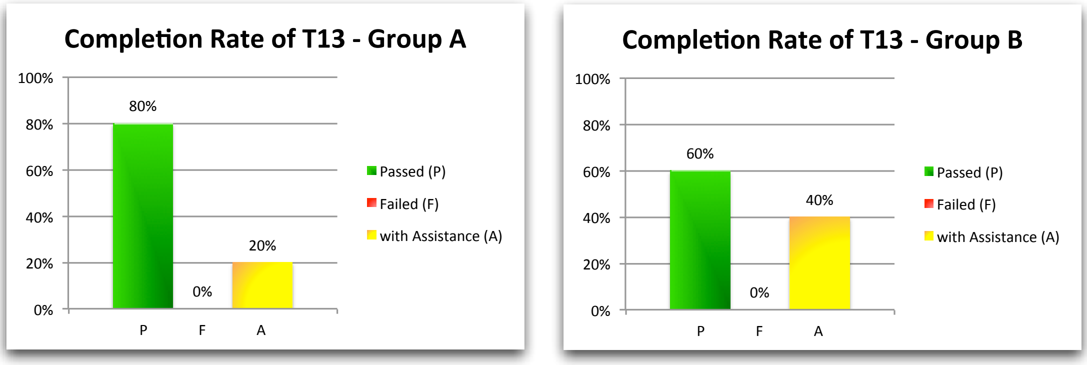


Figure 83: Comparison of Task Completion - T13

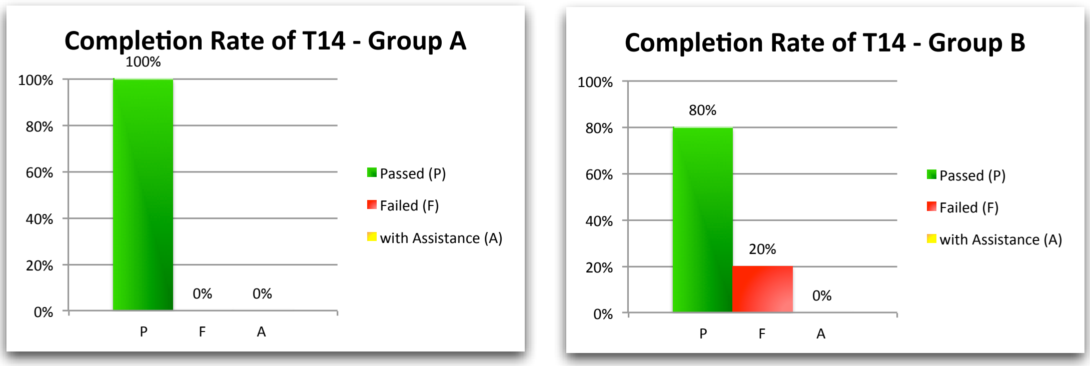


Figure 84: Comparison of Task Completion - T14

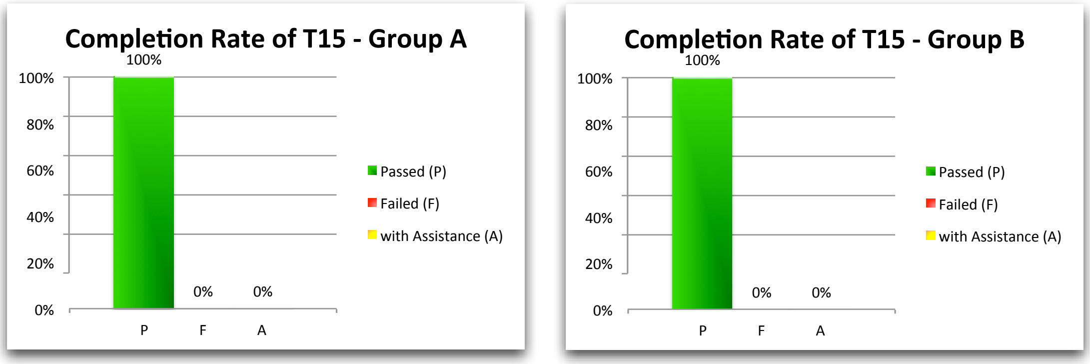


Figure 85: Comparison of Task Completion - T15

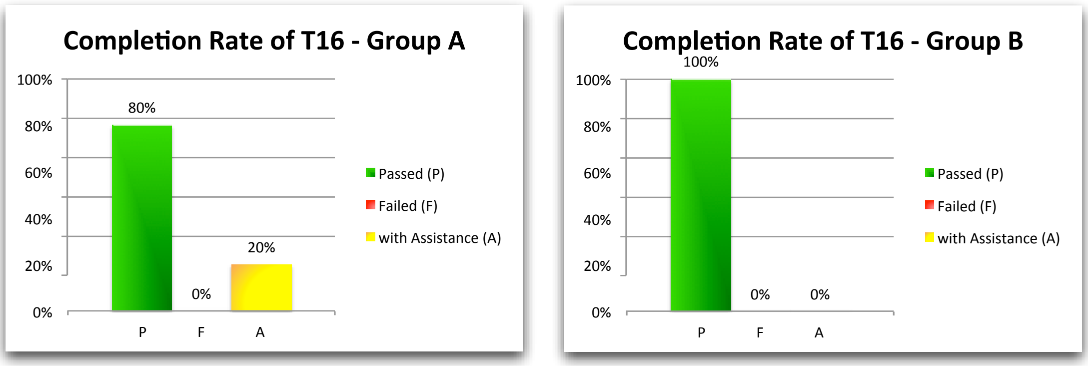


Figure 86: Comparison of Task Completion - T16

5.2. Categorization of Issues and Recommendations

In this chapter, a closer look is given about the main points (see Chapter 5.2.1) that were found by the users who tested AVL InMotion during the usability test. Furthermore, solution approaches, the discussion of several implementations and finally a couple of examples of recommendations are described (see Chapter 5.2.2). The following section also includes various tips and suggestions to improve the design of the software application AVL InMotion, which is integrated into the existing business software suite AVL PUMA Open.

5.2.1. Identified Problems

The identified problems are based on the participants' THA report and observation of the experimenter during the test.

5.2.1.1. Qualitative Statements

In Table 18, the UPs are grouped into manageable categories and they are prioritized according to their severity. Furthermore, a detailed description of each individual UP is provided. The ranking of severity originates from Nielsen (1994), whose scale is stated in Table 17.

Note: Test Participants in test group **B**, who found issues that could also be found in software component AVL InMotion, are set in parentheses in Table 18.

Severity	Description
0	I don't agree that this is a problem at all
1	Cosmetic problem only: need not be fixed unless extra time is available on project
2	Minor usability problem: fixing this should be given low priority
3	Major usability problem: important to fix, so should be given high priority
4	Usability catastrophe: imperative to fix this before product can be released

Table 17: Severity Ratings for Usability Problems
(Nielsen, 1994)

5.2.1.2. Quantitative Statements

Overall, 35 problems were identified, whereby the weighting of the severities is as follows (see Figure 87):

Four problems were assigned to Severity Level 4, eleven problems were classified to the Severity Level 3, nine problems were considered to be Severity Level 2 and eleven problems were rated as Severity Level 1. No problem was referred to as Severity Level 0.

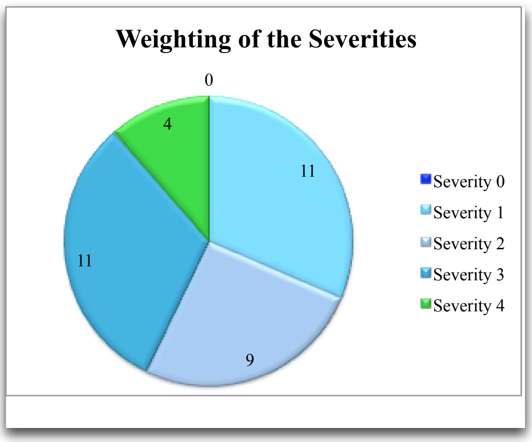


Figure 87: Weighting of the Severities

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
1	T1	General	Structure	Several views of the **MBS editor** are overloaded.	"This is very typical for AVL products.", "Quite complex!"	TP4	1
2	T1, T4, T7, T16	General	Shortcut Keys	Common features of shortcut keys are not available (such as Help `[F1]`, Delete `[DEL]`, Select all `[Ctrl]+[A]`, ...).	"The functionality of `[Ctrl]+[A]` does not work!", "Oh nice, `[F1]` does not work!"	TP1, TP3, TP4, TP6, (TP7,) (TP9,) TP10	3
3	T1	General	Error Messages	Error/ warning messages: Poor wording of buttons. (Label `Yes` versus `OK`)		TP3	2
4	T5	General	Contextual Menu	Inappropriate commands are displayed in context menus. Context menus are available in dialog parts where the user does not expect them.		TP3	1
5	T1	General	Layout	The **File open** dialog is represented in an old windows style.		TP3	1

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
6	T1, T4, T7, T6, T12, T13	General	Labeling	The meaning of labels is often not clear to the user (such as `Bump`, `y-Offset`, `Lateral Dynamics`, ...).	"What does `Bump` mean?!", "No, I cannot add a stop via `Bump` function.", "What does `y-Offset` mean?", "Misleading and quite complex wording!"	TP1, (TP2,) TP3, TP4, (TP5,) TP10	3
7	T9	General	Labeling	The user is missing helpful tooltips for better understanding of labels and buttons.		(TP5), TP10	2
8	T2, T4, T6	General	Drop-Down Lists	Drop-down lists are often not recognized by the user because of the uncommon design.		TP1, TP3, (TP9)	2
9	T7	General	Input Fields	Edit fields change their text color to red if their value is invalid. This is inconsistent to other **PUMA** applications/ dialogs.		TP6, (TP9)	1
10	T10	Driver Settings	Velocity of Driver	The segment marker `Speed Limit` often tempts the user to change the velocity there.		(TP2,) TP3, (TP5,) TP10	1

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
11	T1	Driver Settings	Driver Section	The user is not familiar with the structure of the **driver** section. The representation of enabled and disabled information is not consistent to other **AVL** applications.		TP3, TP6	2
12	T13	Road Settings	Road Settings	The user often tries to set the driving style for road in the **driver** section (for example to drive on the right lane).		(TP2,) TP4, (TP8)	1
13	T2, T4	Segment Settings	Modification of Segments	Rearrangement of road segments is quite difficult; The selection of several road segments is not possible.	“Why am I not able to move two segments?”, “There is no drag and drop functionality.”	(TP2,) TP3, (TP5,) (TP7,) (TP8,) (TP9,) TP10	4
14	T6	Segment Settings	Modification of Segments	The user does not know how to delete a single road segment because the `Delete` button is not obvious to the user.		TP1	1

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
15	T16	Segment Settings	Modification of Segments	There is no `Delete All` button to remove all segments at once. In the current version there is only a menu entry to delete all segments (accessible via the contextual menu).	"I am missing the functionality in order to delete all segments at once."	TP3, TP4	3
16	T2, T4, T9	Segment Settings	Modification of Segments	It takes a lot of time to add new segments. Familiar icons above instead of buttons below the segment list view.	"What the hell... ?"	TP1, TP3, (TP7)	3
17	T9	Segment Settings	Modification of Segments	There is no `Undo` functionality in the segment definition dialog. The user often overwrites the previous added segment, which leads to data loss.	"Oh no, now my previous created settings are deleted."	TP3, (TP5,) (TP8)	4
18	T4	Segment Settings	Layout of Segment Definition	The arrangement of elements in the segment definition view is not obvious to the user.		TP10	2

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
19	T4, T2	Segment Settings	Adding new Segments	Adding new road segments is not intuitive for the user. The selection is not set to the newly inserted segment.	“I am not familiar with this functionality, I miss the common way to insert a new segment.”, “Am I too stupid for this application?”	TP1, (TP2,) TP3, TP4, (TP5,) (TP7,) (TP8,) (TP9,) TP10	4
20	T2, T4, T6	Segment Settings	Adding new Segments	The list segment items are not updated automatically if the user modifies information in the detail view on the right side.	“On the left side there is displayed a `Speed Limit`, but on the right side I have created a `Sign Plate`.”, “Are my modifications stored now?”	TP3, (TP5,) (TP7,) (TP8,) (TP9,) TP10	3
21	T2, T4, T9	Segment Settings	Input	Irrelevant input fields are displayed disabled but can be changed.	“Why am I able to type in something when that edit box is disabled?”, “Invalid input will be ignored.”	TP1, TP3, (TP5,) (TP8,) TP10	3

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici- pants	Sev- erity
22	T8, T6, T13	Segment Settings	Input	There is no input validation on road segment. Whenever the user places a marker that is out of range, the marker is not displayed in the three-dimensional preview.		(TP2,) TP4, TP6, (TP7)	3
23	T13	Segment Settings	Editing of Segments	The user does not recognize the `Override` functionality for single road attributes.		TP4, (TP8)	2
24	T6	Segment Settings	Editing of Segments	Some functions within the segment definition view are not clear to the user. For example the naming of `y-Offset`.	"The y-coordinate normally is a positive value to put the sign on the right place. Why should I enter a negative value to get the marker to the right side?", "To place it on the right side first I have to know the width of the road!"	TP3, (TP5,) TP6, (TP7,) TP10	3

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
25	T9	Segment Settings	Editing of Segments	Some `Bump/ Marker` parameter are similar but differ in their interaction: For example, if a `Sign Plate` should be on the right side, the user has to set the correct offset. But if the user wants to place a tree, he is able to choose between three radio buttons (left, right, both).		(TP5,) TP6, (TP9)	2
26	T5	Bird's Eye View	Bird's Eye View	The user does not find the **Bird's Eye View**, a very helpful functionality for setting up a road based on segments.		TP1, (TP2,) TP4, (TP5,) (TP8,) (TP9)	4
27	T9	Bird's Eye View	Bird's Eye View	There is no `Zoom` functionality in the **Bird's Eye View**, but the displayed route is sometimes cut.		(TP5,) TP10	1

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
28	T6	Bird's Eye View	Bird's Eye View	The yellow makers in the **Bird's Eye View** are very difficult to recognize.	"Oh, there is something - I cannot see that yellow marker that well...", "Yellow on white - nobody will see that!"	(TP2,) TP4, (TP5)	1
29	T10, T7, T12	Maneuver Settings	Maneuver	The functionality behind a **maneuver** is not clear to the user. For example, the user is able to enter values into the `Time` and `Distance` edit field, even though two values are mutually exclusive.	"I am not able to select the road here to define a maneuver."	TP4, (TP5,) TP6, (TP8)	3
30	T3, T8	Movie	Three-dimensional Movie	Information about the road is missing in the three-dimensional road preview movie.	"Where am I able to verify the distance of any road segment?", "Was the 'stop' really on right position?"	TP1, (TP5,) TP6	2
31	T3	Movie	Three-dimensional Movie	There are too many non -relevant settings displayed in the movie menu.		(TP2,) TP4 (TP5,) (TP8)	1

No.	Task No.	Problem Segment	Index	Problem Description	Comments	Test Partici-pants	Sev-erity
32	T3	Movie	Three-dimensional Movie	It is difficult to control the three-dimensional movie. As soon as the user presses the mouse cursor or mouse wheel, the movie perspective changes in an uncommon way.	"The camera allowed the car is able to drive 'under' the road - that makes no sense to me."	TP3, (TP5)	1
33	T8	PUMA online	Three-dimensional Movie in PUMA	The user does not find the `Start Movie` button immediately because the dialog is overloaded.		TP3, TP6, TP10	3
34	T8	PUMA online	Three-dimensional Movie in PUMA	As soon as the user starts the test run in PUMA, the movie window disappears and moves in the back. A correct displayed simulation depends also on the sequence how the user starts the test run.	"Hopefully, the current version will never be delivered."	TP4, TP6	3
35	T9, T8	PUMA online	Online Editing Mode	The user does not recognize that he is able to edit the test in online mode. (The user goes back to state `monitor`, opens the `AVL Explorer` in order to generate the test after modification again)	"What does the generation of a test mean?", "The caption says `Read-Only` , but I am still able to edit parameters?"	TP4, TP6	2

Table 18: Categorization of identified Problems

5.2.2. List of Recommendations

This list of recommendations was prepared on the basis of information provided by the usability test and its identified UPs, which are listed and available in Table 18.

In total, 16 recommendations are presented and each of them includes a problem description as well as suggestion of potential solutions.

5.2.2.1. [R1] Use of Contextual Menu

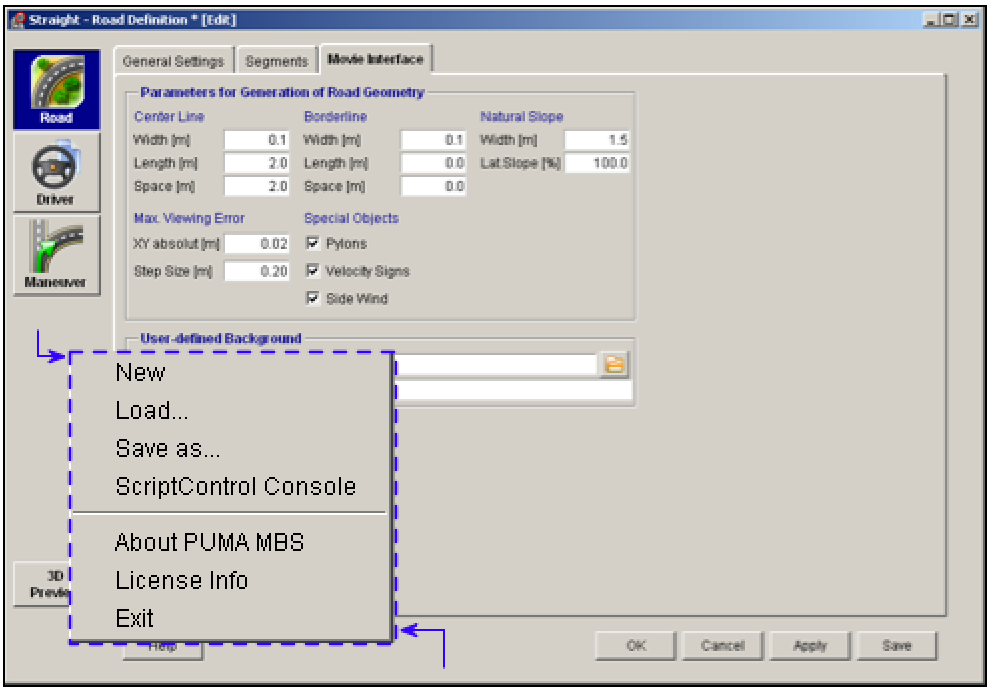


Figure 88: Actual MBS Editor and its Contextual Menu

Severity:	1	Index:	Contextual Menu
Problem Segment:	General MBS Editor	Task:	T5
Problem No.:	4	Test person:	TP3

Problem Description:

Figure 88 shows the context menu in the **MBS Road Definition** dialog by clicking the secondary mouse button on the marked regions.

- The popup menu displays inappropriate commands and, in common use, such content as *loading* or *saving files* is not counted in thus menu structure.
- The user is confused that certain functionalities are only available in this popup menu and the activation of the context menu is only possible in a specified area of the **MBS** editor.
- Context menus are available in dialog parts where the user does not expect them.

Possible Solution:

- Instead of a contextual menu, a *menu bar* is recommended in the **Road Definition** dialog, which displays help documentation or functions to load or save the actual **MBS-Block**.
- Closer inspection of the dialog leads to the assumption that there is a menu bar undesired. In this case, the usage of a menu button is still recommended to provide a small set of related commands.
- However, in general, it is to be diagnosed specifically depending on the usage if such a contextual menu is needed. For this purpose, the context menu can be removed from the **MBS** editor (child dialog) and, instead, integration of the common commands can be done in the previous parent window.

Table 19: [R1] Use of Contextual Menu

5.2.2.2. [R2] Meaning of Labels

Severity:	1-3	Index:	Labeling
Problem Segment:	General MBS Editor	Task:	T1, T4, T6, T7, T9, T10, T12, T13
Problem No.:	6, 7, 10	Test person:	TP1, TP2, TP3, TP4, TP5, TP10

Problem Description:

- The **MBS** editor contains a number of labels, whose meaning is often not clear to the user.
- The problem is that the users do not know what the wording stands for and they are not able to satisfy their work. Figure 89 includes representatives of the misleading naming of such labels and buttons.
- Figure 90 shows an example of an unclear labeling, because the marker `Speed Limit` leads the user to the temptation to change the general velocity instead of doing this in general driver settings.

Possible Solution:

- The dialog should be reworked and improved in its style of presentation. The usage of technical information that is also familiar to the user is a requirement for the provision of a usable software application.
- The translation of some labels or text into English should be revised.
- Infotips, as well as tooltips, are recommended for displaying information about the label or button and explaining the user's choices precisely. The extension of the infotips with an icon or a thumbnail picture used to illustrate multiple operations quickly is more effective.

Table 20: [R2] Meaning of Labels

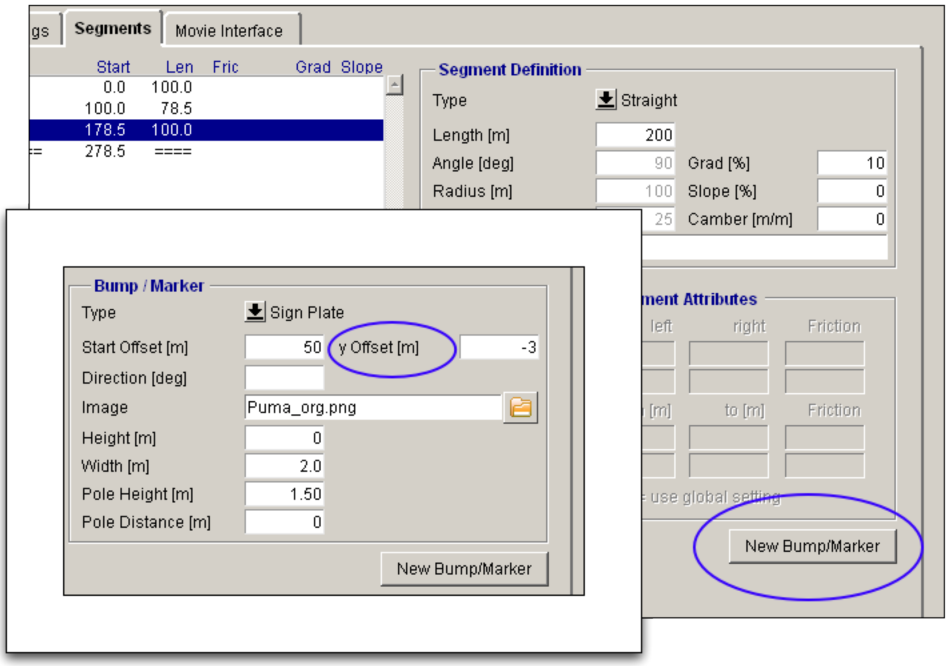


Figure 89: Examples of Misleading Labels

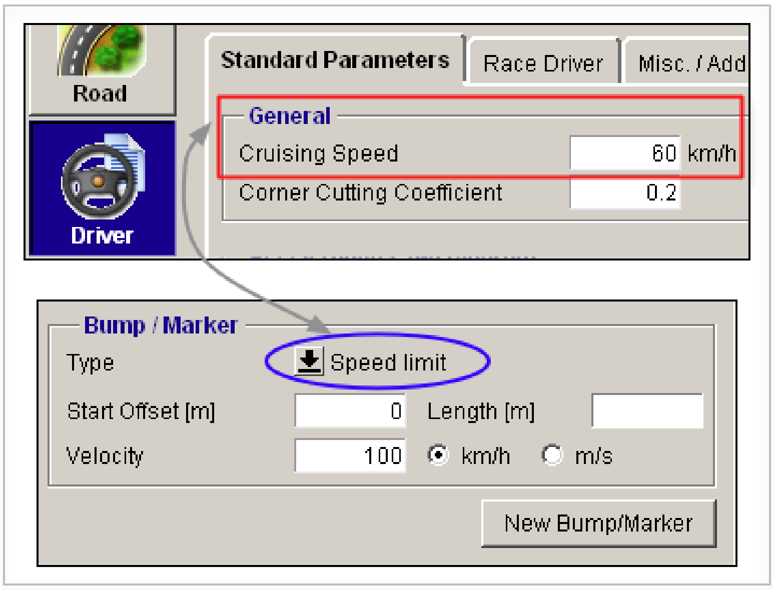


Figure 90: Misunderstanding on the Part of the User

5.2.2.3. [R3] Parameterizing of Segments

Severity:	2-3	Index:	Editing of Segments
Problem Segment:	Segment Settings	Task:	T6, T8, T9, T13
Problem No.:	22, 24, 25	Test person:	TP2, TP3, TP4, TP5, TP6, TP7, TP9, TP10

Problem Description:

Figure 91 displays different kinds of user input options, although the operation mode could be quite the same.

- Some `Bump/ Marker` parameters are similar, but differ in their interactions. For example, adding a `Sign Plate` or `Tree Strip`: If a `Sign Plate` should be positioned on the right hand side of the road, the user has to set the correct offset. Yet if the user wants to place a `Tree Strip`, the user is able to choose between three radio buttons: the tree should be on the left or right hand side, or on both sides of the road.
- The function of the `y-Offset` edit field is not clear to the user. Accordingly, the user is confused, because the label includes the meaning of the vertical axis of the coordinate system. It takes time until the user recognizes that this refers to the marker interval.
- The character `y` suggests a positive value primarily for the user to add the `Sign Plate` to the correct side - but quite the opposite, namely a negative input is valid when completing this task. Whenever the user enters a value that is out of range, the marker is not displayed in the three-dimensional movie so the user is thrown into doubt the instant the sign is added.

Possible Solution:

- In order to avoid misleading outputs, so-called *presets* are recommended that refer values automatically to each parameter when the user decides to add a new marker to a road segment.

- An input validation is also suggested to avoid mistakable inputs of the user.
- Possibly, an adaptation of the radio button mode (see also Figure 92) is suitable to replace the edit field.

Table 21: [R3] Parameterizing of Segments

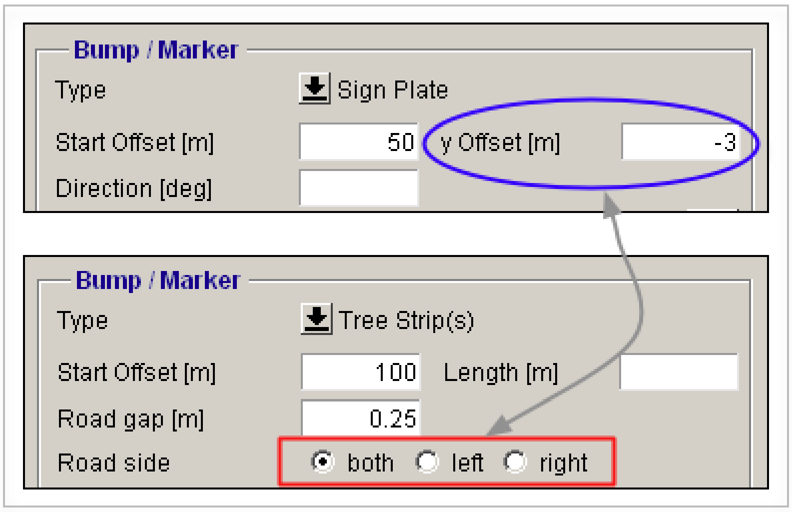


Figure 91: Different Kinds of User Input for quite the same Functionality

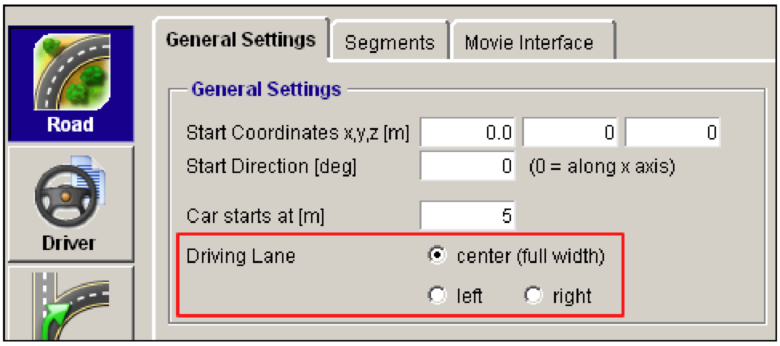


Figure 92: Simple Radio Button Selection

5.2.2.4. [R4] Redesign of Drop-Down Lists

Severity:	2	Index:	Drop-Down Lists
Problem Segment:	General MBS Editor	Task:	T2, T4, T6
Problem No.:	8	Test person:	TP1, TP3, Tp9

Problem Description:

- The drop-down lists are often not recognized by the user. The representation does not relate to the standard design and the drop-down lists look similar to buttons (compare Figure 93).
- Figure 94 shows a specific drop-down menu of AVL InMotion. There are too many items in this menu.

Possible Solution:

- From a design standpoint, the redesign of the drop-down lists is recommended.
- A division of the specific drop-down menu would be useful.
- The usage of structured design is essential in order to support a usable and attractive design feature.

Table 22: [R4] Redesign of Drop-Down Lists

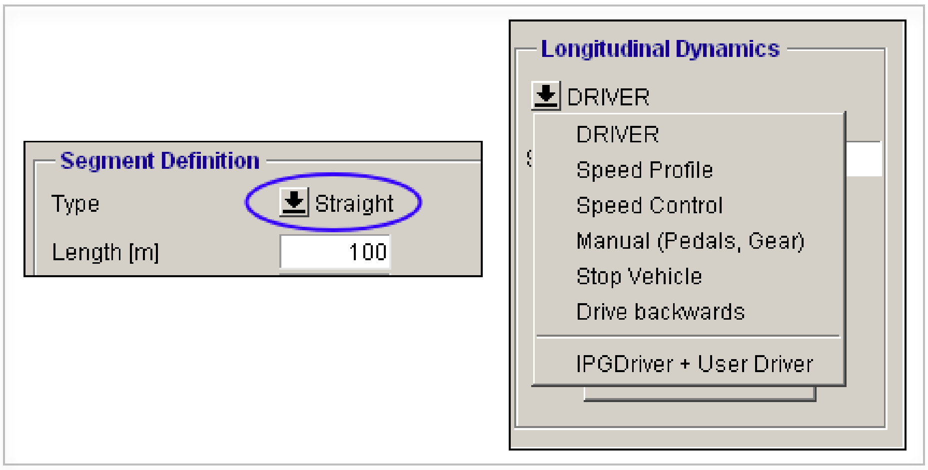


Figure 93: Uncommon Design of Drop-Down Lists

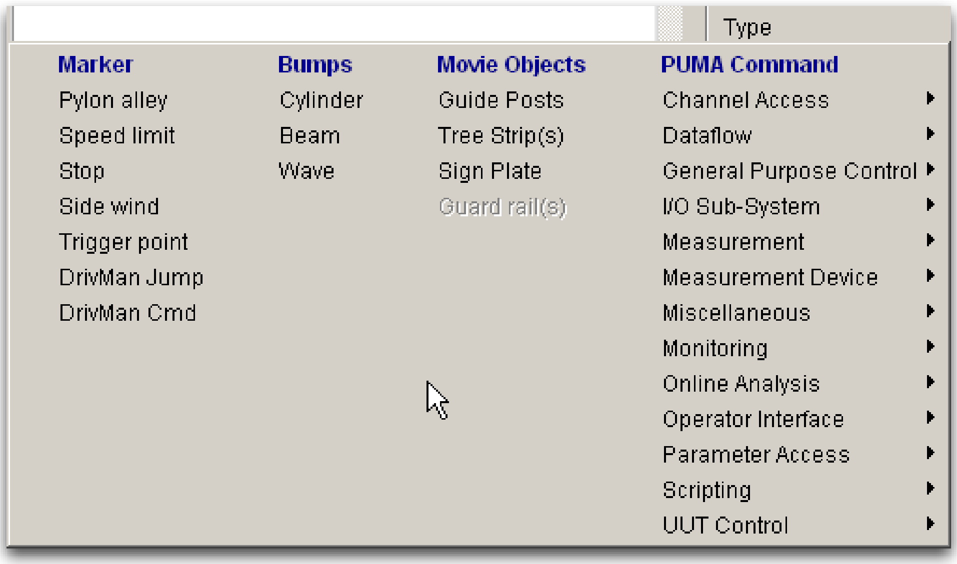


Figure 94: Too many Items in this Drop-Down List

5.2.2.5. [R5] Behavior of Input Fields

Severity:	2	Index:	Input fields
Problem Segment:	General Segment Settings	Task:	T2, T4, T7, T9
Problem No.:	9, 21	Test person:	TP1, TP3, TP5, TP6, TP8, TP9, TP10

Problem Description:

- Irrelevant input fields are displayed as disabled, but can be changed. Figure 95 shows an example where it is possible to enter a `radius` for `Straight` road segment.
- Edit field's color changes to red if their values are invalid. This is inconsistent to other **PUMA** applications and dialogs.

Possible Solution:

- Consistency with all applications is strongly recommended.
- The disabled edit fields must be disabled.

Table 23: [R5] Behavior of Input Fields

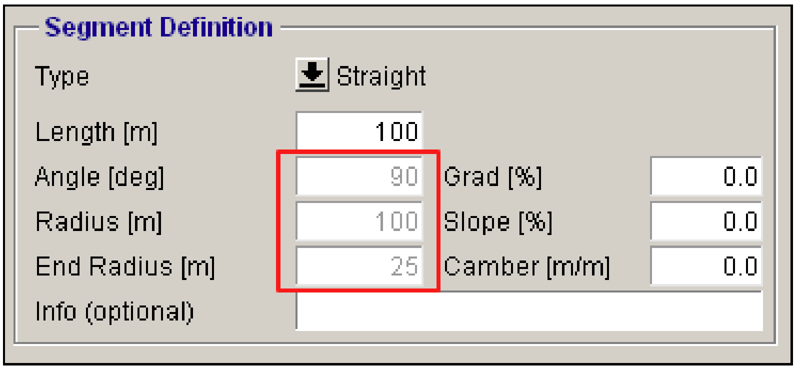


Figure 95: Modifying Irrelevant and Disabled Input Fields

5.2.2.6. [R6] Composition of Driver Input Mask

Severity:	2	Index:	Driver Section
Problem Segment:	Driver Settings	Task:	T1
Problem No.:	11	Test person:	TP3, TP6, TP8

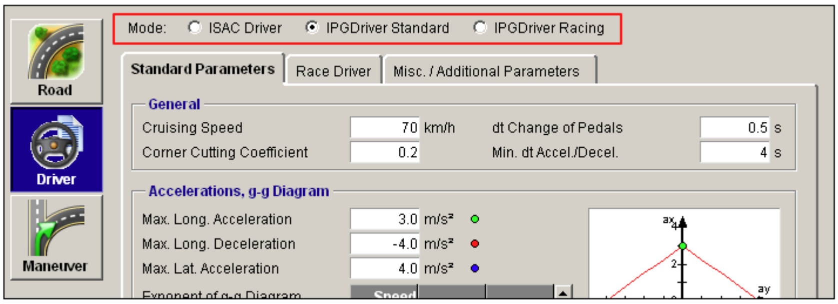
Problem Description:

- The user is not familiar with the structure of the **driver** section.
- The representation of enabled and disabled information is not consistent with other **AVL** applications, which is displayed within Figure 96.

Possible Solution:

- Consistency with all other applications is recommended.

Table 24: [R6] Composition of Driver Input Mask



&

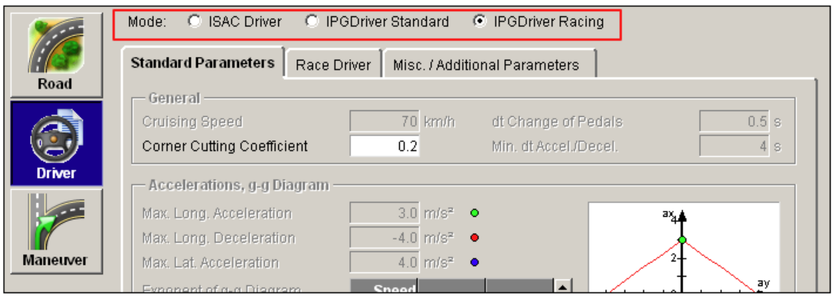


Figure 96: Inconsistent Structure to other AVL Dialogs

5.2.2.7. [R7] Modification of the Sequence of Segments/ Maneuvers

Severity:	4	Index:	Modification of Segments/ Maneuvers
Problem Segment:	Segment/ Maneuver Settings	Task:	T2, T4, T16
Problem No.:	13, 15	Test person:	TP2, TP3, TP4, TP5, TP7, TP8, TP9, TP10

Problem Description:

- The selection of several road segments is not possible so a rearrangement of such segments is quite difficult. There is only one possibility for restructuring (see Figure 97): copy one segment, remove it and add the item to its new position via buttons (`Copy`, `Paste` and `Delete`) again.
- A common functionality is missing, namely to move the elements inside the list view up or down.
- There is no `Delete All` button to remove all segments in the list view at once. In the current version, there is only a menu entry to delete all elements, which is accessible via the contextual menu within the list view (see Figure 98).

Possible Solution:

- The selection of several segments should be possible in order to move, copy and delete multiple route sections - based on drag and drop or keyboard shortcuts.
- The usage of two command buttons (`Move Up` and `Move Down`) that include the functionality to move items in the list view is necessary.
- An additional button, an icon in the toolbar or a shortcut key is required to delete all elements in the segment list at once.

Table 25: [R7] Modification of the Sequence of Segments/ Maneuvers

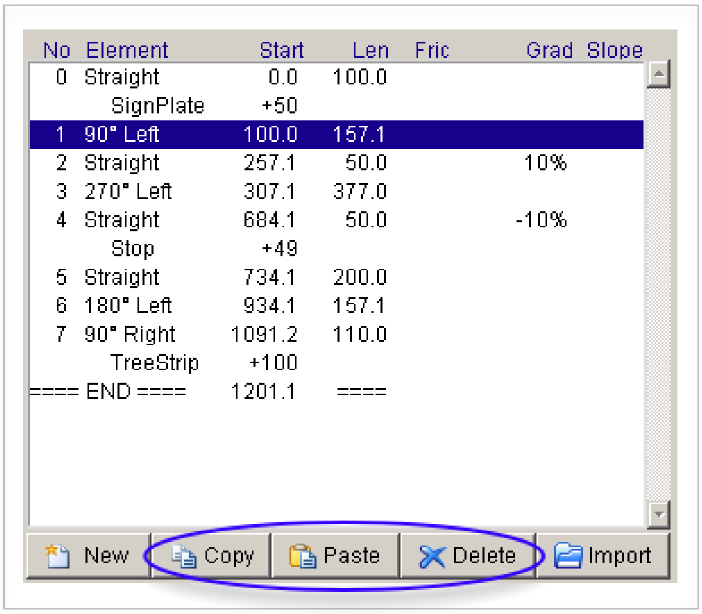


Figure 97: Rearrangement of Items via Button

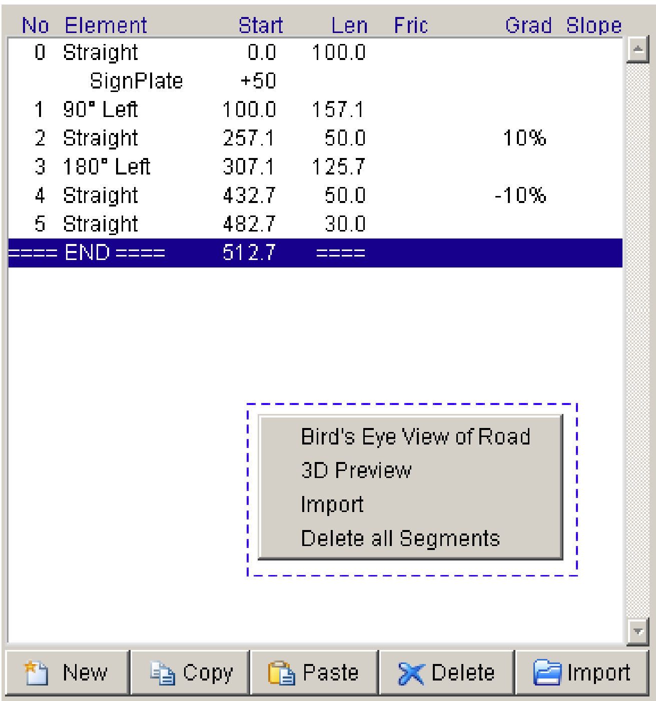


Figure 98: Contextual Menu for Deleting all Items

5.2.2.8. [R8] Adding new Segments/ Maneuvers

Severity:	4/3	Index:	Adding new segments/ maneuvers
Problem Segment:	Segment/ Maneuver Setting	Task:	T2, T4, T6
Problem No.:	19, 20	Test person:	TP1, TP2, TP3, TP4, TP5, TP7, TP8, TP9, TP10
Problem Description: • Adding new segments is not a matter of common sense for the user. The selection is not set to the newly inserted segment. Figure 99 displays an example of the position if the user presses the `New` button to insert a new segment: The newly inserted segment is set before the selected segment. • The list segment items are not updated automatically (see Figure 100) if the user modifies information in the detail view on the right hand side.			
Possible Solution: • The current suggestion is a grid view of the list view providing an intuitive adding mode for segments to the user. • Colored lines and an efficient update algorithm should support the user.			

Table 26: [R8] Adding new Segments/ Maneuvers

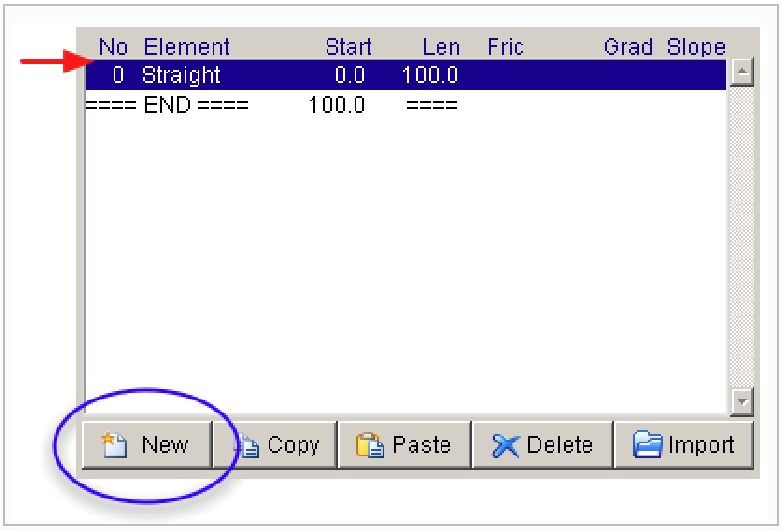


Figure 99: Adding a new Segment is not Obvious

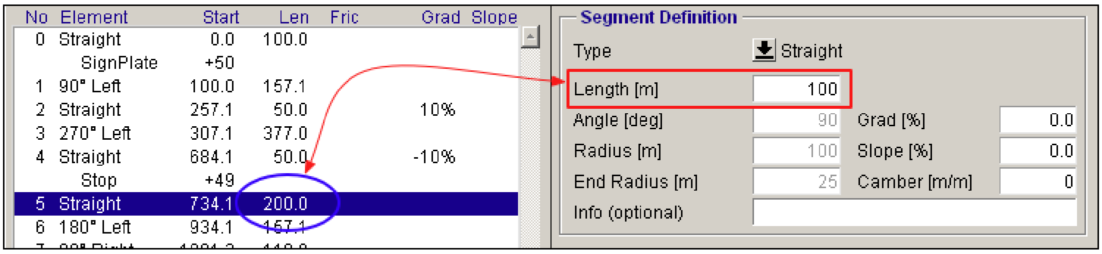


Figure 100: Segments are not Updated Automatically

5.2.2.9. [R9] Structure and Layout of the entire Dialog

Severity:	1-2	Index:	Structure
Problem Segment:	General Layout of Segment Definition View	Task:	T1, T13
Problem No.:	1, 18, 23	Test person:	TP4, TP8, TP10

Problem Description:

- Several views of the **MBS** editor are overloaded.
- The settings of the **Movie Interface** are integrated into the **Road** register (see Figure 101).
- The arrangement of elements in the segment definition view is not obvious to the user.
- In addition, the user also does not recognize the `Override` functionality for single road attributes in this section (see Figure 102).

Possible Solution:

- The separation of the **Movie** Settings would be recommended.
- By providing a clear structure of the arrangement of the edit boxes in segment definition tab, the relationship between different elements could be more transparent.

Table 27: [R9] Structure and Layout of the entire Dialog

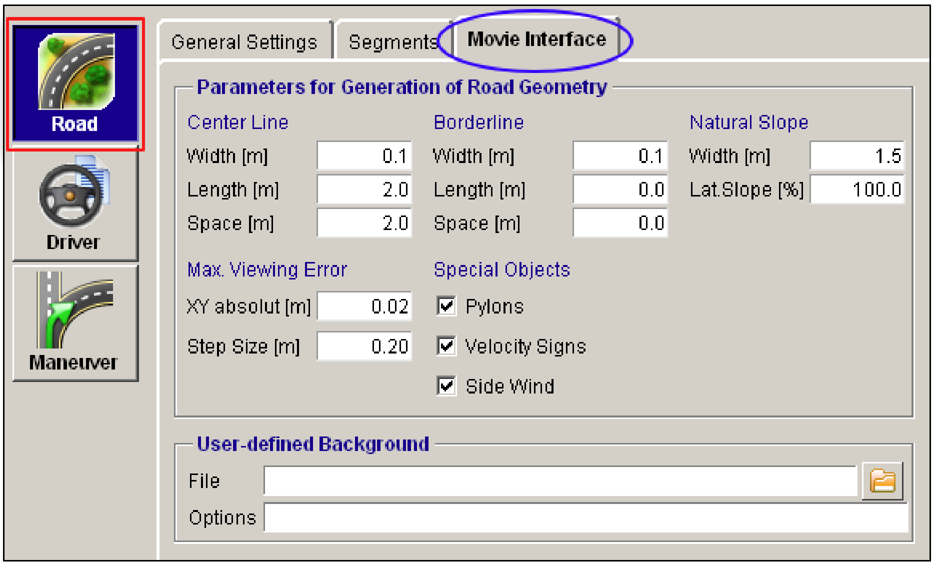


Figure 101: Movie Interface Settings in Road Register

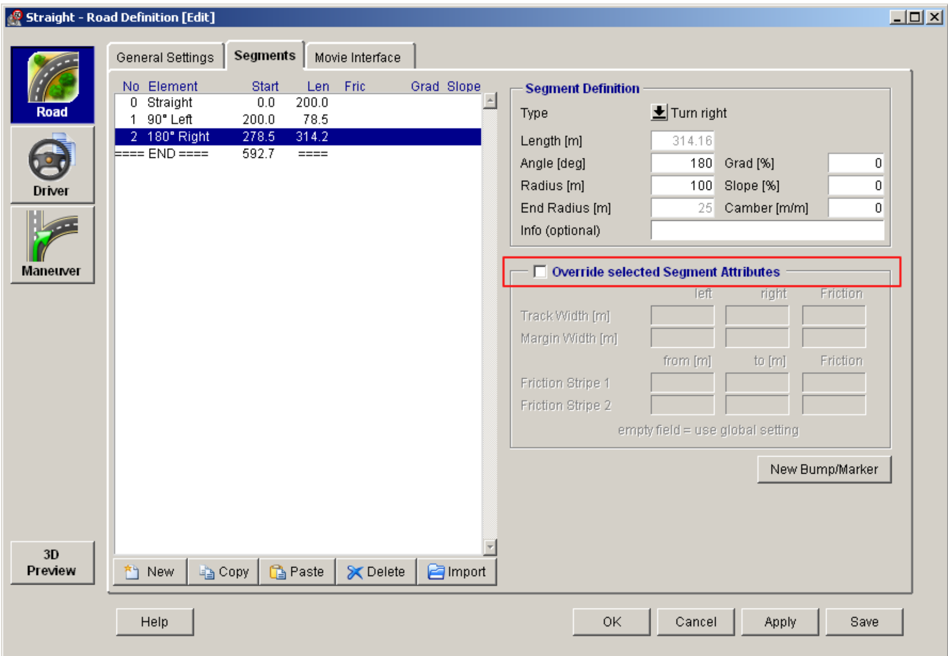


Figure 102: Arrangement of Override Function

5.2.2.10. [R10] Functionality and Layout of Toolbars

Severity:	4	Index:	Modification of Segments
Problem Segment:	Segment/ Maneuver Settings	Task:	T2, T4, T9
Problem No.:	16, 17	Test person:	TP1, TP3, TP5, TP7, TP8

Problem Description:

- One of the main issues in the MBS dialog is that no toolbar exists. It takes a lot of time to add new segments to the list view. The example in Figure 103 displays an incorrect way of adding a new segment by the user. First, the user inserts the segment definition and then the user presses the `New` button.
- There is no `Undo` functionality in the segment definition view. The user often overwrites the previously created segments easily, which leads to data loss.

Possible Solution:

- The Integration of a helpful toolbar into the dialog instead of the buttons below the list view is recommended.
- The usage of icons in the toolbar is more familiar to the user and prevents the loss of previously created settings.

Table 28: [R10] Functionality and Layout of Toolbars

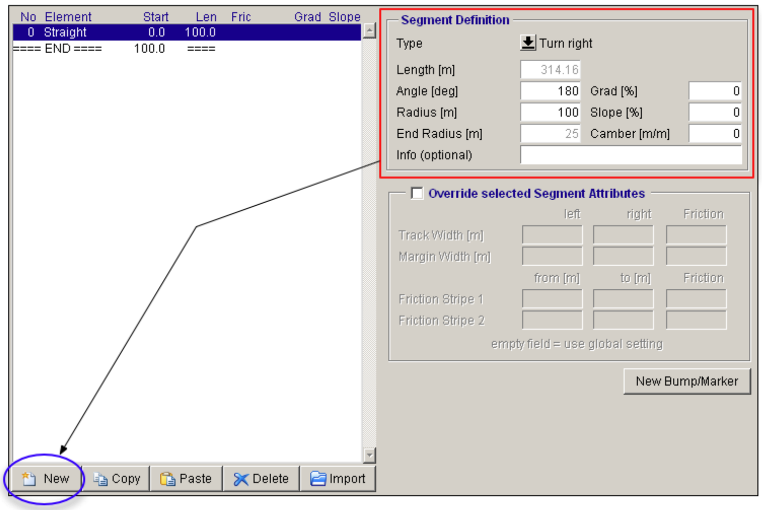


Figure 103: Incorrect way of Adding a Segment

5.2.2.11. [R11] Bird's Eye View of Road

Severity:	4	Index:	Bird's Eye View of Road
Problem Segment:	Bird's Eye View of Road	Task:	T5
Problem No.:	26	Test person:	TP1, TP2, TP4, TP5, TP8, TP9

<u>**Problem Description:**</u>

- The user does not find the **Bird's Eye View of Road** to be a very helpful functionality for setting up a road based on segments.
- In the current dialog, this view is only accessible via the contextual menu within the segments list view (see Figure 104).

<u>**Possible Solution:**</u>

- A button to open the **Bird's Eye View of Road** is strongly recommended.

Table 29: [R11] Bird's Eye View of Road

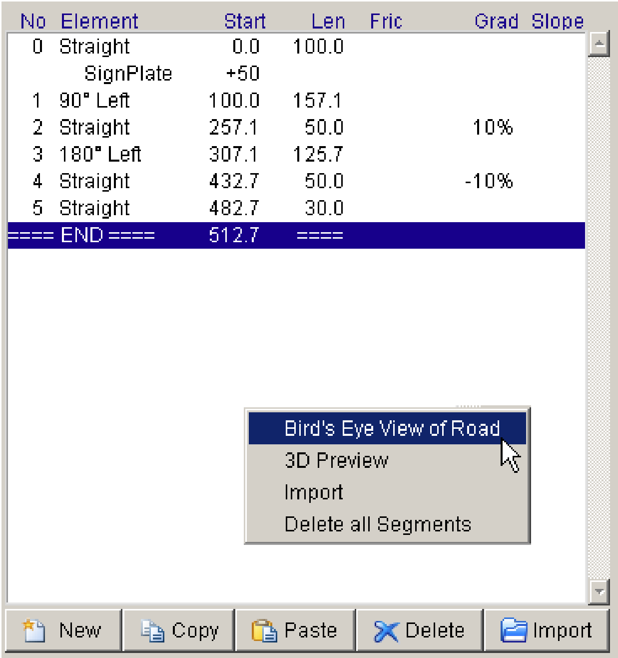


Figure 104: Accessing the Bird's Eye View of Road via Contextual Menu

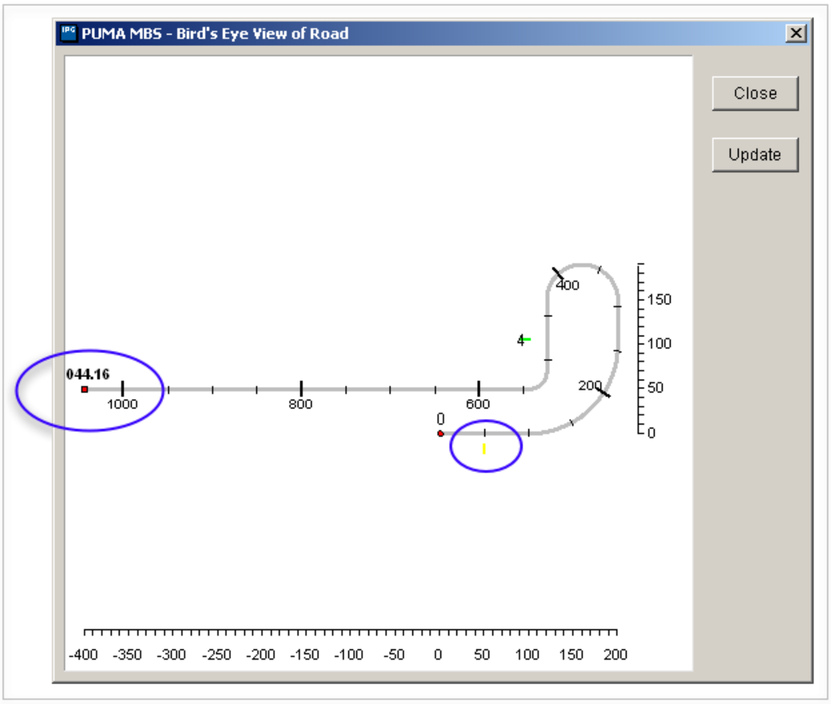


Figure 105: The Bird's Eye View of Road - Displaying Insufficient Information

5.2.2.12. [R12] Functionality of the Bird's Eye View of Road

Severity:	1	Index:	Bird's Eye View of Road
Problem Segment:	Bird's Eye View of Road	Task:	T6, T9
Problem No.:	27, 28	Test person:	TP2, TP4, TP5, TP10

Problem Description:

See Figure 105:

- There is no `Zoom` functionality in the **Bird's Eye View of Road**, but the displayed route, or rather the indicated dimension, is sometimes cut.
- Yellow markers in **Bird's Eye View of Road** are very difficult to recognize, too.

Possible Solution:

- Markers should be displayed in another color or symbol in the preview.
- The user should position the created marker in the correct place as defined in the order, to validate modifications more easily.
- A `Zoom` functionality is also recommended to change the scale of the viewed areas in order to see more or less detail.

Table 30: [R12] Functionality of the Bird's Eye View of Road

5.2.2.13. [R13] Maneuver Settings

Severity:	3	Index:	Maneuver
Problem Segment:	Maneuver Settings	Task:	T7, T10, T12
Problem No.:	29	Test person:	TP4, TP5, TP6, TP8

Problem Description:

- The functionality behind a maneuver is not clear to the user.
- For example see Figure 106, the user is able to enter values into the `Time` and `Distance` edit fields, even though two values are mutually exclusive.

Possible Solution:

- A recommendation would be the usage of radio buttons to prevent technically misleading input by the user.

Table 31: [R13] Maneuver Settings

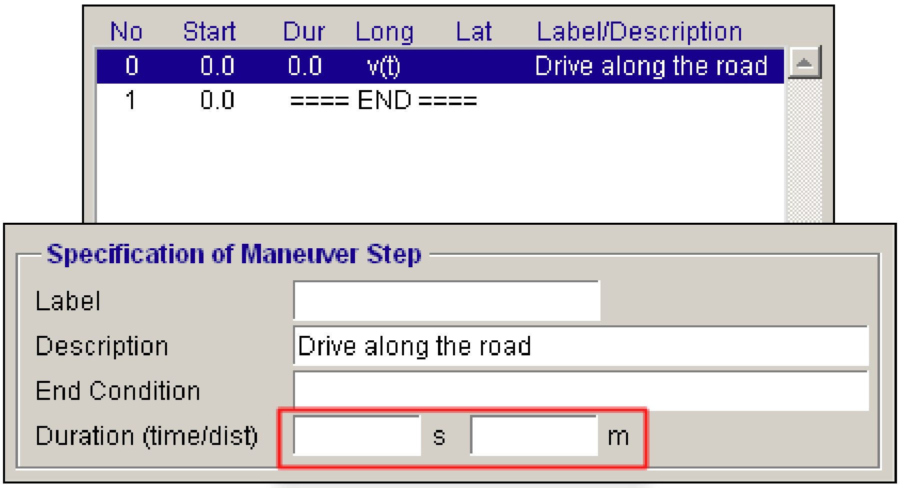


Figure 106: Settings of a Maneuver

5.2.2.14. [R14] Three-dimensional Movie

Severity:	1-2	Index:	Three-dimensional Movie
Problem Segment:	Movie	Task:	T3, T8
Problem No.:	30, 31, 32	Test person:	TP1, TP2, TP3, TP5, TP6, TP8

Problem Description:

- The information about the road is missing in the three-dimensional road preview movie (see Figure 107). The user cannot identify whether the segments match the created settings.
- There are too many irrelevant settings displayed in the movie menu (displayed in Figure 108).
- It is also difficult to control the three-dimensional movie: As soon as the user presses the mouse cursor or mouse wheel, the movie perspective changes in an uncommon way so the user has to return to the default settings via the menu.

Possible Solution:

- Road segments have to be marked with their distance and other information should be provided in the road movie.
- Reduced movie menu items just to essential entries.
- Development of controls to reach a better interaction with the movie perspective.
- The integration of a button or a toolbar to set the default view.

Table 32: [R14] Three-dimensional Movie

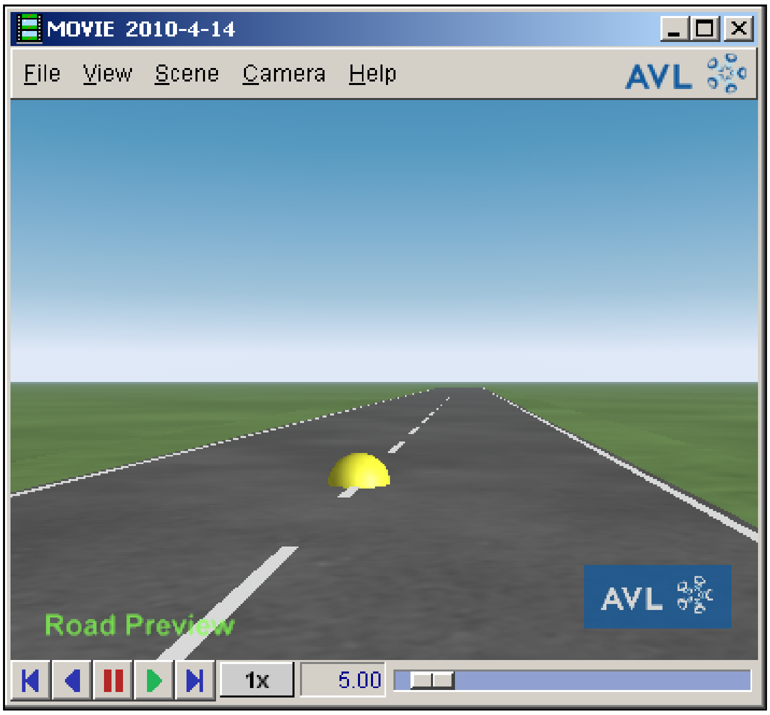


Figure 107: The Three-Dimensional Movie

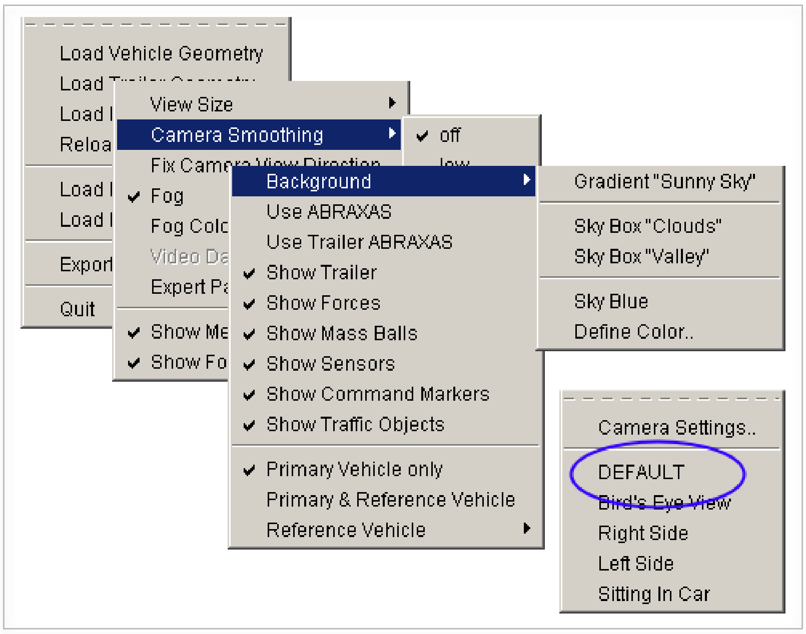


Figure 108: Confusing and Overcharged Movie Menu Entries

5.2.2.15. [R15] Simulation of the 3D movie in PUMA

Severity:	3	Index:	Three-dimensional Movie in PUMA
Problem Segment:	PUMA online	Task:	T8
Problem No.:	33, 34	Test person:	TP3, TP4, TP6, TP10

Problem Description:

- The user does not find the `Start Movie` button immediately because the dialog is overloaded (compare Figure 109).
- As soon as the user starts the test run in PUMA, the movie window disappears and moves to the back.
- A correctly displayed simulation (see Figure 110) depends on the sequence with which the user starts the test run.

Possible Solution:

- Embedding the three-dimensional simulation movie in the **Vehicle Status** dialog is recommendable.
- There are too many irrelevant functionalities displayed in the dialog, which have to be sorted out according to their importance.

Table 33: [R15] Simulation of the 3D movie in PUMA

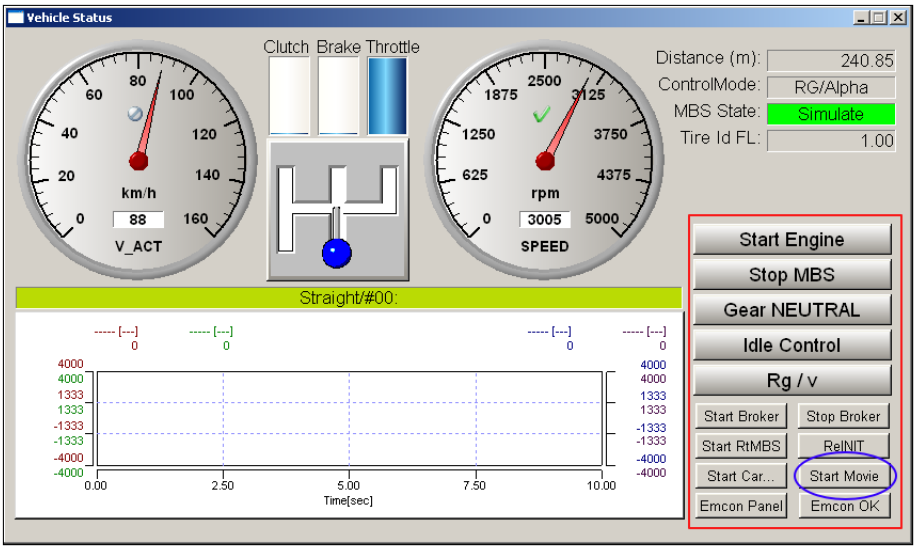


Figure 109: Overloaded Vehicle Status Dialog in PUMA

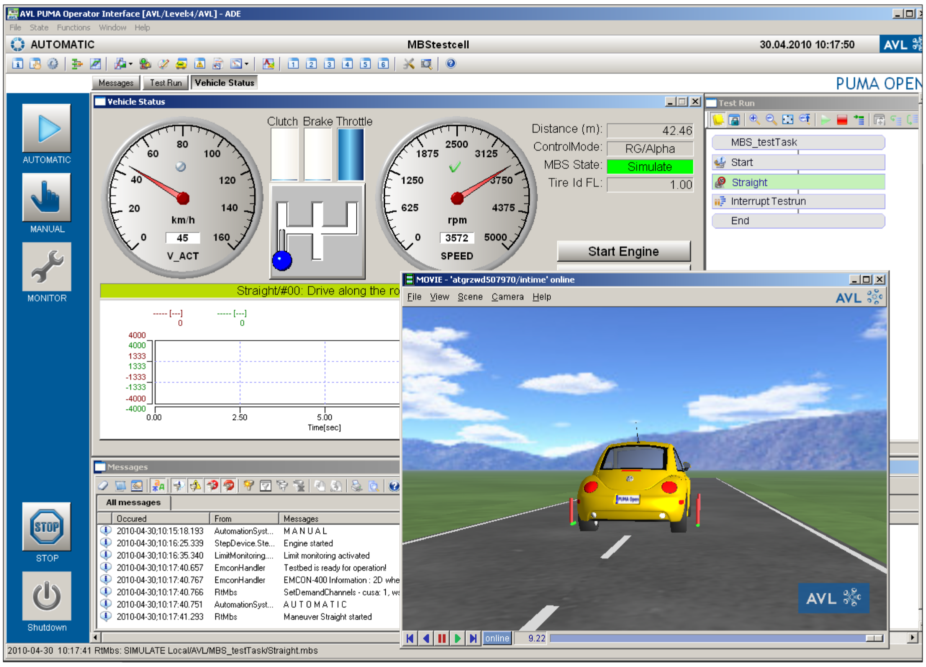


Figure 110: Correct Simulation of Test Run

5.2.2.16. [R16] Online editing mode of Test Run

Severity:	2	Index:	Online Editing mode
Problem Segment:	PUMA online	Task:	T8, T9
Problem No.:	35	Test person:	TP4, TP6

Problem Description:

- The users do not recognize that they are able to edit the test in the online mode. Commonly, the user goes back to the state `Monitor`, opens the `AVL Explorer` in order to generate the test after modification again.
- The user who identifies the online editing mode is confused about the caption of the **MBS** editor that includes the title `Read-Only`.

Possible Solution:

- The functionality for editing the test in the online mode should be presented in a visible way to the user.
- For example, a meaningful label for a button to edit, indicating to user to modify the test parameters online would be a potential solution.

Table 34: [R16] Online Editing Mode of Test Run

5.3. The Redesigned Graphical User Interface

Figure 111 shows a newly revised user interface for AVL InMotion that partially includes the proposed recommendations, which are described in the previous chapter. This dialog view is improved in its style of presentation and provides an intuitive operability of the user interface.

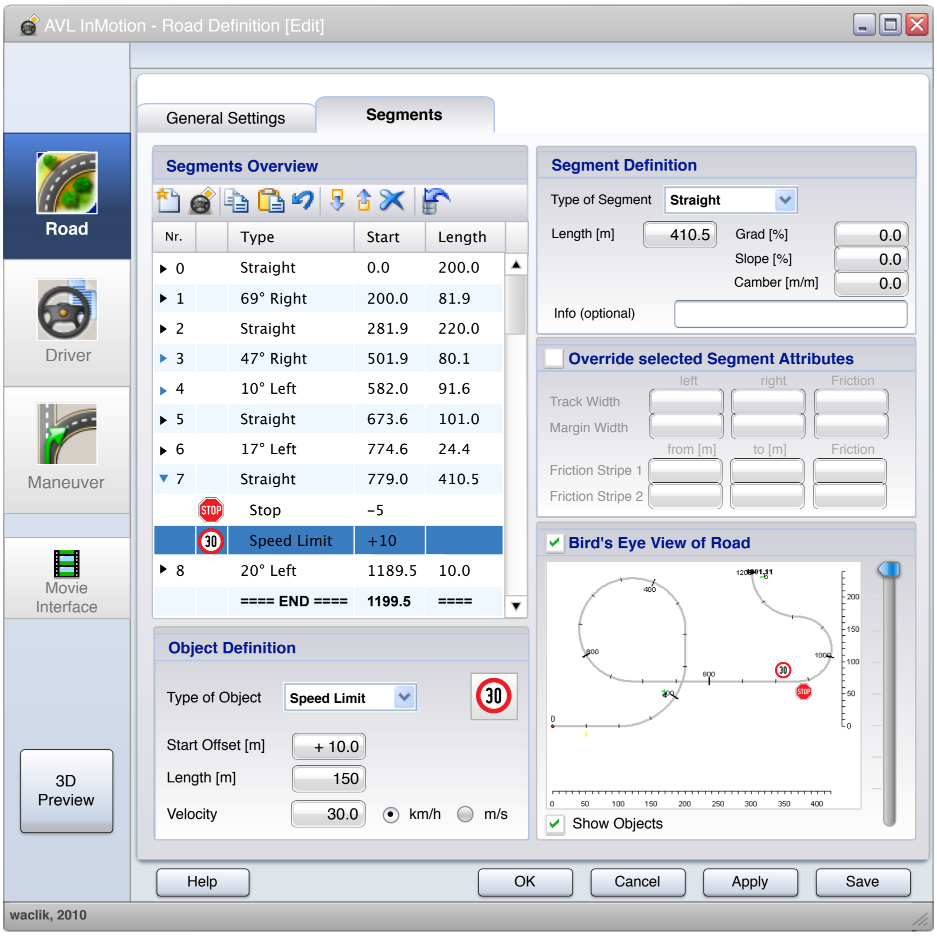


Figure 111: Redesigned User Interface

To get familiar with the created mockup design, Figure 112 is additionally marked with tooltips, which should make the changes clear and understandable. As mentioned before, the revised draft contains a number of design solutions, which were taken from the list of recommendations (see Subchapter 5.2.2).

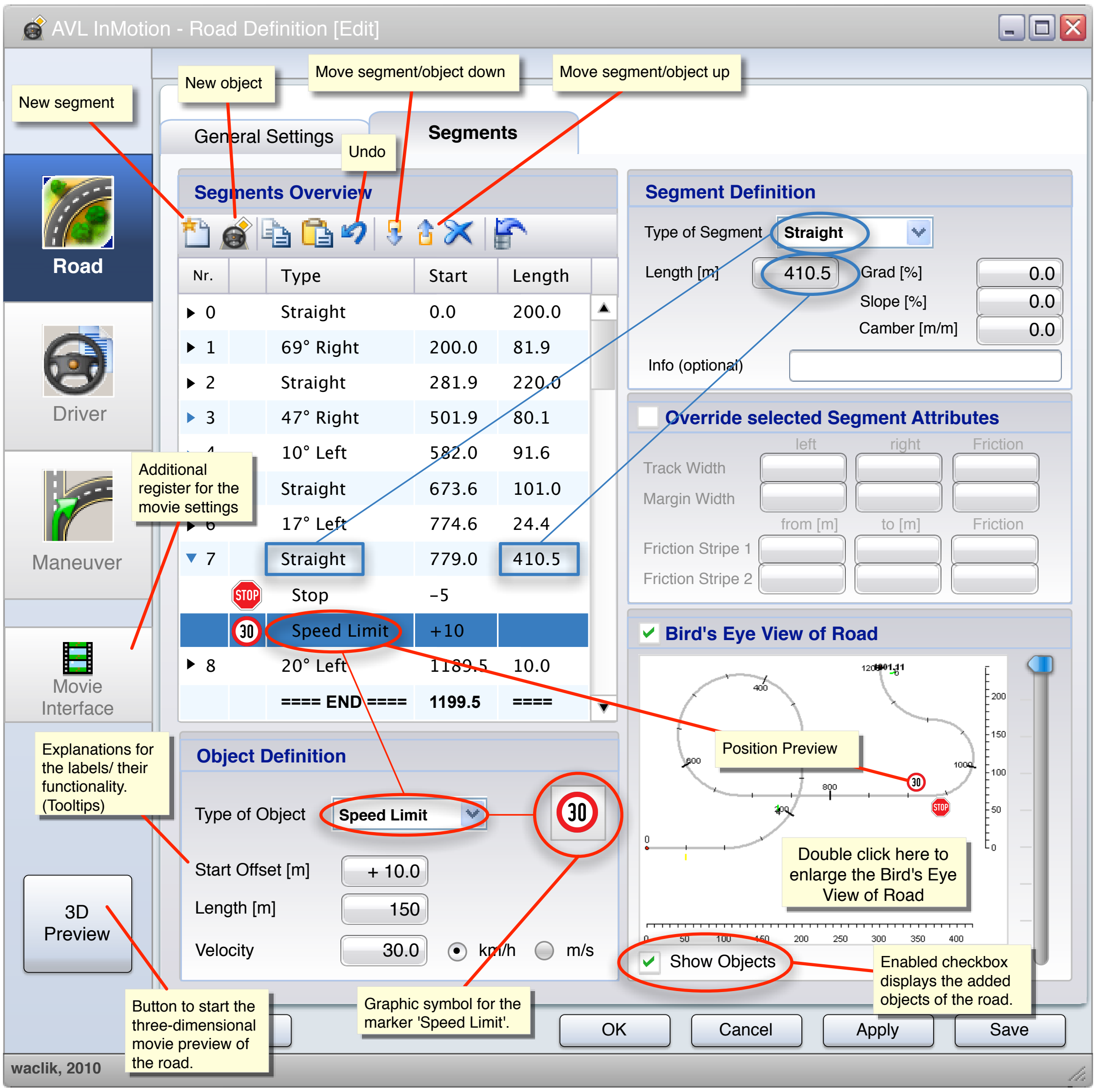


Figure 112: Rearrangement and Redesign of Dialog Elements

First of all, an additional register called *Movie Interface* extended the MBS dialog. This new element specifies the movie settings that are regulated in the *Road* register of the current version of AVL InMotion as described in R9 (see Subsection 5.2.2.9). The separation of these settings to a separate sector is recommended, because there is absolutely no connection between movie and road settings and so it might be better to remove the movie settings from the *Road* register. Furthermore, a button for the preview of the road is specially created on the left hand side of the dialog in order to start the three-dimensional movie with the current configuration.

The redesigned dialog includes explanations for the labels and their functionality, which can be displayed via tooltips (see R2 in the corresponding Subsection 5.2.2.2). Another improvement is the symbolic representation of the chosen object, or rather marker, to avoid unclear operations by the user. In Figure 112, the object Speed Limit is provided with a graphical symbol both in the segment list and in the definition field.

R7, R8 and R10 have got a high severity level (see Subsections 5.2.2.7, 5.2.2.8 and 5.2.2.10). These recommendations relate to the tabs for *Segments* and *Maneuver* settings[16]. A supporting toolbar, on which icons are placed, replaces the buttons at the bottom of the segment list in the redesign. In addition to the existing key-functionalities, there are at least four new function icons, which represent the basic functions such as "Add new object", "Undo", "Move segment/object down" and "Move segment/object up". Moreover, the system contains the drag-and-drop functionality in order to put the selected segment(s) in a different location (see also Subchapter 5.3.1.2). This technique is a common operation to users to add and modify their segments to the list. The colored gridview has been taken into account to provide an intuitive adding mode for the segments to the user. Additionally, the shortcut keys are integrated to this redesigned version too to press the `Delete` button for example.

The display of mandatory input fields as described in R5 (see Subsection 5.2.2.5) must also be mentioned. The selection of the entry as well as adding of new segments to the segment list activates only the associated information fields, which belong to the relevant list view item. For example, the segment *Straight* displays no detail about radius or angle in order to avoid confusion among the users with irrelevant and disabled input fields.

Subchapter 5.3.1.1 provides a design to transform the different object definition fields to have unique appearance to ensure that the format of the configuration settings is consistent and that they include all information considered necessary by developers (see R3 in Subsection 5.2.2.3).

In the current version of AVL InMotion, the Bird's Eye View of Road is only accessible via the contextual menu within the segment list view as described in R11 (see Subsection 5.2.2.11). In the redesigned dialog this specific view is integrated directly. Another alternative listed in R12 (see Subsection 5.2.2.12) is to double click on the Bird's Eye View of Road to enlarge this demonstration of view. Additionally, each road preview can be *zoomed in* on to gain a more detailed view. Furthermore, the added markers of the road configurations are sometimes very difficult to recognize. Instead of colored point symbols improve the display of the geometrical preview and are also positioned in

[16]For simplicity, only the term *Segments* will be used in the following explanations for R7, R8 and R10.

the correct place as defined. An additional checkbox controls the display of the added objects in the geometrical preview of the road.

5.3.1. Adaptation of Individual Elements of the User Interface

Here it needs to be added, that the following dialog designs were created by DI(FH) Stephan Lenhart[17]. The concepts were taken from the recommendations gathered after performing the usability test (see the list of recommendations in Subchapter 5.2.2).

5.3.1.1. Standardized Design for Configuration Settings

As described in R2 (see Subsection 5.2.2.3), a uniform appearance of the marker preferences contributes to a better understanding of using these settings (see Figures 114 and 113). Figure 113 shows the design for the marker of type *Tree Strip(s)*, whereby Figure 114 was altered to use radio buttons instead of edit fields for placing the *Sign Plate* in a desired position (compare to Figure 91, which shows different kinds of user input for quite the same functionality as in application AVL InMotion).

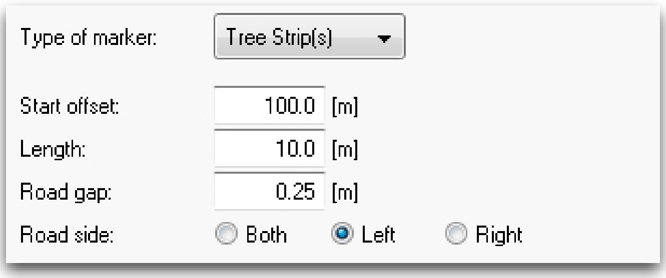


Figure 113: Design of Configuration Settings for Marker Tree Strip(s)

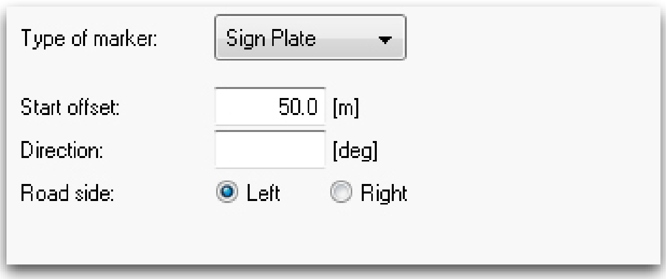


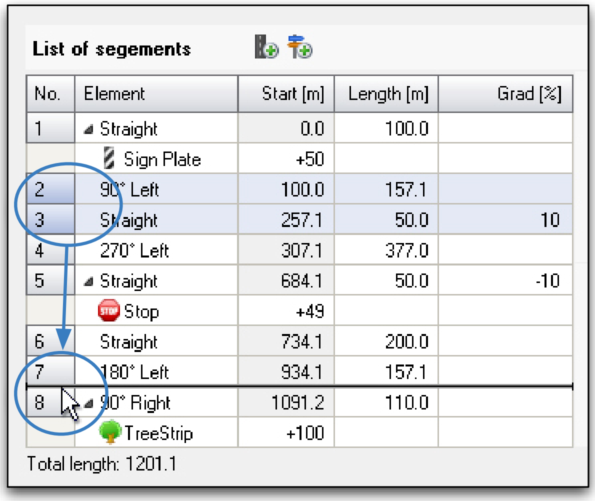
Figure 114: Design Recommendation for Configuration Settings of Type Sign Plate

[17]Employee of AVL List GmbH and at the same time also the secondary supervisor of this master's thesis.

5.3.1.2. Modification of Segments in Road Register

Figure 115 visualizes the drag-and-drop operation mode in order to copy or move list entries within the segment list. R7 (see Subsection 5.2.2.7) describes the reason for the missing function. In Figure 116, the editing of specified information within the segment list is shown, which is a normal operation mode.

Furthermore, the various graphical representations of the different markers are shown once again, for instance by a sign, stop and tree symbol.

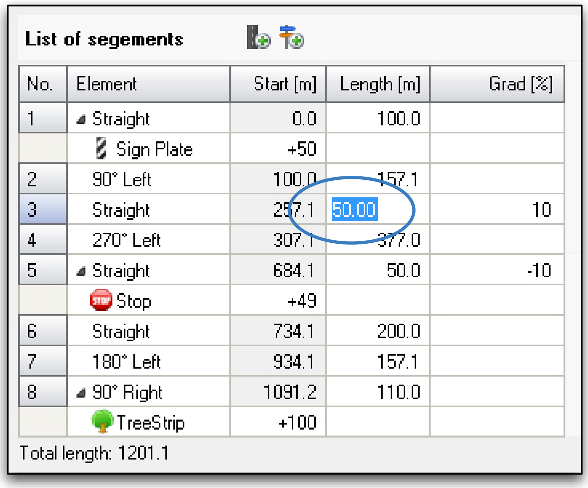


List of segements

No.	Element	Start [m]	Length [m]	Grad [%]
1	Straight	0.0	100.0	
	Sign Plate	+50		
2	90° Left	100.0	157.1	
3	Straight	257.1	50.0	10
4	270° Left	307.1	377.0	
5	Straight	684.1	50.0	-10
	Stop	+49		
6	Straight	734.1	200.0	
7	180° Left	934.1	157.1	
8	90° Right	1091.2	110.0	
	TreeStrip	+100		

Total length: 1201.1

Figure 115: The Drag-and-Drop Functionality



List of segements

No.	Element	Start [m]	Length [m]	Grad [%]
1	Straight	0.0	100.0	
	Sign Plate	+50		
2	90° Left	100.0	157.1	
3	Straight	257.1	50.00	10
4	270° Left	307.1	377.0	
5	Straight	684.1	50.0	-10
	Stop	+49		
6	Straight	734.1	200.0	
7	180° Left	934.1	157.1	
8	90° Right	1091.2	110.0	
	TreeStrip	+100		

Total length: 1201.1

Figure 116: Editing of Information within the Segment List

5.3.1.3. Improvement of Maneuver Setting

R13 describes a problem relating to the maneuver input fields (see Subsection 5.2.2.13). The suggested improvement is to change the label description from the duration definition (compare Figure 117 with Figure 118). An adaptation of the radio button mode is also suitable in replacing the edit fields, which is more transparent to the user. The maximum duration for maneuvers is to determine whether to use seconds or meters in the new design.

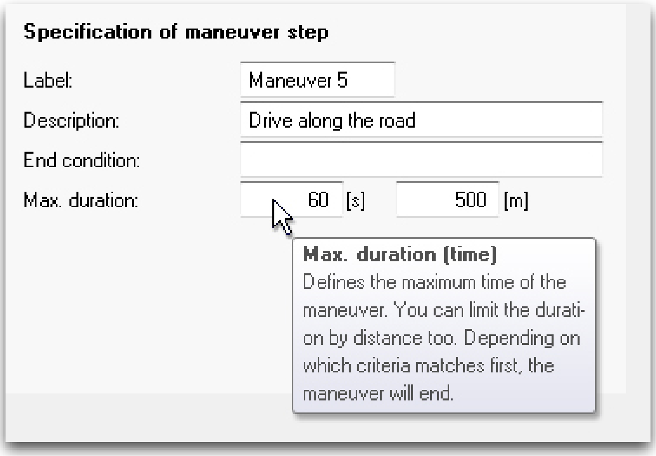


Figure 117: Current Design and Functionality of Maneuver Settings

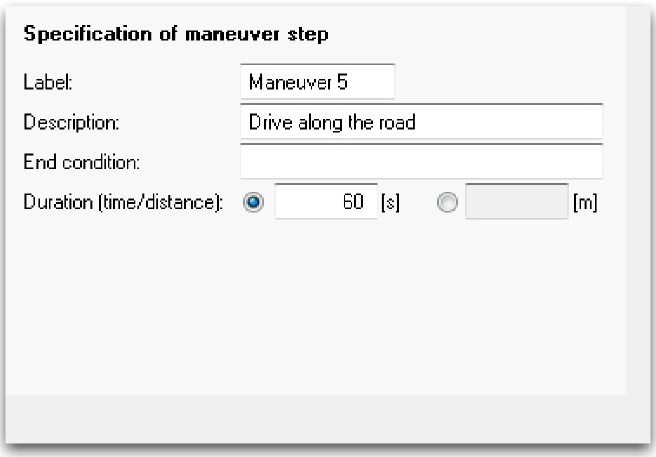


Figure 118: Suggested Improvement of Maneuver Settings

5.3.1.4. Redesign of Online View

Figure 119 shows the possible new online view in AVL PUMA. The solution, which is suggested in R15 (see Subsection 5.2.2.15), is the embedding of the three-dimensional simulation movie in the *Vehicle Status* dialog. The redesigned dialog displays the movie online, but no other movie settings. Moreover, all the non-relevant functionalities are hidden and are no longer visible to the user.

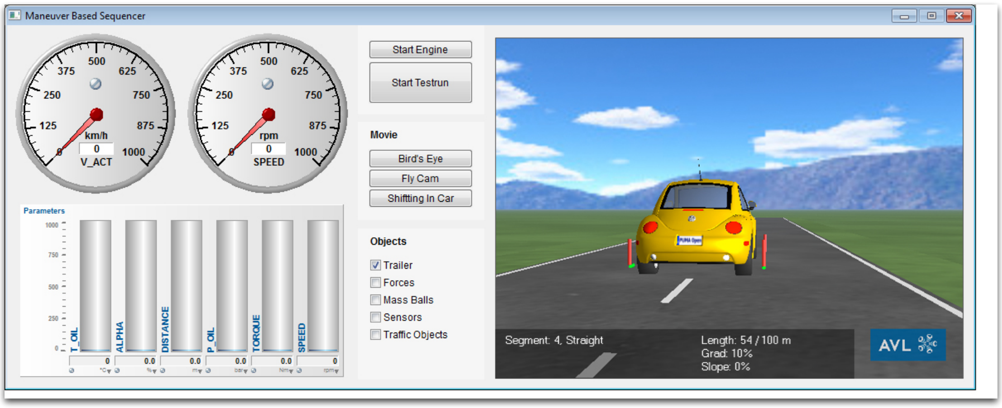


Figure 119: Redesigned Online View with Integrated 3D Simulation

Figure 120 shows an extract of the simulation movie, which is embedded in the online dialog of AVL PUMA. In this suggested dialog view, only the necessary data are clearly represented to the user. Information about the current segment, the length and other settings are displayed during the three-dimensional simulation of the test run.

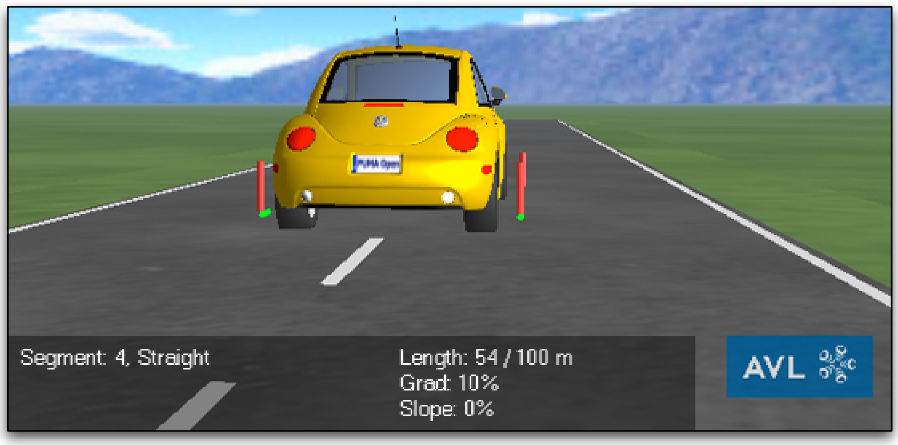


Figure 120: Enlarged Extract of Redesigned Online View

5.4. Evaluation of the Used Questionnaires

The following chapters present the analysis of the questionnaires answered in the usability test. The evaluated results are an indicator of system quality, and they provide important information about the various opinions of the subjects in respect of the usability of the system.

5.4.1. Attributes of Usability

In this section, Nielsen's attributes of usability (see Chapter 2.1) are evaluated. The characteristics of learnability, efficiency, memorability, error tolerability and satisfaction were analyzed using a specific questionnaire (see Appendix A.5.2), which contained a score of 0 (*Disagree*), 1, 2, 3, 4, or 5 (*Agree*) (the scale was reduced to five points) as well as *Not Applicable* (N/A). The test users chose N/A, if they were not able to make a statement about the assessment of the respective attribute.

Table 35 lists the retraceable data of each usability attribute that was rated by the test participants.

Attribute of Usability \ Task	TP1	TP3	TP4	TP6	TP10	TP2	TP5	TP7	TP8	TP9
Learnability	3	4	5	5	4	3	2	2	2	4
Efficiency	4	5	4	5	5	5	5	4	1	5
Memorability	4	4	5	5	4	5	3	4	2	3
Error Tolerability	N/A	4	5	5	4	4	2	2	N/A	4
Satisfaction	4	4	5	5	5	3	3	1	2	4

Table 35: Raw Data of the Usability Attributes Questionnaire
[Blue = Group **A** (AVL InMotion), Red = Group **B** (Standalone third-party application)]

With regard to the attributes of usability, the software component AVL InMotion achieved excellent results (see graphical illustration in Figure 121).

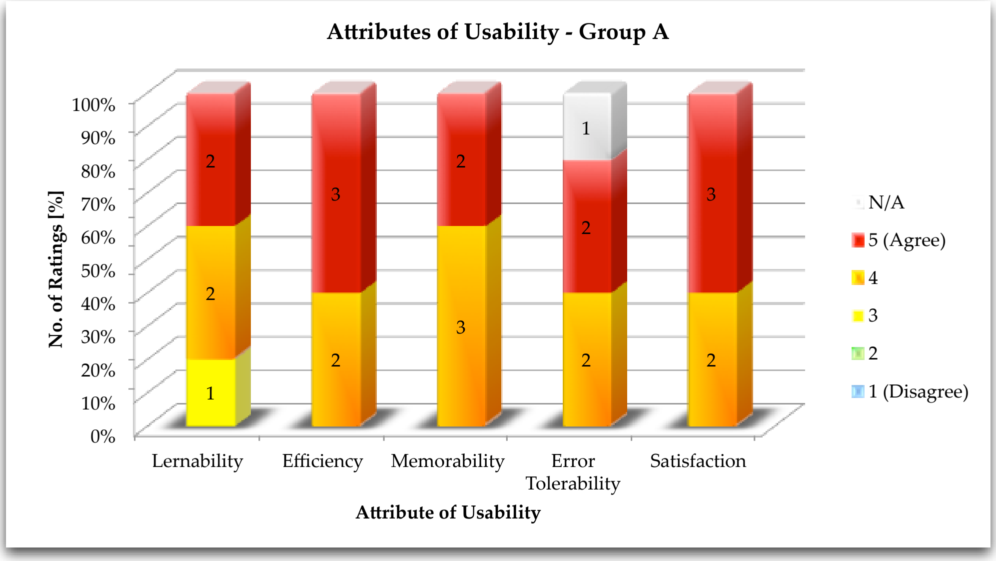


Figure 121: Group A - Rating of Attributes of Usability

Relating to the results of the standalone third-party application, the rating of each attribute of usability was slightly lower than the measurements of AVL InMotion (see graphical illustration in Figure 122).

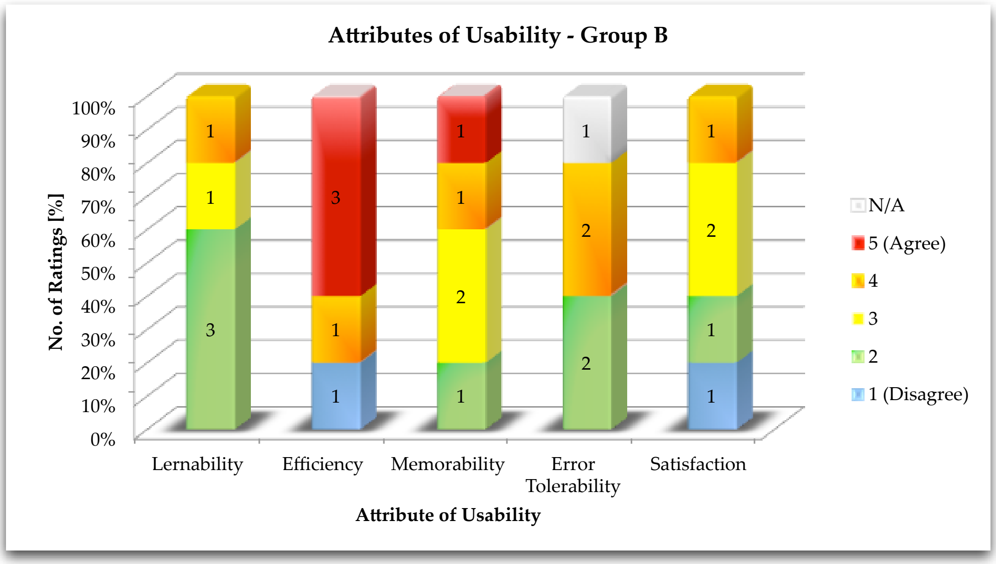


Figure 122: Group B - Rating of Attributes of Usability

The individual results of both test software systems relating to each of the five attributes of usability are presented in following Subchapter 5.4.1.1.

5.4.1.1. Comparison of Attributes of Usability

In the following subchapter, the ratings of both tested applications AVL InMotion (Group A) and the standalone third-party application (Group B) are compared.

Learnability

- Figure 123 shows the percentage of the rating of AVL InMotion. 40% of the test users in group **A** rated this product with a score of 5, 40% of this group valued it with a score of 4 and a further 20% classified it with the score of 3.
- Relating to the valuation of the standalone third-party application, 20% of the test users in group **B** evaluated this product with a scoring of 4, 20% of this group rated it with a score of 3 and 60% of the test users in group **B** assessed the system with the score of 2 (see Figure 124).

It is interesting that the ratings of learnability of the standalone third-party application included no score of 5. In contrast, the majority of test users of AVL InMotion agreed with the specified attribute that the system is easy to get to know and users are able to work productively after a short training period.

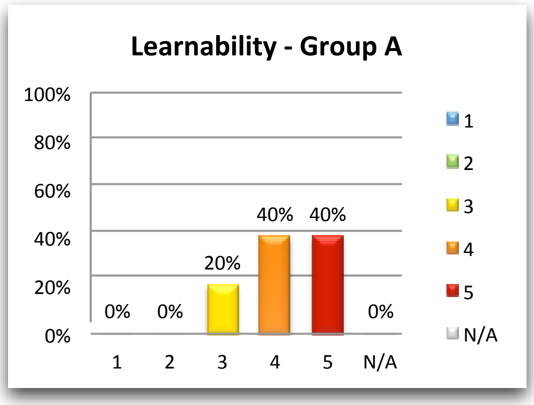


Figure 123: Gr. A - Learnability

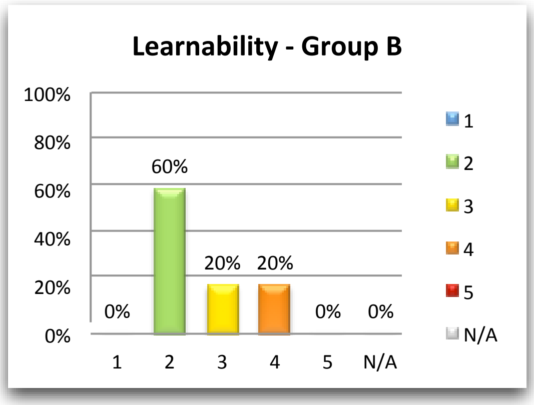


Figure 124: Gr. B - Learnability

Efficiency

- The efficiency of AVL InMotion was rated with a score of 5 by 60% of the test users in group **A** and 40% of this group valuated the system with a scoring of 4 (see Figure 125).

- Figure 126 displays the percentage of the evaluation of the standalone third-party application. 60% of the test users in group **B** rated the system with a score of 5, 20% of this group rated it with a score of 4 and further 20% of the test participants in group **B** valued the system with a score of 1.

The majority of both user groups rated the criterion efficiency with the highest score of 5.

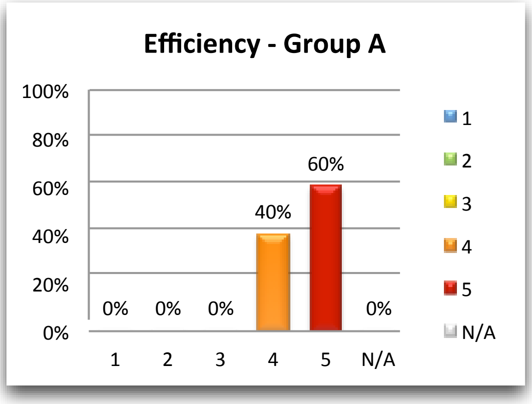


Figure 125: Gr. A - Efficiency

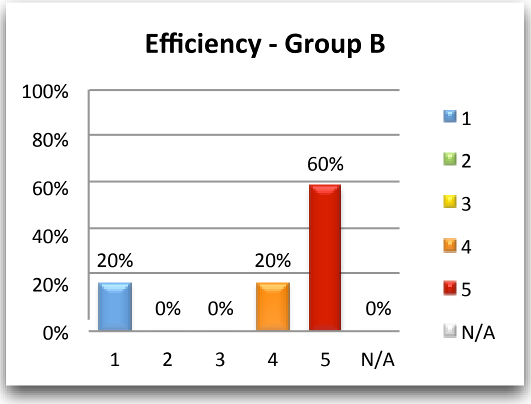


Figure 126: Gr. B - Efficiency

Memorability

- 40% of the test users in group **A** rated AVL InMotion with a score of 5 and 60% of this group rated the system with a score of 4 (see Figure 127).

- 20% of test group **B** rated the standalone third-party application with a score of 5, a further 20% of the test users of this group gave it a score of 4, 40% assessed the system with a score of 3 and 20% of this group rated it with a scoring of 2 (see Figure 128).

AVL InMotion achieved good results relating to the evaluation of the attribute memorability, because the ratings are between scores 4 and 5. In contrast, the test users of group **B** who rated the standalone third-party application had different opinions regarding the rating of memorability.

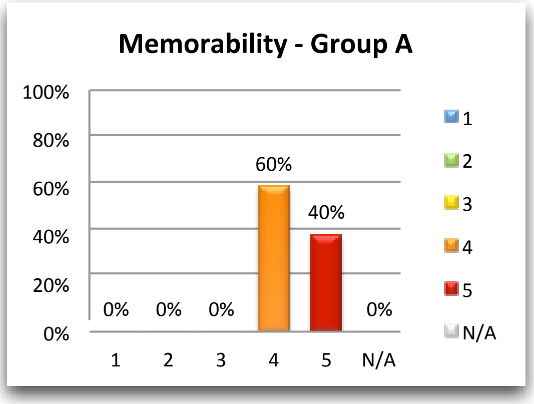


Figure 127: Gr. A - Memorability

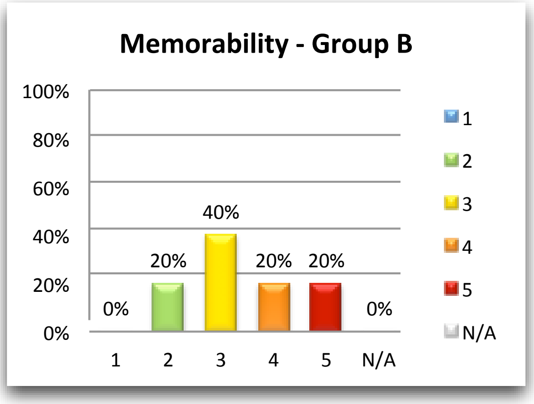


Figure 128: Gr. B - Memorability

Error Tolerability

- Figure 129 shows the rating of AVL InMotion relating to the usability attribute error tolerability. 40% of the test users in group **A** gave the system a score of 5, a further 40% in this group rated it with a score of 4 and 20% of the test users in group **A** selected the option N/A in rating this attribute of usability.
- 40% of the test users in group **B** rated the standalone third-party application with a score of 4, a further 40% of this group assessed it with a score of 2 and 20 % of the test group **B** selected the option N/A (see Figure 130).

20 % of each test group selected the option N/A when rating the usability attribute error tolerability. The majority of test users in group **A** who evaluated AVL InMotion agreed to the statement that the software prevented them from generating errors by using the system.

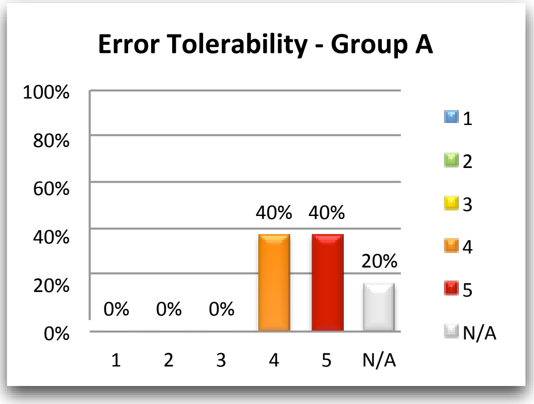


Figure 129: Gr. A - Error Tolerability

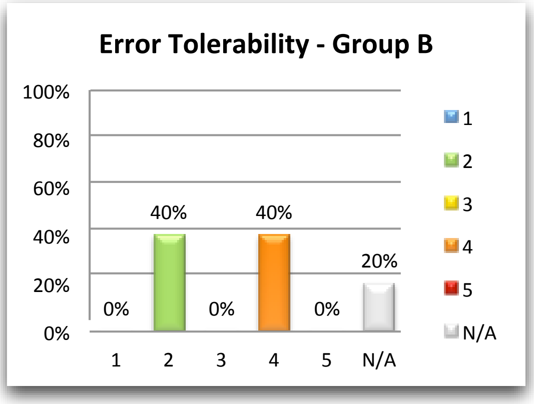


Figure 130: Gr. B - Error Tolerability

Satisfaction

- 60% of the test users in group **A** who evaluated AVL InMotion rated it with a score of 5 and 40 % of this group gave this system a scoring of 4 (see Figure 131).
- The standalone third-party application was rated with score of 4 by 20% of test group **B**, 40% of this group rated it with a score of 3, 20% gave the system a scoring of 2 and a further 20% rated it with a score of 1 (see Figure 132).

The test users in group **A** were satisfied with the usage of the system. In contrast, the test users in group **B** who rated the standalone third-party application had different opinions.

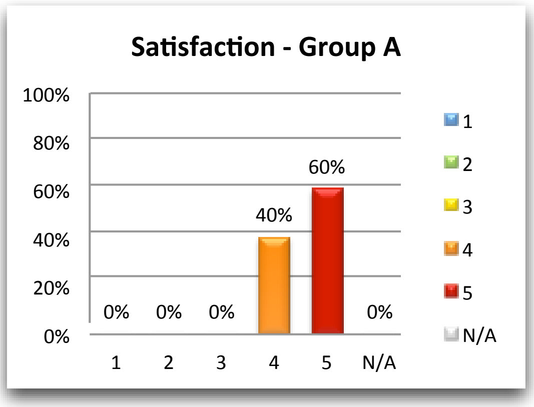


Figure 131: Gr. A - Satisfaction

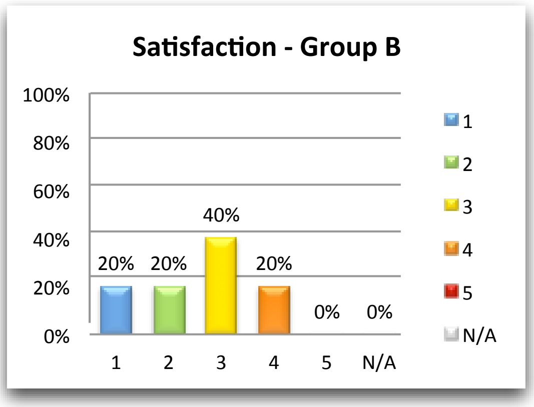


Figure 132: Gr. B - Satisfaction

5.4.2. Reactions of the System and Aspects of Satisfaction

Another questionnaire used in this usability study was the Post-Study System Usability Questionnaire (PSSUQ), which was developed at IBM (Lewis, 1995). This questionnaire consisted of 19 items and contained a rating scale of 0 *(Disagree), 1, 2, 3, 4* to 5 *(Agree)* (the scale was reduced to 5 points) as well as *Not Applicable*.

By utilizing the PSSUQ, there is an opportunity to identify the reactions of the subjects concerning the used system, and also to obtain information regarding their levels of satisfaction. Table 36 lists the retraceable data of each individual item of the PSSUQ that was rated by the participants, whereby the N/A represents the PSSUQ rating scale *not applicable*. The test users chose N/A, if they were not able to make a statement about the assessment of the respective attribute. Subsequent statistics and charts are created with the values in this table.

No.	Question	TP1	TP3	TP4	TP6	TP10	TP2	TP5	TP7	TP8	TP9
Q1	Overall, I am satisfied with how easy it is to use this system	4	3	3	1	4	3	4	2	3	4
Q2	It is simple to use this system	3	4	3	1	3	4	2	1	3	3
Q3	I can effectively complete my work using this system	4	4	3	1	N/A	3	3	N/A	2	N/A
Q4	I am able to complete my work quickly using this system	4	3	5	1	N/A	3	3	N/A	2	N/A
Q5	I am able to efficiently complete my work using this system	3	4	4	1	N/A	3	2	N/A	N/A	N/A
Q6	I feel comfortable using this system	4	3	4	1	4	3	2	2	2	4
Q7	It was easy to learn to use this system	4	5	5	1	4	4	3	N/A	2	5
Q8	I believe I became productive quickly using this system	4	4	5	1	4	4	3	N/A	1	4
Q9	The system gives error messages that clearly tell me how to fix problems	3	3	1	1	2	3	2	2	3	4
Q10	Whenever I make a mistake using the system , I recover easily and quickly	3	4	5	1	3	4	N/A	1	3	4
Q11	The information (such as online help, on-screen messages, and other documentation) provided with this system is clear	2	2	4	1	1	N/A	1	N/A	N/A	4
Q12	It is easy to find the information I needed	3	4	4	1	2	N/A	2	2	2	N/A
Q13	The information provided for the system is easy to understand	3	4	5	1	3	N/A	3	N/A	N/A	4
Q14	The information is effective in helping me complete the tasks and scenarios	3	5	4	1	N/A	N/A	N/A	N/A	N/A	N/A
Q15	The organization of information on the system screens is clear	4	4	5	2	4	4	1	3	2	N/A
Q16	The interface of this system is pleasant	5	3	5	1	4	2	1	2	2	4
Q17	I like using the interface of this system	4	4	5	1	4	2	2	2	3	3
Q18	This system has all the functions and capabilities I expect it to have	4	4	5	5	N/A	4	4	N/A	N/A	2
Q19	Overall, I am satisfied with this system	4	4	4	1	4	4	3	N/A	2	4

Table 36: Raw Data of the PSSUQ
[Blue = Group **A** (AVL InMotion), Red = Group **B** (Standalone third-party application)]

Figure 133 displays an overview of the PSSUQ rating of AVL InMotion with regard to the system usability. The detailed evaluation result of the standalone third-party application is displayed in Figure 134.

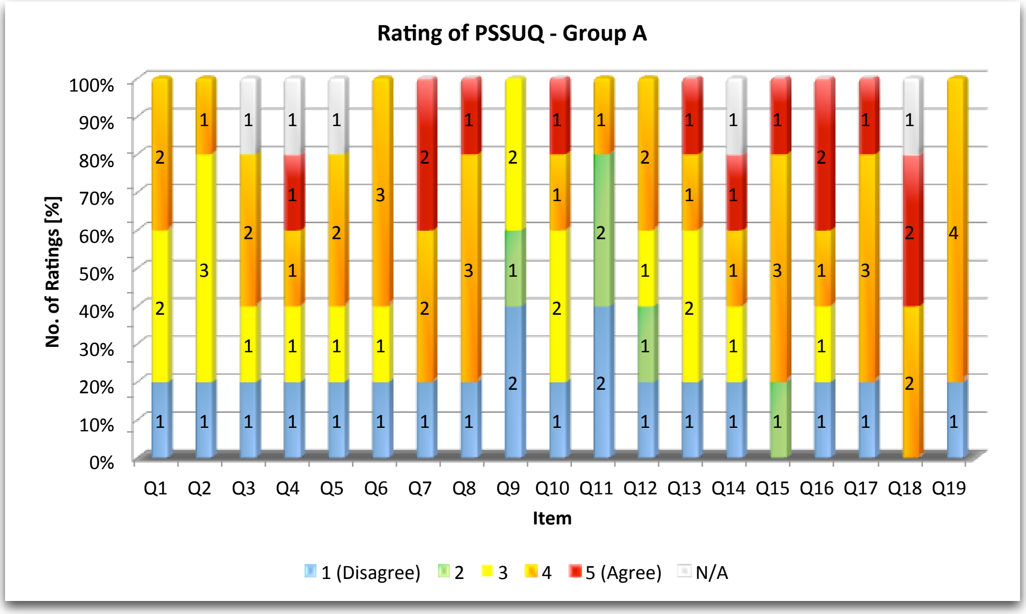


Figure 133: Group A - Rating of PSSUQ

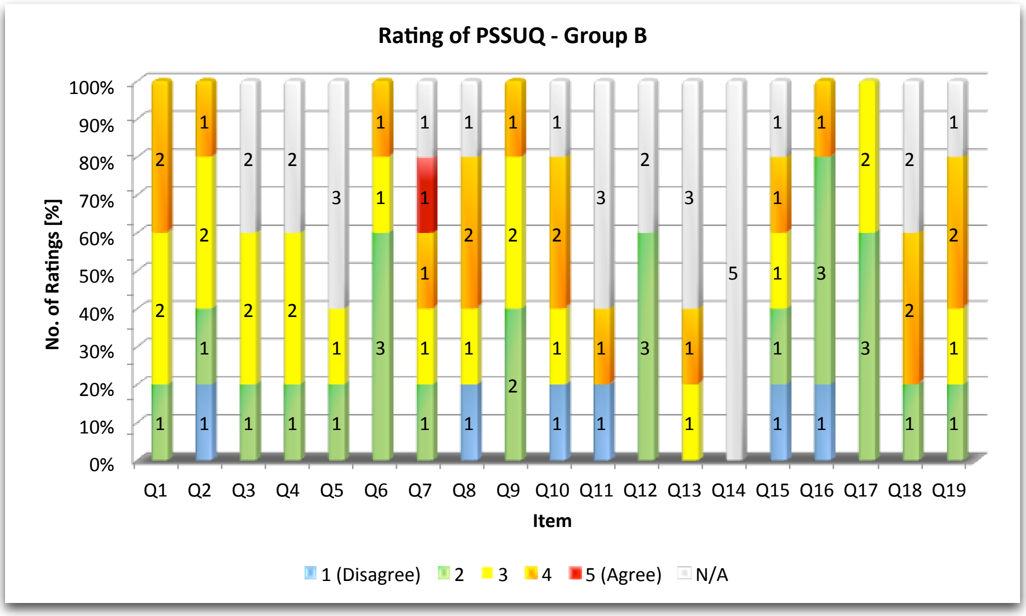


Figure 134: Group B - Rating of PSSUQ

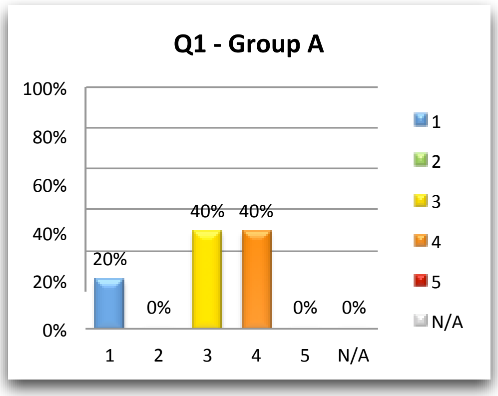
Subchapter 5.4.2.1 presents an overview of an individual analysis of specifically selected PSSUQ items.

5.4.2.1. Individual Analysis of PSSUQ Items

With regard to the previous overall result presentation, the following selected ratings are interesting due to the valuation of the two tested software systems AVL InMotion and the standalone third-party application:

Q1 *Overall, I am satisfied with how easy it is to use this system.*

- With regard to the rating of AVL InMotion, 40% of the test group **A** rated this specified question with a score of 4, a further 40% of this group gave the system a score of 3 and 20% of the test users in group **A** rated it with a score of 1.
- Relating to this statement, the standalone third-party application is rated with a score of 4 by the 40% of the test users in group **B**, 40% of this group evaluated it with a score of 3 and 20% rated it with a score of 2.
- Both tested software systems were rated similarly concerning their system usability and its ease of use. It is interesting that both ratings included no score of 5.
- See Figure 135 for the individual assessment of item 1.



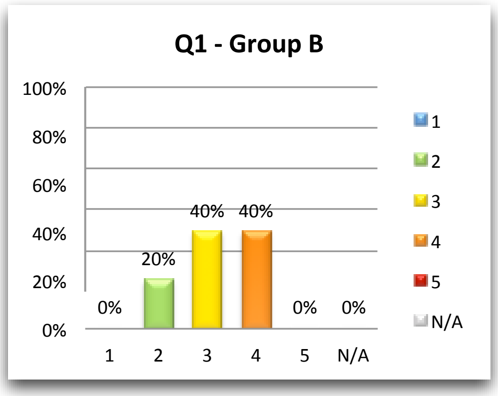


Figure 135: Individual Rating - PSSUQ Item 1

Q4 *I was able to complete the tasks and scenarios quickly using this system.*

- The opinions strongly varied concerning the quick completion of tasks and scenarios by the test participants in test group **A**, who tested the software component AVL InMotion. Half of the test users agreed with the statement, half of them did not. 20% of this group rated this statement with a score of 5, a further 20% of this group gave it a score of 4, 20% with a score of 3, 20% rated it with a score of 1 and 20 % of this group selected the option N/A.
- 40% of the test users in test group **B** who tested the standalone third-party application rated the usage of the software with a score of 3. 20% of this group gave the system a score of 3 when rating the completion of the tasks and scenarios quickly. 40% of user group **B** provided no information about this specified question.
- See Figure 136 for the individual assessment of item 4.

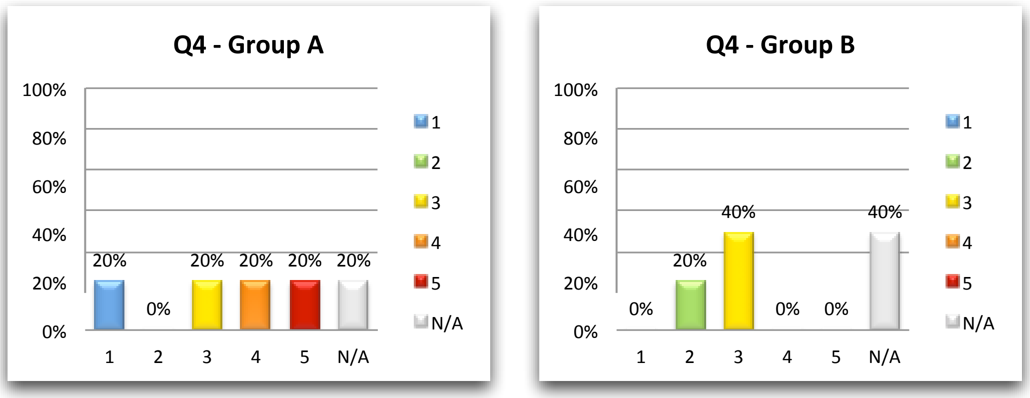
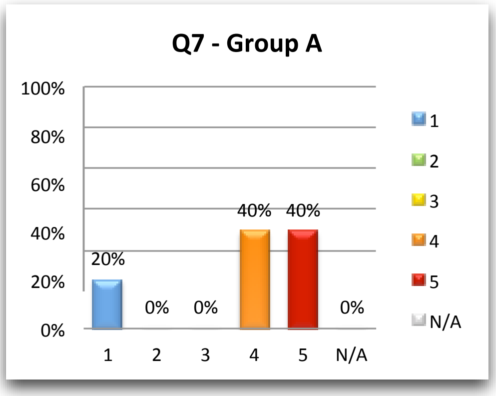


Figure 136: Individual Rating PSSUQ Item 4

Q7 *It was easy to learn to use this system.*

- 40% of user group **A** who tested AVL InMotion rated this item with a score of 5, a further 40% gave it a score of 4 and 20 percent of the users rated it with a scoring of 1.
- 20% of the test users in group **B** who evaluated the standalone third-party application rated this statement with a score of 5, a further 20% with a score of 4, 20% with a score of 3, 20% rated the this specified item with a score of 2 and 20 % of the test group **B** selected the option N/A.
- The opinions strongly varied concerning the learnability to use the system: Most test users confirmed that this software was relatively easy to handle and get familiar with. Only 20% of the test participants in group **A** and 40% in test group **B** disagreed with this statement, or provided no information about this specified question.
- See Figure 137 for the individual assessment of item 7.



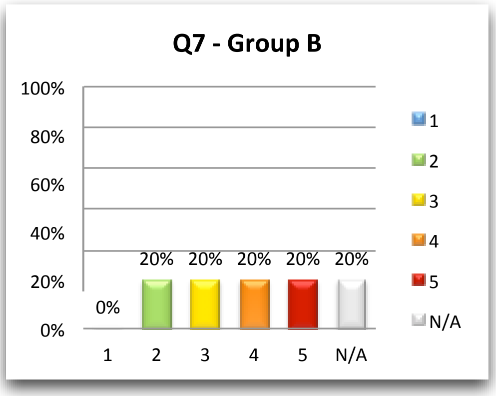


Figure 137: Individual Rating PSSUQ Item 7

Q9 *The system gave error messages that clearly told me how to fix problems.*

- 40% of the test users in group **A**, who tested the software component AVL InMotion, rated the system with a score of 3 relating to its provided error messages. 20% of this group gave this statement a score of 2 and a further 40% with a score of 1.
- This specified item was rated with a score of 4 by 20% of the test users in group **B** who evaluated the standalone third-party application. 40% of the test users in group **B** rated this statement with a score of 3 and a further 40% gave it a score of 2.
- The test participants were only moderately satisfied with respect to the use of error messages provided by both tested systems. Concerning the evaluation of both software systems, the standalone third-party application was rated a little better than AVL InMotion. Accordingly, the test users of AVL InMotion were not satisfied and there is a need to redesign the handling and presentation of error messages.
- See Figure 138 for the individual assessment of item 9.

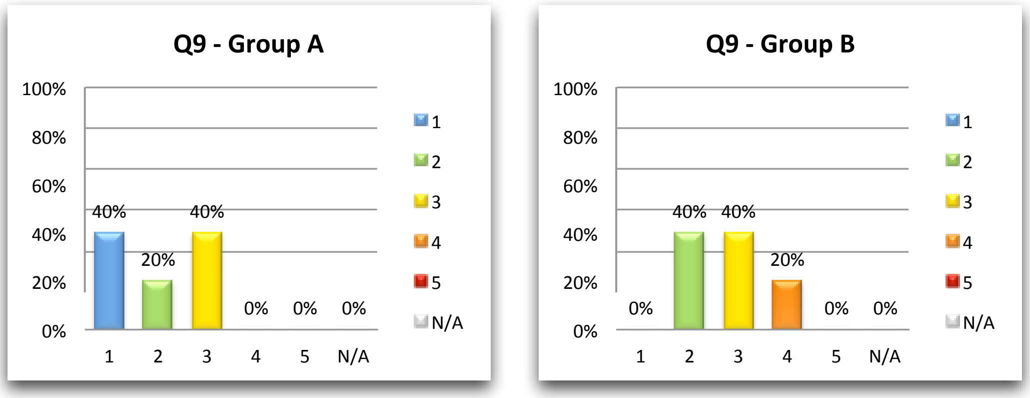


Figure 138: Individual Rating - PSSUQ Item 9

Q14 *The information was effective in helping me complete the tasks and scenarios.*

- 20% of the test users in group **A**, who tested AVL InMotion, rated this question with a score of 5, a further 20% with a score of 4, 20% with a score of 3, 20% with a score of 1 and 20 % of this group selected the option N/A.
- This statement was rated as not applicable by 100% of test group **B**, who evaluated the standalone third-party application. In this case, it was unanimously agreed that the information such as online help, on-screen messages and other documentation material did not provide enough assistance to the test users to complete the tasks and scenarios efficiently.
- In contrast to AVL InMotion, the majority of the test users in group **A** were satisfied regarding the support materials provided in order to solve the tasks and scenarios effectively.
- See Figure 139 for the individual assessment of item 14.

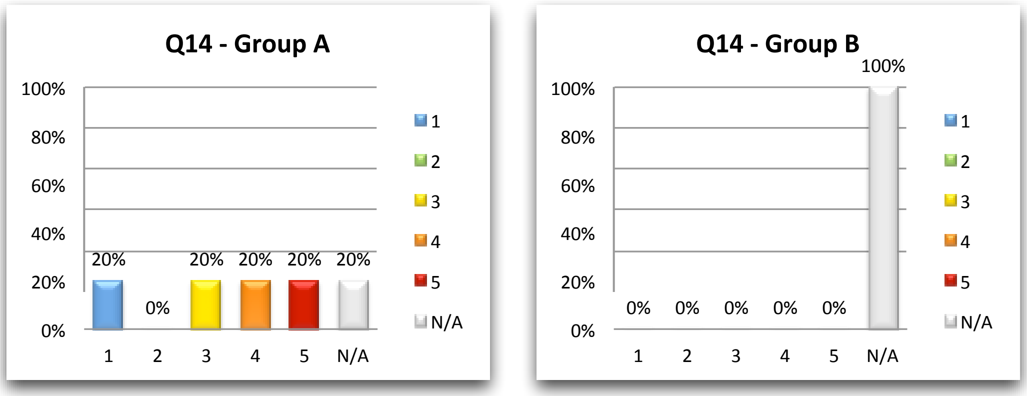


Figure 139: Individual Rating - PSSUQ Item 14

Q17 *I liked using the interface of this system.*

- 20% of the test users in group **A** rated the interface of AVL InMotion with a score of 5. 60% of this group gave the UI a score of 4 and a further 20% with a score of 1.

- The interface of the standalone third-party application was assessed with a score of 3 by 40% of the test users in group **B**. 60% of this group rated it with a score of 2.

- The opinions varied concerning the rating of the user interface of the tested applications: Most test users confirmed that the usage of the UI of AVL InMotion was enjoyable. Only 20% of the test users in group **A** completely disagreed with this statement. In contrast to test group **B**, the test participants did not like the current user interface of the standalone third-party application that much.

- See Figure 140 for the individual assessment of item 17.

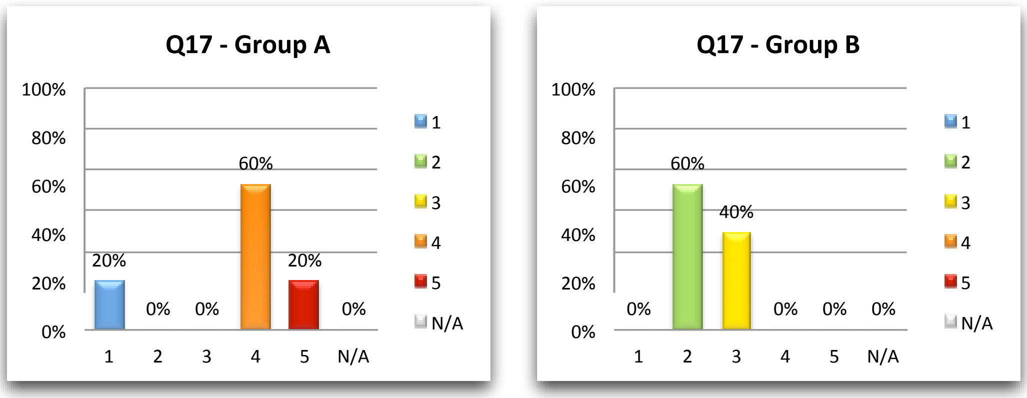


Figure 140: Individual Rating - PSSUQ Item 17

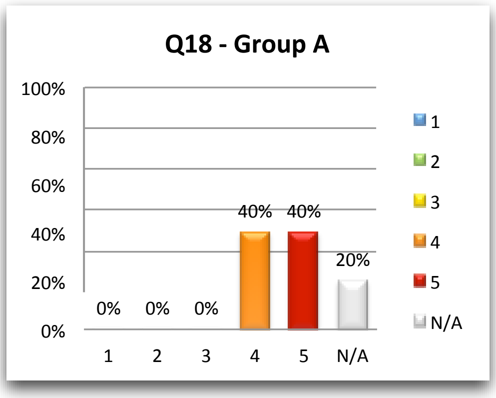
Q18 *This system has all the functions and capabilities I expect it to have.*

- 40% of the test users in group **A** rated this question with a score of 5 and a further 40% valued it with a score of 4. The option N/A was selected by 20% of this test group.

- 40% of the test users in group **B**, who evaluated the standalone third-party application, rated this statement with a score of 4 and a further 20% with a score of 2. 40% of this group selected the option N/A and provided no information about the specified question.

- The majority of the test users in group **A** rated with higher scores. This means, that the tested software has got a large range of functions, which the test users were expecting of this system.
  In contrast to the other tested software system, the opinions of the test users in group **B** strongly varied concerning the range of functions and capabilities of the application. 40% of this group were satisfied with the functional scope of the standalone third-party application and 20% of this group were missing some more functions.

- See Figure 141 for the individual assessment of item 18.



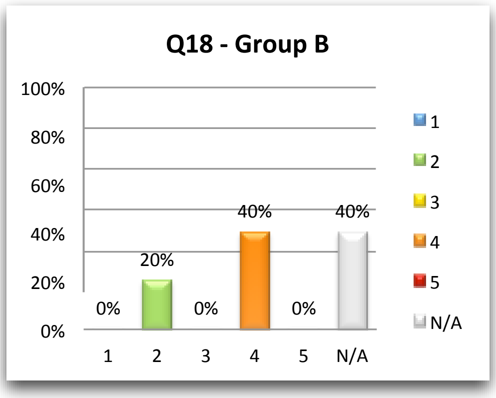


Figure 141: Individual Rating - PSSUQ Item 18

Q19 *Overall, I am satisfied with this system.*

- 80% of the test users in group **A** who rated AVL InMotion, were satisfied with the overall system usability and rated it with a score of 4. The other 20% of this group gave the system a score of 1.
- The satisfaction of the standalone third-party application was rated with a score of 4 by 40% of the test users in group **B**. 20% of this group gave it a score of 3, a further 20% rated it with a scoring of 2 and 20% of this group provided no information about this specified question.
- See Figure 142 for the individual assessment of item 19.

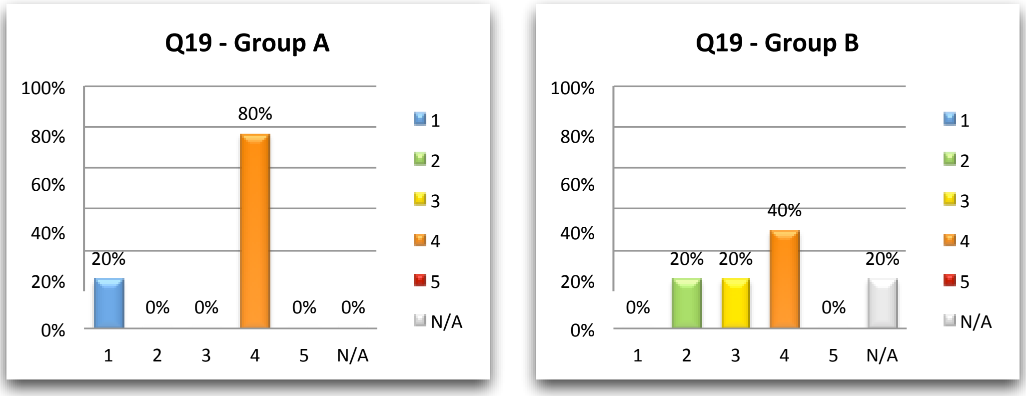


Figure 142: Individual Rating - PSSUQ Item 19

5.4.3. The Components' Pragmatic and Hedonic Quality

This chapter deals with the evaluation of the attractiveness of both tested software components using the AttrakDiff™ questionnaire. As already described in Chapter 2.1.2.3, the evaluation of both pragmatic and hedonic quality provides results of the total score of the appeal. This evaluation technique affords a preliminary contribution to the measurement of quality aspects whose impact goes beyond the normal usability.

The selected evaluation mode was the *Comparison of two Products* in order to analyze the different views of two user groups, which tested two different software applications. After completion of the AttrakDiff™ study by all participants, a report was prepared automatically, which included all the evaluation data and an interpretation of results. Next, the key results and main differences of both software components are summarized. It is important to mention that the following figures are taken from the AttrakDiff™ report. Later on, a few quotations directly from this expert assessment will be mentioned.

5.4.3.1. Average Values and Rating of the User Interface

Figure 143 shows a representation of the hedonic quality on the vertical axis and the pragmatic quality on the horizontal axis. The average values of the dimensions (represented by the characters **A** and **B**) are drawn in the portfolio. In an ideal situation, a classification is located on the right hand side at the top of the picture. This result would lead to the positive conclusion that both a high pragmatic and hedonic quality is achieved. In contrast to the previously described constellation, a placement would be very negative on the left hand side at the bottom. The rectangles, which are also shown in the figure, are the respective confidence rectangles[18]. These give statistical statements on the magnitude of the variance, relating

Figure 143: Average Values of the Dimensions & the Confidence Rectangles

[18]The orange rectangle is assigned to AVL InMotion, the blue rectangle refers to the standalone third-party application.

to specified evaluations between the test participants. In the following evaluations, the variances are marked widely. This is remarkable, because in these special cases, the number of test users was too low which leads to the assumption that the statements of the test participants diverged with a high degree of probability.

Considering the results of the AttrakDiff™ portfolio, the user interface of the tested software component *AVL InMotion* was rated as *"fairly self-oriented"* by the test users (refer to character **A** in portfolio). The respective confidential interval in the portfolio differed very widely, and so it is to be noted that the user is assisted and stimulated by the product, but the value of pragmatic as well hedonic quality only achieved average ratings. One result of the evaluation implies that a potential for improvement concerning the usability and hedonic aspects of the application AVL InMotion exists.

In contrast, the test participants rated the user interface of the second and standalone third-party application as "neutral" (see character **B** in portfolio). The main part of this analysis is that AVL InMotion performed better than the standalone application. The pragmatic quality and hedonic quality were evaluated more highly, whereby the difference in qualities for both tested systems is statistically insignificant.

5.4.3.2. Mean Values of the four AttrakDiff™ Dimensions

The next statistical analysis is based on the average values of the AttrakDiff™ dimensions of the evaluated applications. Figure 144 displays the mean values of the four AttrakDiff™ dimensions, namely the pragmatic quality (PQ), attractiveness (ATT), hedonic quality - identity (HQ-I) and hedonic quality - stimulation (HQ-S). The mean values in the following graphs are illustrated with color. The orange nodes (graph line) are assigned to AVL InMotion (Project part **A**) whereas the blue nodes (graph line) are assigned to the standalone third-party application (Project part **B**).

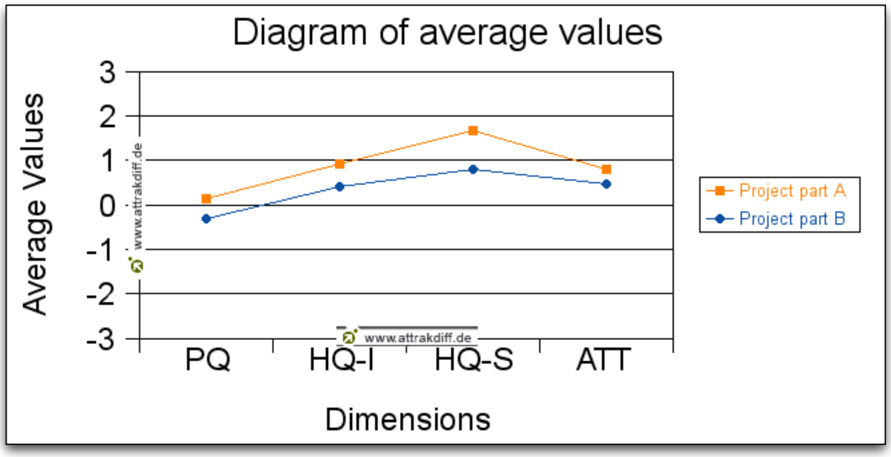


Figure 144: Mean Values of the AttrakDiff™ Dimensions

Concerning the pragmatic quality, AVL InMotion is located in the average region and just about meets ordinary standards. In the case of AVL List requiring the support of the user, the aim is to improve the existing solution. Furthermore, the hedonic quality - identity is located in the average region too. This means that the integrated component provides the user with identification and thus meets ordinary standards. Yet in the case of binding the user more strongly to the simulation software, AVL List must concentrate on improvement. In contrast to previous qualities, the hedonic quality - stimulation is located in the above-average region. In other words, AVL InMotion stimulates users, causes curiosity and motivates them. In terms of aspects of stimulation, the system is classified optimally. However, the value of attractivity is located in the average region again, which roughly means that the overall impression of the component is moderately attractive.

With regard to the second tested component, the external third-party application, the evaluation is nearly the same as the analysis of the component AVL InMotion. The only difference to the previously described result is the hedonic quality - stimulation, which is located in the average region and meets ordinary standards. Ultimately, there is still potential to improve the standalone application in order to motivate, enthrall and stimulate users even more intensely.

5.4.3.3. Summary

Summarizing the conclusions for the mean values of the four AttrakDiff™ dimensions, AVL InMotion performs better on all tested dimensions than the standalone third-party

application, whereby the difference of all dimensions for both tested applications is statistically insignificant.
Finally, Figure 145 shows the AttrakDiff™ word pairs represented with their mean values. The extreme values, whose characteristics are especially critically valued or rated very well are particularly interesting. For example, these include the word pairs technical - human, unprofessional - professional or unimaginative - creative; thereby a statement can be made about that the integrated application being technical, professional and creative in the view of the users, which supports a positive assessment of the component AVL InMotion.

In addition, Figure 145 also marks the seven semantic differentials of PQ, HQ-I, HQ-S and ATT.

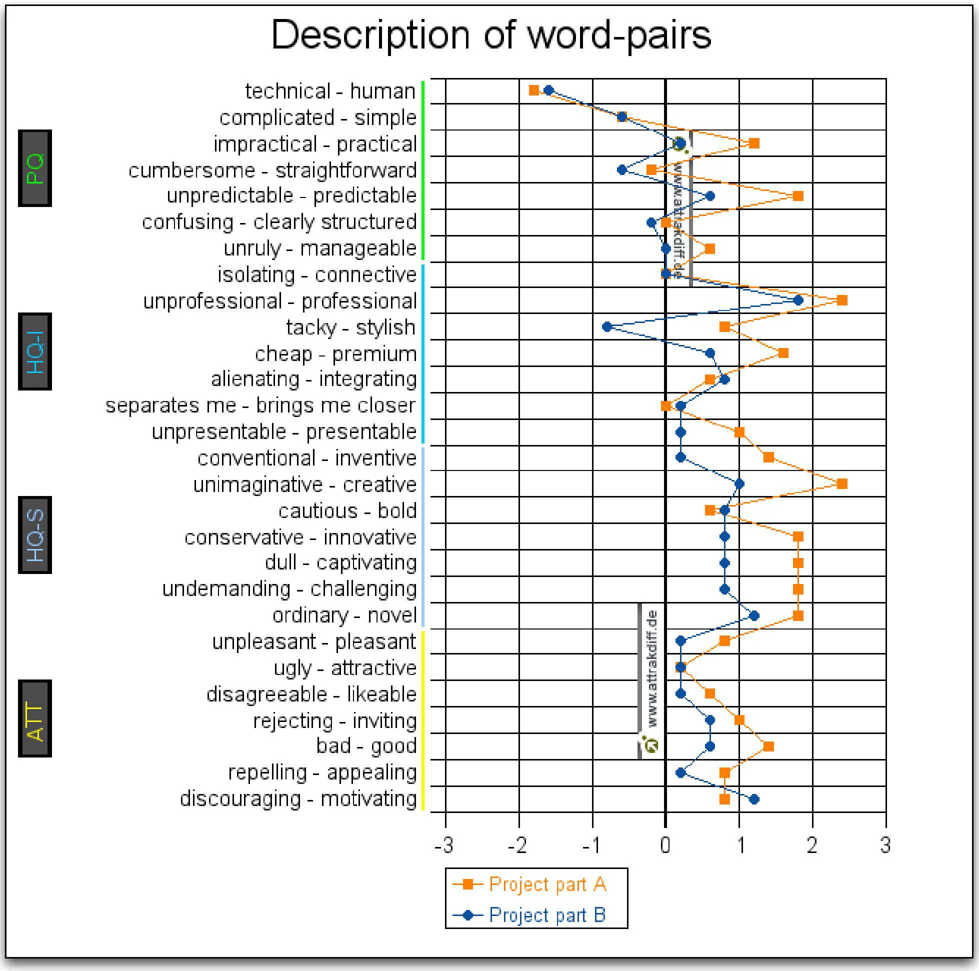


Figure 145: Mean Values of the AttrakDiff™ Word Pairs

6. Lessons Learned

In research, usability pioneers and specialists emphasize the importance of usability testing in order to improve the usability of a software system. Software project managers or product owners need to integrate usability techniques (such as usability evaluations, measurements and iterative design) in the process of software engineering. However, in common practice there is quite a difference to recommendations in literary studies.

The number of usability deniers in software development teams is still very high - especially in the case of non-designers, who create the required design in accordance with their own conceptions. This constitutes an even bigger barrier to acceptance and taking into account any usability aspects. In most cases, these non-designers try to avoid usability tests. In the worst case scenario, the findings of such usability evaluations could mean that all their work was a waste of time in terms of meeting the users' expectations. It is often hard for software engineers to understand why end-users have serious problems with the usage of their developed software systems. Yet in truth, this results, for example, from a badly designed user interface, which additionally requires a good knowledge of the user manual as precondition. Another reason for avoiding usability activities might be that there is not enough time for comprehensive usability tests: Either these evaluations are not scheduled from the beginning, or the usability tests are canceled during the project cycle because managers have to meet the deadlines. Another reason for the refusal of usability in software development is lacking the financial means. A usability budget is often rarely included in project costs, because some members of the management assume that including usability in developing software is too expensive. This leads to the assumption that the financial scope influences the decision maker in a software project.

The usability test conducted within the context of this master's thesis proves the importance of usability in software developing at AVL. Before carrying out this usability test, the test users were not particularly enthusiastic and it was not obvious to them, how the usability test would benefit the development of software projects. After their participation, they found such a usability study to be useful. The test subjects realized that usability studies could provide great value in developing usable software and that usability tests often identified information on how to resolve end-users' problems. It is important to mention that some of the test users were developers of AVL software systems. The conducted usability study gave them a comprehensive and objective insight into the usage of their own developed software. It is quite clear that through this usability study, the usability awareness was enhanced among this target group.

Concerning further lessons learned from this conducted usability test, the number of test subjects was not sufficient for the common use of standardized questionnaires. Literature recommends the involvement of at least 30 test users or rather end-users. This is an indispensable condition to obtain a reliable statement about the subjective satisfaction of the users via questionnaires. Due to the fact that the main task of gathering reasonable usability recommendations was already achieved with five THA tests in each test group, a larger number of test users was not taken into account for the inspection of the usability of this software component. Moreover, in a practice-orientated enterprise like AVL List GmbH, however, the allocation of a sufficient number of employees to participate in usability tests, which are combined with specified test methods (THA, field observation or questionnaires, for example), is almost impossible. In general, the cost factor as well as the amount of work of potential test users are reasons why carrying out usability tests is difficult in practice. This is one of the lessons learned from conducting this usability study.

Another issue relates to the development of prototypes and providing them at the correct time; the usage of this technique is a particularly attractive approach in gaining a quick and cost-efficient impression of the possible design structure of the UI. The original concept of this usability study was to conduct a comparable usability test with the standalone third-party application and a lo-fi prototype of AVL InMotion. Yet as the preliminary version of the user interface of AVL InMotion was created and demonstrated to AVL stakeholders, the implementation and technical development followed immediately, without any inspections of the UI design relating to various usability criteria. For this reason, the planned usability test was conducted with a hi-fi prototype, which was the latest build of the software component AVL InMotion, instead of using the prepared lo-fi prototype.

After the completion of the usability test, several issues were identified concerning the usage and design of AVL InMotion. This led to the conclusion that it is not always beneficial to present a prototype to responsible parties without including a user perspective. Although solutions have to be developed in time and within budget constraints, it is far more complex and takes more effort to implement the usability issues after the software product has already been developed.

7. Conclusions

In this master's thesis, the objective was to analyze the target system AVL InMotion, create a design for its graphical user interface and perform a comparative usability study with the usage of several test methods in order to gather adequate usability recommendations for an improvement of this specific software component. Empirical usability tests are one of the important techniques in detecting problems with the UI of a software system. Nowadays, the development of well-designed user interfaces is increasingly becoming a topic. It is a fact that the satisfaction of customers is crucial for an enterprises competition. A conclusion about the results and identified issues will now be provided.

This master's thesis has addressed the hypothesis (see Chapter 4.5) that the integration of a third-party software component into a larger system improves its usability for users familiar with the existing system. The hypothesis was confirmed by the evaluation of the results of the used test methods applied to the two software systems that were compared. The users in group **A**[19] acted as though they were more familiar with the system than the test subjects in group **B**[20]. These test users often had difficulties in finding the approach for solving the particular test task. With regard to the comparison of the task completion rate, the test users of group **A** mastered the tasks with fewer problems than the test users in group **B**. It also needs to be highlighted that the majority of group **B** took longer to accomplish the tasks compared to the same tasks completed by group **A.**

On the one hand, AVL InMotion includes only the features summarized in Chapter 4.1.2, which describes the overview of the target functionality of AVL InMotion. It is definitely an advantage if only selected system functions of a third-party software application are used for the development of the new system. The complexity decreases and the graphical user interface is easier to adapt. On the other hand, the standalone third-party application contains its full functionality which made it more difficult for group **B** to complete the tasks sufficiently.

The effects relating to this usability test of AVL InMotion resulted in a summary of issues and possible design solutions based on the experience of the test participants.

[19] Users who tested the extended software system with the integrated software component AVL InMotion
[20] Users who performed the test with the standalone third-party application

These main results were presented to the project manager and developers who were stunned by the number of insightful findings based on the study. At the same time, however, it is worth emphasizing that the design and usage of AVL InMotion performed better than those of the standalone third-party simulation software.

Such software used in the automotive sector, must be primarily as effective and efficient as possible in handling and operation. However, an additional fundamental question is: Does the software component AVL InMotion need to be attractive or is the functionality sufficient enough? The purpose of the questionnaire AttrakDiff™ was to analyze to what extent the perceived attractiveness played a role in relation to the choice of a software component, if various design concepts existed. With regard to pragmatic quality, a significant potential for the improvement of AVL InMotion exists. Hedonic quality is clearly evaluated more positively due to the stimulation and motivation of the user. Altogether, the objective is to make various functions of the integrated software component AVL InMotion even more attractive in order to be more competitive in the future.

Regarding the attributes of usability (learnability, efficiency, memorability, error tolerability and satisfaction), AVL InMotion achieved excellent results for each individual attribute and, the assessments of efficiency and satisfaction achieved particularly high scores. Yet if these rated attributes are compared with the results provided through the PSSUQ, a deviation is shown with respect to the statements of the test subjects. The classification and characterization of specified items in PSSUQ are more accurate and the test users rated these questions with a stricter seriousness. The survey on these questions emphasized, however, that on average the test participants were satisfied with the usage and design of AVL InMotion. However, there is still a need for investigation of improvements to enhance overall usability.

The usability study, which was conducted within this master's thesis, provides an essential input for all further activities of the development of AVL InMotion. Based on the results and feedback of the test groups, it would also be a big improvement to perform usability tests in other projects at AVL List GmbH. Usability studies verify the systems from the user's point of view and provide certainty that the developed software will be tailored to suit the demands of the automotive market. Iterative design and the involvement of user experience ensure usability, as shown in this master's thesis, and focus on developing user interfaces, which are easy to use.

8. Outlook and Future Work

Many recommendations are presented in this master's thesis relating to an improved user interface design of AVL InMotion. A further step would to redesign AVL InMotion as proposed in Chapter 5.3 followed by a re-inspection of the redesigned graphical user interface through a usability test. This aims at renewed qualitative assessment and usability evaluation that shows if the redesigned GUI would lead to a greater user satisfaction, especially in comparison to the initial version.

Another additional task would be to test the current version of AVL's testbed solution with the integrated software component AVL InMotion directly on the customers' premises. This technique is known as a field test, which is briefly described in Chapter 2.2.3.1. Such on-site field studies gather information about end-users by using the system to be tested under real conditions. Field observation presents a good opportunity to capture customer feedback, requirements and requests in order to develop improved solutions for them.

Relating to usability studies in large companies, one interesting aspect would be the involvement of software product/project managers in addition to the software developers as shown in this master's thesis in particular. Of course, as described earlier, representative test users (those are usually end-users) have to be selected in order to evaluate the level of how easy a software product is to use. Product managers are usually in a sales executive position and it is quite uncommon for them to review and rate the developed software systems. However, it is important to find out if the usability awareness among the responsible managers increases if they conduct a usability test themselves. Based on such accumulated experiences, the upper-management would pay more attention (and would invest a larger part of the budget) in usability activities within the software development process at the company.

Future work could also focus on the investigation of whether an attractive and good design of the user interface increases the usability of a software system. The GUI development of AVL InMotion is currently based on `Tcl`[21], which is an open-source script language. Tcl supports graphical application programming by means of the widget toolkit

[21] Tool Command Language

Tk, which provides an element library for building a GUI. In the current version of AVL InMotion, the standard widget sets of the Tk library are used, which include outdated GUI elements.

Future work could emphasize the use of the libraries of Tile (additional package of the release Tcl/Tk 8.4) and Ttk (standard package in version Tcl/Tk 8.5) in order to enable the development of a native appearance of graphical user interfaces. Tk 8.5 provides a broad variety of new features and some of these functions[22] are described in Table 37.

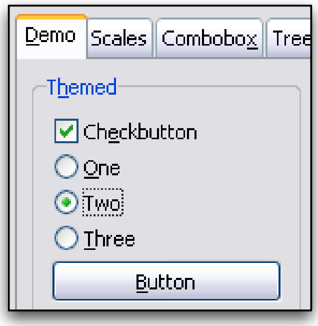


Figure 146: Xpnative Theme XP Silver [from SourceForge]

By using these innovative packages, the building of modern and contemporary looks of Tcl/Tk applications is possible. For example, the usage of the **xpnative** built-in theme[23] is an interesting feature, which makes use of the visual styles of Windows XP (see Figure 146).

New modern theming engine:	New and complementary widgets that make use of platform-specific theming on Mac OS X and Windows to better fit in with those environments, and feature an improved look and feel under X11.
New widgets:	Part of the themed widget set, Tk now has core notebook, combobox, treeview and progressbar widgets.
text **widget:**	Smooth scrolling, widget peering, and improved procedures for counting and replacing text.
Font rendering:	Now uses anti-aliased text under X11, and a more modern text engine (ATSUI) on Mac OS X.
Additional improvements:	Window transparency, new fullscreen option for windows, enhancements to specific widgets and window layout, and more.

Table 37: New Features of Tk 8.5
[from Tcl Developer Xchange]

Other future research could focus on the integration of the MBS Editor of AVL InMotion into the GUI framework of AVL PUMA Open. For now, the third-party software component AVL InMotion is only embedded without using the company-owned

[22]Mentioned highlights are taken from http://www.tcl.tk/
[23]http://tktable.sourceforge.net/tile/

framework of AVL List. It is quite obvious that this idea would involve considerable amounts of extra time and a lot of effort for developing this change. However, this consideration offers the following advantages: Existing resources could be reused (user interface components) and the architecture of AVL's testbed software suite would fit together perfectly.

Finally, it might make sense to develop an AVL style guide for the further use of applications that are provided from third-party companies. The aim of such a style guide would be to achieve a uniform design that makes external applications compatible with AVL software products. It should be guaranteed that the developers of the third-party company adopt their software systems in order to meet the requirements defined by AVL.

List of Figures

List of Tables

A. Test Materials

A.1. Orientation Script

Based on: (Andrews, 2010)

Hi, my name is Olivia. I'll be working with you in today's session. [Stephan here will be observing]. We are here to test a software application and would like your help.

I will ask you to perform some typical tasks with the application. Do your best, but don't be overly concerned with results - the system is being tested, and not your performance.

[I conduct this study in context of my master thesis, and have no affiliation with the system whatsoever]. My only role here today is to discover the flaws and advantages of this system from your perspective. Don't act or say things based on what you think I might want to see or hear, I need to know what you really think.

So that we can better follow your trail of thoughts during the test, we will ask you to think aloud while you work. Please do ask questions at any time, but I may only answer them at the end of the session.

While you are working, I will be taking some notes. We will also be recording the session for the benefit to do some analyses about usability afterwards.

If you feel uncomfortable, you may stop the test at any time.

Do you have any questions?

If not, then let's begin by filling out a short background questionnaire and having you sign the nondisclosure agreement and consent form.

A.2. Background Questionnaire

Based on: (Andrews, 2010)

Date:	_____________	Test:	_____________
Time:	_____________	Test No.:	_____________

Thank you for participating in our test. Please answer the following questions:

Personal information	
Age	☐ < 20 years ☐ 20 - 29 years ☐ 30 - 39 years ☐ 40 - 49 years ☐ 50 - 59 years ☐ > 60 years
Gender	☐ Female ☐ Male
Do you use a sight aid when working on the computer?	☐ None ☐ Glasses ☐ Contact lenses ☐ Other:
Do you have any form of colour blindness?	☐ No ☐ Yes, _____________ ✍
Education Level Attained:	☐ Vocational training ☐ General qualification for university entrance ☐ University degree ☐ Doctorate
If you are studying or have studied, please describe your main area of study:	✍
Do you have any foreign language competence?	☐ English ☐ French ☐ Italian ☐ Spanish ☐ None ☐ Others:

Personal information	
Profession / Job position	✍
For how long have you been working in your current position?	☐ 0 - 1 year ☐ 1 - 3 years ☐ 3 - 6 years ☐ More than 6 years
For how long have you been using a computer?	☐ 0 - 1 year ☐ 1 - 3 years ☐ 3 - 6 years ☐ More than 6 years
At how many days a week on average do you spend on your computer?	☐ Less than 2 days ☐ 2 - 5 days ☐ More than 5 days
How long do you usually use a computer on average per day?	☐ Less than 2 hours ☐ 2 - 6 hours ☐ More than 6 hours
Select the experience level which matches best your computational skills:	☐ Amateur ☐ Advanced user ☐ Professional
Which kind of computer do you normally use?	☐ Microsoft Windows ☐ Apple Macintosh ☐ Unix
Have you participated in a usability study before?	☐ As a test user ☐ As part of the test team ☐ No
If yes, what kind of study was it?	✍

Experience with AVL product AVL PUMA Open

Do you use AVL PUMA Open? ☐ Yes ☐ No

*If previous answer is **Yes**, please fill out the following questions:*

Which of the following subcomponents of AVL PUMA Open do you use? How often do you utilize them?	☐ Application Desktop ☐ PAM ☐ AVL Explorer ☐ PUMA online ☐ BSQ ☐ Data Manager ☐ NED ☐ PUC	☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently
If so, in which way does the product support you at your daily work?	✍ . .	
For how long have you been using this product?	☐ 0 - 1 year ☐ 1 - 3 years ☐ 3 - 6 years ☐ More than 6 years	
How many days on average a week do you use this product?	☐ Less than 2 days ☐ 2 - 5 days ☐ More than 5 days	
How long do you usually use that product on average per day?	☐ Less than 2 hours ☐ 2 - 6 hours ☐ More than 6 hours	
Select the experience level which matches best your computational skill using this product:	☐ Amateur ☐ Advanced user ☐ Professional	
Which of the following support materials are available?	☐ Product description ☐ User manual ☐ Online documentation ☐ Tutorial ☐ Hotline ☐ Other: ✍	How often do you utilize them? ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently ☐ never ☐ on occasion ☐ frequently
What language-version of the application do you use?	✍	
How many people use this product?	☐ Unknown ☐ 0 - 1 ☐ 10 - 100 ☐ 100 - 1000 ☐ More than 1000	

Table 38: Background Questionnaire

A.3. Non-Disclosure and Consent Form

Adapted from: (Andrews, 2010)

Thank you for participating in our study. Please be aware that confidential information may be disclosed to you and that you must not reveal information that you learn during the course of your participation. In addition, audio and video recordings will be made of your session, to allow others who are not present to observe your session and benefit from your feedback.

Please read the statements below and sign where indicated. Thank you.

I agree that I will disclose no information about the study.

I understand that audio and video recordings will be made of my session.
I grant permission to use these recordings for teaching, analyses and research purposes.

Place:	______________	Date:	______________
Date of Birth:	______________		
Name:	______________	Signature:	______________

A.4. Data Collection Form

Adapted from: (Andrews, 2010)

Task	Elapsed Time	Observations

Table 39: Data Collection Form

A.5. Feedback Questionnaire

A.5.1. Post-Study System Usability Questionnaire

Adapted from: (Lewis, 1995)

1.	Overall, I am satisfied with how easy it is to use this system.	disagree	1	2	3	4	5	agree	not applicable
2.	It was simple to use this system.	disagree	1	2	3	4	5	agree	not applicable
3.	I could effectively complete the tasks and scenarios using this system.	disagree	1	2	3	4	5	agree	not applicable
4.	I was able to complete the tasks and scenarios quickly using this system.	disagree	1	2	3	4	5	agree	not applicable
5.	I was able to efficiently complete the tasks and scenarios using this system.	disagree	1	2	3	4	5	agree	not applicable
6.	I felt comfortable using this system.	disagree	1	2	3	4	5	agree	not applicable
7.	It was easy to learn to use this system.	disagree	1	2	3	4	5	agree	not applicable
8.	I believe I could become productive quickly using this system.	disagree	1	2	3	4	5	agree	not applicable
9.	The system gave error messages that clearly told me how to fix problems.	disagree	1	2	3	4	5	agree	not applicable

10.	Whenever I made a mistake using the system, I could recover easily and quickly.	disagree	1	2	3	4	5	agree	not applicable
11.	The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.	disagree	1	2	3	4	5	agree	not applicable
12.	It was easy to find the information I needed.	disagree	1	2	3	4	5	agree	not applicable
13.	The information provided for the system was easy to understand.	disagree	1	2	3	4	5	agree	not applicable
14.	The information was effective in helping me complete the tasks and scenarios.	disagree	1	2	3	4	5	agree	not applicable
15.	The organization of information on the system screens was clear.	disagree	1	2	3	4	5	agree	not applicable
16.	The interface of this system was pleasant.	disagree	1	2	3	4	5	agree	not applicable
17.	I liked using the interface of this system.	disagree	1	2	3	4	5	agree	not applicable
18.	This system has all the functions and capabilities I expect it to have.	disagree	1	2	3	4	5	agree	not applicable
19.	Overall, I am satisfied with this system.	disagree	1	2	3	4	5	agree	not applicable

Table 40: Post-Study System Usability Questionnaire

A.5.2. Attributes of Usability

Adapted from: (Nielsen, 1993)

For an explanation of the different attributes please refer to Chapter 2.1.

1.	Learnability	bad	1	2	3	4	5	good	not applicable
2.	Efficiency	bad	1	2	3	4	5	good	not applicable
3.	Memorability	bad	1	2	3	4	5	good	not applicable
4.	Error tolerability	bad	1	2	3	4	5	good	not applicable
5.	Satisfaction	bad	1	2	3	4	5	good	not applicable

Table 41: Attributes of Usability Questionnaire

A.5.3. AttrakDiff™

Adapted from: AttrakDiff[24]

human	○ ○ ○ ○ ○ ○ ○	technical
isolating	○ ○ ○ ○ ○ ○ ○	connective
pleasant	○ ○ ○ ○ ○ ○ ○	unpleasant
inventive	○ ○ ○ ○ ○ ○ ○	conventional
simple	○ ○ ○ ○ ○ ○ ○	complicated
professional	○ ○ ○ ○ ○ ○ ○	unprofessional
ugly	○ ○ ○ ○ ○ ○ ○	attractive
practical	○ ○ ○ ○ ○ ○ ○	impractical
likeable	○ ○ ○ ○ ○ ○ ○	disagreeable
cumbersome	○ ○ ○ ○ ○ ○ ○	straightforward
stylish	○ ○ ○ ○ ○ ○ ○	tacky
predictable	○ ○ ○ ○ ○ ○ ○	unpredictable
cheap	○ ○ ○ ○ ○ ○ ○	premium
alienating	○ ○ ○ ○ ○ ○ ○	integrating
brings me closer to people	○ ○ ○ ○ ○ ○ ○	separates me from people
unpresentable	○ ○ ○ ○ ○ ○ ○	presentable
rejecting	○ ○ ○ ○ ○ ○ ○	inviting
unimaginative	○ ○ ○ ○ ○ ○ ○	creative
good	○ ○ ○ ○ ○ ○ ○	bad
confusing	○ ○ ○ ○ ○ ○ ○	clearly structured
repelling	○ ○ ○ ○ ○ ○ ○	appealing
bold	○ ○ ○ ○ ○ ○ ○	cautious
innovative	○ ○ ○ ○ ○ ○ ○	conservative
dull	○ ○ ○ ○ ○ ○ ○	captivating
undemanding	○ ○ ○ ○ ○ ○ ○	challenging
motivating	○ ○ ○ ○ ○ ○ ○	discouraging
novel	○ ○ ○ ○ ○ ○ ○	ordinary
unruly	○ ○ ○ ○ ○ ○ ○	manageable

Table 42: AttrakDiff™ Questionnaire

[24] http://www.attrakdiff.de

Bibliography

Alrajeh, D., Russo, A., & Uchitel, S. (2006). Inferring operational requirements from scenarios and goal models using inductive learning. In *SCESM '06: Proceedings of the 2006 international workshop on Scenarios and state machines: models, algorithms, and tools* (pp. 29–36). New York, NY, USA: ACM. doi:10.1145/1138953.1138960. http://doi.acm.org/10.1145/1138953.1138960.

Andrews, K. (2010). Materials for Human-Computer Interaction Practicals. Graz University of Technology, Austria. Last Access on May 5th, 2010. http://courses.iicm.tugraz.at/hci/practicals/materials/en/.

Anwer, S. & Ikram, N. (2006). Goal oriented requirement engineering: A critical study of techniques. In *Software Engineering Conference, 2006. APSEC 2006. 13th Asia Pacific* (pp. 121–130).: IEEE Computer Society Press. doi:10.1109/APSEC.2006.38. http://dx.doi.org/10.1109/APSEC.2006.38.

Araujo, J., Whittle, J., & Kim, D.-K. (2004). Modeling and Composing Scenario-Based Requirements with Aspects. In *Proceedings of the Requirements Engineering Conference, 12th IEEE International* (pp. 58–67). Washington, DC, USA: IEEE Computer Society. doi:10.1109/RE.2004.32. http://portal.acm.org/citation.cfm?id=1018443.1022094.

Arhippainen, L. (2003). *Use and integration of third-party components in software development.* VTT Publications 489, Espoo: Technical Research Centre of Finland. doi:10.1.1.95.5346. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.95.5346.

Bertoa, M. F., Troya, J. M., & Vallecillo, A. (2006). Measuring the usability of software components. *J. Syst. Softw.*, 79, 427–439, doi:10.1016/j.jss.2005.06.026. http://dx.doi.org/10.1016/j.jss.2005.06.026.

Bevan, N. (1995a). Measuring usability as quality of use. *Software Quality Journal*, 4(2), 115–130, doi:10.1007/BF00402715. http://dx.doi.org/10.1007/BF00402715.

Bevan, N. (1995b). Usability is quality of use. In *Anzai & Ogawa (eds) Proc. 6th International Conference on Human Computer Interaction*, volume 20, Part 2 of *Advances in Human Factors/Ergonomics* (pp. 349–355).: Elsevier. doi:10.1016/S0921-2647(06)80241-8. `http://dx.doi.org/10.1016/S0921-2647(06)80241-8`.

Bevan, N. (1999). Quality in use: meeting user needs for quality. *Journal of Systems and Software*, 49(1), 89–96, Elsevier, doi:10.1016/S0164-1212(99)00070-9. `http://dx.doi.org/10.1016/S0164-1212(99)00070-9`.

Beyer, H. & Holtzblatt, K. (1999). Contextual design. *Interactions*, 6, 32–42, doi:10.1145/291224.291229. `http://doi.acm.org/10.1145/291224.291229`.

Blomkvist, S. (2002). Persona–an overview. *Extract from the paper The User as a personality. Using Personas as a tool for design. Position paper for the course workshop 'Theoretical perspectives in Human-Computer Interaction' at IPLab, KTH, September 3, 2002*, 22, 2004.

Brinkman, W., Haakma, R., & Bouwhuis, D. (2008). Component-specific usability testing. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 38(5), 1143–1155, doi:10.1109/TSMCA.2008.2001056. `http://dx.doi.org/10.1109/TSMCA.2008.2001056`.

Brown, S. & Holzinger, A. (2008). Low Cost Prototyping: Part 1, or How to Produce Better Ideas Faster by Getting User Reactions Early and Often. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2*, BCS-HCI '08 (pp. 213–214). Swinton, UK, UK: British Computer Society. `http://portal.acm.org/citation.cfm?id=1531826.1531895`.

Burmester, M., Hassenzahl, M., & Koller, F. (2002). Usability ist nicht alles–Wege zu attraktiven Produkten. *iCom*, 1(1), 32–40. Last access on Jan 7th, 2011. `http://www.marc-hassenzahl.de/pdfs/Usability%20ist%20nicht%20alles%20-%20I-Com%201%202002.pdf`.

Carter, J. (1999). Incorporating standards and guidelines in an approach that balances usability concerns for developers and end users. *Interacting with Computers*, 12(2), 179 – 206, doi:10.1016/S0953-5438(99)00011-9. `http://dx.doi.org/10.1016/S0953-5438(99)00011-9`.

Casaday, G. & Rainis, C. (1996). Requirements, models, and prototypes for HCI design. In *CHI '96: Conference companion on Human factors in computing systems* (pp. 361–

362). New York, NY, USA: ACM. doi:10.1145/257089.257373. http://doi.acm.org/10.1145/257089.257373.

Casas, R., Blasco Marín, R., Robinet, A., Delgado, A. R., Yarza, A. R., Mcginn, J., Picking, R., & Grout, V. (2008). User Modelling in Ambient Intelligence for Elderly and Disabled People. In *Proceedings of the 11th international conference on Computers Helping People with Special Needs*, ICCHP '08 (pp. 114–122). Berlin, Heidelberg: Springer-Verlag. doi:10.1007/978-3-540-70540-6_15. http://dx.doi.org/10.1007/978-3-540-70540-6_15.

Cheng, B. H. C. & Atlee, J. M. (2007). Research Directions in Requirements engineering. In *FOSE '07: 2007 Future of Software Engineering* (pp. 285–303). Washington, DC, USA: IEEE Computer Society. doi:10.1109/FOSE.2007.17. http://dx.doi.org/10.1109/FOSE.2007.17.

Cooper, A., Reimann, R., & Cronin, D. (2007). About Face 3.0: The essentials of interaction design. *Wiley Publishing, Inc., Indianapolis, Indiana*.

Crnkovic, I. (2003). Component-based software engineering-new challenges in software development. In *Information Technology Interfaces, 2003. ITI 2003. Proceedings of the 25th International Conference on* (pp. 9–18).: IEEE. doi:10.1109/ITI.2003.1225314. http://dx.doi.org/10.1109/ITI.2003.1225314.

Dahm, M. (2006). *Grundlagen der Mensch-Computer-Interaktion*. Muenchen, Germany: Pearson Studium.

Dellarocas, C. (1997). Toward a Design Handbook for Integrating Software Components. In *Assessment of Software Tools and Technologies, 1997., Proceedings Fifth International Symposium on* (pp. 3–13).: IEEE Computer Society. doi:10.1109/AST.1997.599906. http://dx.doi.org/10.1109/AST.1997.599906.

Drapala, J., Żatuchin, D., & Sobecki, J. (2010). Multidimensional data visualization applied for user's questionnaire data quality assessment. In *Proceedings of the 4th KES international conference on Agent and multi-agent systems: technologies and applications, Part I*, KES-AMSTA'10 (pp. 351–360). Berlin, Heidelberg: Springer-Verlag. 10.1007/978-3-642-13480-7_37. http://dx.doi.org/10.1007/978-3-642-13480-7_37.

Duechting, M., Zimmermann, D., & Nebe, K. (2007). Incorporating User Centered Requirement Engineering into Agile Software Development. *Human-Computer Interac-*

tion. Interaction Design and Usability, (pp. 58–67)., doi:10.1007/978-3-540-73105-4_7. http://dx.doi.org/10.1007/978-3-540-73105-4_7.

Ebner, M., Deutschmann, L., Zechner, J., & Holzinger, A. (2006). Qualitätssicherung durch Evaluation: Neue Ansätze zur Evaluierung selbstgesteuerter Lerntools. *Qualitätssicherung im E-Learning. Medien in der Wissenschaft*, 36, 99–107. Last Access on January 9th, 2011. http://user.meduni-graz.at/andreas.holzinger/holzinger%20de/papers%20de/E04_EBNER%20et%20al%20(2006)%20Evaluierung%20selbstgesteuerter%20Lerntools.pdf.

Ecklund, Jr., E. F., Delcambre, L. M. L., & Freiling, M. J. (1996). Change cases: use cases that identify future requirements. *SIGPLAN Not.*, 31, 342–358, doi:10.1145/236338.236372. http://doi.acm.org/10.1145/236338.236372.

Ferré, X., Juristo, N., Windl, H., & Constantine, L. (2001). Usability basics for software developers. *IEEE Softw.*, 18, 22–29, doi:10.1109/52.903160. http://dx.doi.org/10.1109/52.903160.

Gould, J. D. & Lewis, C. (1985). Designing for usability: key principles and what designers think. *Commun. ACM*, 28, 300–311, doi:10.1145/3166.3170. http://doi.acm.org/10.1145/3166.3170.

Grudin, J. (1988). Why CSCW applications fail: problems in the design and evaluation of organizational interfaces. In *CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work* (pp. 85–93). New York, NY, USA: ACM. doi:10.1145/62266.62273. http://doi.acm.org/10.1145/62266.62273.

Hassenzahl, M., Burmester, M., & Koller, F. (2003). Attrakdiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In *Mensch & Computer 2003: Interaktion in Bewegung* (pp. 187–196). Last Access on December 27th, 2010. http://www.attrakdiff.de/files/mc2003_hassenzahl_review.pdf.

Hassenzahl, M., Schöbel, M., & Trautmann, T. (2008). How motivational orientation influences the evaluation and choice of hedonic and pragmatic interactive products: The role of regulatory focus. *Interact. Comput.*, 20, 473–479, doi:10.1016/j.intcom.2008.05.001. http://dx.doi.org/10.1016/j.intcom.2008.05.001.

Holzinger, A. (2004). Rapid Prototyping for a Virtual Medical Campus Interface. *IEEE Softw.*, 21, 92–99, doi:10.1109/MS.2004.1259241. http://dx.doi.org/10.1109/MS.2004.1259241.

Holzinger, A. (2005). Usability engineering methods for software developers. *Communications of the ACM*, 48(1), 71–74, doi:10.1145/1039539.1039541. http://doi.acm.org/10.1145/1039539.1039541.

Holzinger, A. (2006). Thinking-aloud - eine Königsmethode im Usability Engineering. In *OCG Journal, Vol. 31, Nr. 1* (pp. 4–5). Last Access on December 28th, 2010. http://user.meduni-graz.at/andreas.holzinger/holzinger%20de/papers%20de/G49_HOLZINGER%20(2006)%20Thinking%20aloud.pdf.

Holzinger, A. (2010). *Process Guide for Students for Interdisciplinary Work in Computer Science/Informatics*. Norderstedt: Books on Demand GmbH.

Holzinger, A. & Brown, S. (2008). Low Cost Prototyping: Part 2, or How to Apply the Thinking-Aloud Method Efficiently. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 2*, BCS-HCI '08 (pp. 217–218). Swinton, UK, UK: British Computer Society. http://portal.acm.org/citation.cfm?id=1531826.1531897.

Hopkins, J. (2000). Component primer. *Commun. ACM*, 43(10), 27–30, doi:10.1145/352183.352198. http://doi.acm.org/10.1145/352183.352198.

Hornbæk, K. (2006). Current practice in measuring usability: Challenges to usability studies and research. *Int. J. Hum.-Comput. Stud.*, 64, 79–102, doi:10.1016/j.ijhcs.2005.06.002. http://dx.doi.org/10.1016/j.ijhcs.2005.06.002.

Hornbæk, K. & Frokjær, E. (2005). Comparing usability problems and redesign proposals as input to practical systems development. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 391–400). New York, NY, USA: ACM. doi:10.1145/1054972.1055027. http://doi.acm.org/10.1145/1054972.1055027.

Hsia, P., Davis, A., & Kung, D. (1993). Status report: requirements engineering. *Software, IEEE*, 10(6), 75 –79, doi:10.1109/52.241974. http://dx.doi.org/10.1109/52.241974.

ISO, E. (1998). 9241-11. Ergonomic requirements for office work with visual display terminals (VDTs)–Part 11: Guidance on usability. *International Organization for Standardization*.

Jani, R. & Schrepp, M. (2004). Influence of Accessibility Related Activities on the Usability of Business Software. *Computers Helping People with Special Needs*, (pp. 629–629)., doi:10.1007/978-3-540-27817-7_8. `http://dx.doi.org/10.1007/978-3-540-27817-7_8`.

Johnson, G., Clegg, C., & Ravden, S. (1989). Towards a practical method of user interface evaluation. *Applied Ergonomics*, 20(4), 255–260, doi:10.1016/0003-6870(89)90187-7. `http://dx.doi.org/10.1016/0003-6870(89)90187-7`.

Jokela, T. (2002). Making user-centred design common sense: striving for an unambiguous and communicative UCD process model. In *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction* (pp. 19–26). New York, NY, USA: ACM. doi:10.1145/572020.572024. `http://doi.acm.org/10.1145/572020.572024`.

Jokela, T., Iivari, N., Matero, J., & Karukka, M. (2003). The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In *CLIHC '03: Proceedings of the Latin American conference on Human-computer interaction* (pp. 53–60). New York, NY, USA: ACM. doi:944519.944525. `http://doi.acm.org/944519.944525`.

Junior, P. T. A. & Filgueiras, L. V. L. (2005). User modeling with personas. In *Proceedings of the 2005 Latin American conference on Human-computer interaction*, CLIHC '05 (pp. 277–282). New York, NY, USA: ACM. doi:10.1145/1111360.1111388. `http://doi.acm.org/10.1145/1111360.1111388`.

Keinonen, T. (2008). User-centered design and fundamental need. In *NordiCHI '08: Proceedings of the 5th Nordic conference on Human-computer interaction* (pp. 211–219). New York, NY, USA: ACM. doi:10.1145/1463160.1463183. `http://doi.acm.org/10.1145/1463160.1463183`.

Kramer, J., Noronha, S., & Vergo, J. (2000). A user-centered design approach to personalization. *Commun. ACM*, 43, 44–48, doi:10.1145/345124.345139. `http://doi.acm.org/10.1145/345124.345139`.

Larsson, S., Crnkovic, I., & Ekdahl, F. (2004). On the Expected Synergies between Component-Based Software Engineering and Best Practices in Product Integration. *EUROMICRO Conference*, 0, 430–436, doi:10.1109/EURMIC.2004.1333399. `http://dx.doi.org/10.1109/EURMIC.2004.1333399`.

Law, E. L.-C. & Hvannberg, E. T. (2004). Analysis of combinatorial user effect in international usability tests. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04 (pp. 9–16). New York, NY, USA: ACM. doi:10.1145/985692.985694. `http://doi.acm.org/10.1145/985692.985694`.

Law, L.-C. & Hvannberg, E. T. (2002). Complementarity and convergence of heuristic evaluation and usability test: a case study of universal brokerage platform. In *Proceedings of the second Nordic conference on Human-computer interaction*, NordiCHI '02 (pp. 71–80). New York, NY, USA: ACM. `http://doi.acm.org/10.1145/572020.572030`.

Lee, J. & Xue, N.-L. (1999). Analyzing user requirements by use cases: a goal-driven approach. *Software, IEEE*, 16(4), 92 –101, doi:10.1109/52.776956. `http://dx.doi.org/10.1109/52.776956`.

Lee, W. J., Cha, S. D., & Kwon, Y. R. (1998). Integration and analysis of use cases using modular Petri nets in requirements engineering. *Software Engineering, IEEE Transactions on*, 24(12), 1115 –1130, doi:10.1109/32.738342. `http://dx.doi.org/10.1109/32.738342`.

Letier, E. & van Lamsweerde, A. (2002). Deriving operational software specifications from system goals. In *SIGSOFT '02/FSE-10: Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering* (pp. 119–128). New York, NY, USA: ACM. doi:10.1145/587051.587070. `http://doi.acm.org/10.1145/587051.587070`.

Lewis, J. R. (1995). IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *Int. J. Hum.-Comput. Interact.*, 7(1), 57–78, doi:10.1080/10447319509526110. `http://dx.doi.org/10.1080/10447319509526110`.

Lin, H. X., Choong, Y.-Y., & Salvendy, G. (1997). A proposed index of usability: A method for comparing the relative usability of different software systems. *Behaviour & Information Technology*, (pp. 267–277). doi:10.1.1.126.4148. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.126.4148`.

Macleod, M. (1994). Usability in context: Improving quality of use. In *Proceedings of the International Ergonomics Association 4th International Symposium on Human Factors in Organizational Design and Management, Stockholm*: Citeseer. doi:10.1.1.113.1564. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.1564`.

Maguire, M. (2001). Context of Use within usability activities. *International Journal of Human-Computer Studies*, 55(4), 453 – 483, doi:10.1006/ijhc.2001.0486. http://dx.doi.org/10.1006/ijhc.2001.0486.

Mari, M. & Eila, N. (2003). The Impact of Maintainability on Component-based Software Systems. In *Proceedings of the 29th Conference on EUROMICRO* (pp. 25–). Washington, DC, USA: IEEE Computer Society. doi:10.1006/ijhc.2001.0486. http://dx.doi.org/10.1006/ijhc.2001.0486.

McNamara, N. & Kirakowski, J. (2006). Functionality, usability, and user experience: three areas of concern. *interactions*, 13(6), 26–28, doi:10.1145/1167948.1167972. http://doi.acm.org/10.1145/1167948.1167972.

Molich, R. & Dumas, J. S. (2008). Comparative usability evaluation (CUE-4). *Behav. Inf. Technol.*, 27, 263–281, doi:10.1080/01449290600959062. http://portal.acm.org/citation.cfm?id=1451129.1451137.

Molich, R. & Nielsen, J. (1990). Improving a human-computer dialogue. *Commun. ACM*, 33(3), 338–348, doi:10.1145/77481.77486. http://doi.acm.org/10.1145/77481.77486.

Niegemann, H. M. (2008). Usability. In *Kompendium multimediales Lernen*, X.media.press (pp. 419–453). Springer Berlin Heidelberg. doi:10.1007/978-3-540-37226-4_28. http://dx.doi.org/10.1007/978-3-540-37226-4_28.

Nielsen, C. M., Overgaard, M., Pedersen, M. B., & Stage, J. (2005). Feedback from Usability Evaluation to User Interface Design: Are Usability Reports Any Good? In *Human-Computer Interaction - INTERACT 2005*, volume 3585 of *Lecture Notes in Computer Science* (pp. 391–404). Springer Berlin Heidelberg. doi:10.1007/11555261_33. http://dx.doi.org/10.1007/11555261_33.

Nielsen, J. (1992). The usability engineering life cycle. *Computer*, 25(3), 12 –22, doi:10.1109/2.121503. http://dx.doi.org/10.1109/2.121503.

Nielsen, J. (1993). *Usability Engineering*. San Francisco, California: Morgan Kaufmann.

Nielsen, J. (1994). Heuristic evaluation. *In Nielsen, J. and Mack, R., (Eds.) Usability inspection methods*, (pp. 25–64). John Wiley and Sons, Inc. New York.

Nielsen, J. & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human* *factors in computing systems*, CHI '93 (pp. 206–213). New York, NY, USA: ACM. http://doi.acm.org/10.1145/169059.169166.

Nielsen, J. & Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90 (pp. 249–256). New York, NY, USA: ACM. http://doi.acm.org/10.1145/97243.97281.

Nielson, J. (1999). *Designing Web Usability.* New Riders Publishing, Indianapolis.

Nuseibeh, B. & Easterbrook, S. (2000). Requirements engineering: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering* (pp. 35–46). New York, NY, USA: ACM. doi:10.1145/336512.336523. http://doi.acm.org/10.1145/336512.336523.

Ozkaya, I. (2006). Representing Requirement Relationships. In *REV '06: Proceedings of the 1st international workshop on Requirements Engineering Visualization* (pp.$\tilde{3}$). Washington, DC, USA: IEEE Computer Society. doi:10.1109/REV.2006.4. http://dx.doi.org/10.1109/REV.2006.4.

Ozkaya, I. & Akin, O. (2006). Requirement-driven design: assistance for information traceability in design computing. *Design Studies*, 27(3), 381 – 398, doi:10.1016/j.destud.2005.11.005. http://dx.doi.org/10.1016/j.destud.2005.11.005.

Petrovic, K. & Weissenberger, U. (2007). User centered design als teil des requirements engineering prozesses. Last Access on April 21th, 2010. http://www6.hs-esslingen.de/static/354/Paper_UCD_als_Teil_des_RE.pdf.

Pleuss, A., Vitzthum, A., & Hussmann, H. (2007). Integrating Heterogeneous Tools into Model-Centric Development of Interactive Applications. In G. Engels, B. Opdyke, D. Schmidt, & F. Weil (Eds.), *Model Driven Engineering Languages and Systems*, volume 4735 of *Lecture Notes in Computer Science* (pp. 241–255). Springer Berlin / Heidelberg. 10.1007/978-3-540-75209-7_17. http://dx.doi.org/10.1007/978-3-540-75209-7_17.

Procaccino, J. D. & Verner, J. M. (2009). Software developers' views of end-users and project success. *Commun. ACM*, 52(5), 113–116, doi:10.1145/1506409.1506437. http://doi.acm.org/10.1145/1506409.1506437.

Rettig, M. (1994). Prototyping for tiny fingers. *Commun. ACM*, 37, 21–27, doi:http://doi.acm.org/10.1145/175276.175288. http://doi.acm.org/10.1145/175276.175288.

Rosenbaum, S., Bloomer, S., Rinehart, D., Rohn, J., Dye, K., Humburg, J., Nielsen, J., & Wixon, D. (1999). What makes strategic usability fail?: lessons learned from the field. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems* (pp. 93–94). New York, NY, USA: ACM. doi:10.1145/632716.632773. http://doi.acm.org/10.1145/632716.632773.

Ryan, C. & Gonsalves, A. (2005). The effect of context and application type on mobile usability: an empirical study. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, ACSC '05 (pp. 115–124). Darlinghurst, Australia: Australian Computer Society, Inc. http://portal.acm.org/citation.cfm?id=1082161.1082174.

Sauro, J. & Kindlund, E. (2005). A method to standardize usability metrics into a single score. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05 (pp. 401–409). New York, NY, USA: ACM. http://doi.acm.org/10.1145/1054972.1055028.

Schrepp, M., Held, T., & Laugwitz, B. (2006). The influence of hedonic quality on the attractiveness of user interfaces of business management software. *Interact. Comput.*, 18, 1055–1069, doi:10.1016/j.intcom.2006.01.002. http://dx.doi.org/10.1016/j.intcom.2006.01.002.

Seffah, A., Djouab, R., & Antunes, H. (2001). Comparing and reconciling usability-centered and use case-driven requirements engineering processes. In *AUIC '01: Proceedings of the 2nd Australasian conference on User interface* (pp. 132–139). Washington, DC, USA: IEEE Computer Society. doi:10.1109/AUIC.2001.906292. http://dx.doi.org/10.1109/AUIC.2001.906292.

Sharp, H., Finkelstein, A., & Galal, G. (1999). Stakeholder Identification in the Requirements Engineering Process. In *DEXA '99: Proceedings of the 10th International Workshop on Database & Expert Systems Applications* (pp. 387). Washington, DC, USA: IEEE Computer Society. doi:10.1109/DEXA.1999.795198. http://dx.doi.org/10.1109/DEXA.1999.795198.

Stone, D., Jarret, C., Woodroffe, M., & Minocha, S. (2005). *User interface design and evaluation.* Morgan Kaufmann.

Strohmaier, M., Yu, E., Horkoff, J., Aranda, J., & Easterbrook, S. (2007). Analyzing Knowledge Transfer Effectiveness–An Agent-Oriented Modeling Approach. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences* (pp. 188b). Washington, DC, USA: IEEE Computer Society. doi:10.1109/HICSS.2007.80. `http://dx.doi.org/10.1109/HICSS.2007.80`.

Sutcliffe, A. (1998). Scenario-based requirements analysis. *Requirements Engineering*, 3, 48–65, doi:10.1007/BF02802920. `http://dx.doi.org/10.1007/BF02802920`.

Sutcliffe, A., Maiden, N., Minocha, S., & Manuel, D. (1998). Supporting scenario-based requirements engineering. *Software Engineering, IEEE Transactions on*, 24(12), 1072 –1088, doi:10.1109/32.738340. `http://dx.doi.org/10.1109/32.738340`.

van Lamsweerde, A. (2001). Goal-Oriented Requirements Engineering: A Guided Tour. In *RE '01: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering* (pp. 249). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ISRE.2001.948567. `http://dx.doi.org/10.1109/ISRE.2001.948567`.

van Lamsweerde, A. (2008). Requirements engineering: from craft to discipline. In *SIGSOFT '08/FSE-16: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering* (pp. 238–249). New York, NY, USA: ACM. doi:10.1145/1453101.1453133. `http://doi.acm.org/10.1145/1453101.1453133`.

van Veenendaal, E. P. W. M. (1998). Questionnaire based usability testing. In *Proceedings of European Software Quality Week* (pp. 118–134). Brussels. Last Access on December 28th, 2010. `http://www.erikvanveenendaal.nl/UK/files/Questionnaire%20based%20usability%20testing.pdf`.

van Wyk, E. & de Villiers, R. (2008). Usability context analysis for virtual reality training in South African mines. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, SAICSIT '08 (pp. 276–285). New York, NY, USA: ACM. doi:10.1145/1456659.1456691. `http://doi.acm.org/10.1145/1456659.1456691`.

Wilding, C. (1998). Practical GUI screen design: making it usable. In *CHI '98: CHI 98 conference summary on Human factors in computing systems* (pp. 125–126). New York, NY, USA: ACM. doi:10.1145/286498.286574. `http://doi.acm.org/10.1145/286498.286574`.

Yu, E. S. K. (1997). Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In *RE '97: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering* (pp. 226). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ISRE.1997.566873. `http://dx.doi.org/10.1109/ISRE.1997.566873`.

Zave, P. (1997). Classification of research efforts in requirements engineering. *ACM Comput. Surv.*, 29(4), 315–321, doi:10.1145/267580.267581. `http://doi.acm.org/10.1145/267580.267581`.