A Comprehensive Model of Usability

Sebastian Winter, Stefan Wagner, and Florian Deissenboeck

Institut für Informatik
Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
{winterse, wagnerst, deissenb}@in.tum.de

Abstract. Usability is a key quality attribute of successful software systems. Unfortunately, there is no common understanding of the factors influencing usability and their interrelations. Hence, the lack of a comprehensive basis for designing, analyzing, and improving user interfaces. This paper proposes a 2-dimensional model of usability that associates system properties with the activities carried out by the user. By separating activities and properties, sound quality criteria can be identified, thus facilitating statements concerning their interdependencies. This model is based on a tested quality meta-model that fosters preciseness and completeness. A case study demonstrates the manner by which such a model aids in revealing contradictions and omissions in existing usability standards. Furthermore, the model serves as a central and structured knowledge base for the entire quality assurance process, e.g. the automatic generation of guideline documents.

Keywords: usability, quality models, quality assessment

1 Introduction

There is a variety of standards concerning the quality attribute *usability* or *quality in use* [1, 2]. Although in general all these standards point in the same direction, due to different intuitive understandings of usability, they render it difficult to analyze, measure, and improve the usability of a system. A similar situation also exists for other quality attributes, e.g. *reliability* or *maintainability*. One possibility to address this problem is to build a comprehensive model of the quality attribute. Most models take recourse to the decomposition of quality proposed by Boehm et al. [3]. However, this decomposition is still too abstract and imprecise to be used concretely for analysis and measurement.

More comprehensive models have been proposed for product quality in general [4] or even usability [5]. However, these models have three problems: First, they do not decompose the attributes and criteria to a level that is suitable for actually assessing them for a system. Secondly, these models tend to omit rationale of the required properties of the system. Thirdly, the dimensions used in these models are heterogeneous, e.g. the criteria mix properties of the system with properties of the user. The first problem constrains the use of these models as the basis for analyses. The second one makes it difficult to describe impacts precisely and therefore to

convince developers to use it. The third problem hampers the revelation of omissions and inconsistencies in these models. The approach to quality modeling by Broy, Deissenboeck, and Pizka [6] is one way to deal with these problems. Using an explicit meta-model, it decomposes quality into system properties and their impact on activities carried out by the user. This facilitates a more structured and uniform means of modeling quality.

Problem. Although usability is a key quality attribute in modern software systems, the general understanding of its governing factors is still not good enough for profound analysis and improvement. Moreover, currently there are no comprehensive objective criteria for evaluating usability.

Contribution. This paper proposes a comprehensive 2-dimensional model of usability based on a quality meta-model that facilitates a structured decomposition of usability and descriptions of the impacts of various facts of the system. This kind of model has proven to be useful for the quality attribute *maintainability* [6]. Several benefits can be derived by using this type of model:

1. The ability to reveal omissions and contradictions in current models and guidelines.
2. The ability to generate guidelines for specific tasks automatically.
3. A basis for (automatic) analysis and measurement.
4. The provision of an interface with other quality models and quality attributes.

We demonstrate the applicability of the 2-dimensional model in a case study of the ISO 15005 [7] which involves domain-specific refinements. By means of this model we are able to identify several omissions in the standard and suggest improvements.

Consequences. Based on the fact that we can pinpoint omissions and inconsistencies in existing quality models and guidelines, it seems advisable to use an explicit meta-model for usability models, precisely to avoid the weaknesses of the other approaches. Furthermore, it helps to identify homogeneous dimensions for the usability modeling. We believe that our model of usability is a suitable basis for domain- or company-specific models that must be structured and consistent.

Outline. In Sec. 2 we describe prior work in the area of quality models for usability and the advances and shortcomings it represents. In Sec. 3, using an explicit meta-model, we discuss the quality modeling approach. The 2-dimensional model of usability that we constructed using this approach is presented in Sec. 4. This model is refined to a specific model based on an ISO standard in the case study of Sec. 5. The approach and the case study are discussed in Sec. 6. In Sec. 7 we present our final conclusions.

2 Related Work

This section describes work in the area of quality models for usability. We discuss general quality models, principles and guidelines, and first attempts to consolidate the quality models.

2.1 Quality Models for Usability

Hierarchical structures as quality models which focus mainly on *quality assurance* have been developed. A model first used by Boehm [3] and McCall et al. [8] consists of three layers: factors, criteria, and metrics. Consequently, the approach is referred to as the factor-criteria-metrics model (FCM model). The high-level factors model the main quality goals. These factors are divided into criteria and sub-criteria. When a criterion has not been divided, a metric is defined to measure the criteria. However, this kind of decomposition is too abstract and imprecise to be used for analysis and measurement. In addition, since usability is not a part of the main focus, this factor is not discussed in detail.

In order to provide means for the operational measurement of usability several attempts have been made in the domain *human-computer interaction* (HCI). Prominent examples are the models from Shackel and Richardson [9] or Nielsen [10]. Nielsen, for example, understands usability as a property with several dimensions, each consisting of different components. He uses five factors: *learnability*, *efficiency*, *memorability*, *errors*, and *satisfaction*. *Learnability* expresses how well a novice user can use the system, while the efficient use of the system by an expert is expressed by *efficiency*. If the system is used occasionally the factor *memorability* is used. This factor differentiates itself from *learnability* by the fact that the user has understood the system previously. Nielsen also mentions that the different factors can conflict with each other.

The ISO has published a number of standards which contain usability models for the operational evaluation of usability. The ISO 9126-1 [11] model consists of two parts. The first part models the *internal* as well as the *external* quality, the second part the *quality in use*. The first part describes six characteristics which are further divided into sub-characteristics. These measurable attributes can be observed during the use of the product. The second part describes attributes for *quality in use*. These attributes are influenced by all six product characteristics. Metrics are given for the assessment of the sub-characteristics. It is important to note that the standard does not look beyond the sub-characteristics intentionally.

The ISO 9241 describes human-factor requirements for the use of software systems with user interface. The ISO 9241-11 [12] provides a framework for the evaluation of a running software system. The framework includes the context of use and describes three basic dimensions of usability: *efficiency*, *effectiveness*, and *satisfaction*.

2.2 Principles and Guidelines

In addition to the models which define usability operationally, a lot of design principles have been developed. Usability principles are derived from knowledge of the HCI domain and serve as a design aid for the designer. For example, the "eight golden rules of dialogue design" from Shneiderman [13] propose rules that have a positive effect on usability. One of the rules, namely *strive for consistency*, has been criticized by Grudin [14] for its abstractness. Grudin shows that consistency can be decomposed into three parts that also can be in conflict with each other. Although Grudin does not offer an alternative model, he points out the limitations of the design guidelines.

Dix et al. [15] argue as well that if principles are defined in an abstract and general manner, they do not help the designer. In order to provide a structure for a comprehensive catalogue of usability principles Dix et al. [15] divide the factors which support the usability of a system into three categories: *learnability*, *flexibility*, and *robustness*. Each category is further divided into sub-factors. The ISO 9241-110 [16] takes a similar approach and describes seven high-level principles for the design of dialogues: *suitability for the task*, *self-descriptiveness*, *controllability*, *conformity with user expectations*, *error tolerance*, *suitability for individualization*, and *suitability for learning*. These principles are not independent of each other and some principles have an impact on other principles. For example *self-descriptiveness* influences *suitability for learning*. Some principles have a part-of relation to other principles. For example, *suitability for individualization* is a part of *controllability*. The standard does not discuss the relations between the principles and gives little information on how the principles are related to the overall framework given in [12].

2.3 Consolidated Quality Models for Usability

There are approaches which aim to consolidate the different models. Seffah et al. [5] applied the FCM model to the quality attribute *usability*. The developed model contains 10 factors which are subdivided into 26 criteria. For the measurement of the criteria the model provides 127 metrics.

The motivation behind this model is the high abstraction and lack of aids for the interpretation of metrics in the existing hierarchically-based models. Put somewhat differently, the description of the relation between metrics and high-level factors is missing. In addition, the relations between factors, e.g. *learnability* vs. *understandability*, are not described in the existing models. Seffah et al. [5] also criticize the difficulty in determining how factors relate to each other, if a project uses different models. This complicates the selection of factors for defining high-level management goals. Therefore, in [5] a consolidated model that is called *quality in use integrated measurement model* (QUIM model) is developed.

Since the FCM decomposition doesn't provide any means for precise structuring, the factors used in the QUIM model are not independent. For example, *learnability* can be expressed with the factors *efficiency* and *effectiveness* [12].

The same problem arises with the criteria in the level below the factors: They contain attributes as well as principles, e.g. *minimal memory load*, which is a

principle, and *consistency* which is an attribute. They contain attributes about the user (*likeability*) as well as attributes about the product (*attractiveness*). And lastly, they contain attributes that are similar, e.g. *appropriateness* and *consistency*, both of which are defined in the paper as capable of indicating whether visual metaphors are meaningful or not.

To describe how the architecture of a software system influences usability, Folmer and Bosch [17] developed a framework to model the quality attributes related to usability. The framework is structured in four layers. The high-level layer contains *usability definitions*, i.e. common factors like *efficiency*. The second layer describes concrete measurable *indicators* which are related to the high-level factors. Examples of indicators are *time to learn*, *speed*, or *errors*. The third layer consists of *usability properties* which are higher level concepts derived from design principles like *provide feedback*. The lowest layer describes the *design knowledge* in the community. Design heuristics, e.g. the *undo pattern*, are mapped to the *usability properties*. Van Welie [18] also approaches the problem by means of a layered model. The main difficulty with layered models is the loss of the exact impact to the element on the high-level layer at the general principle level when a design property is first mapped to a general principle.

Based on Norman's action model [19] Andre et al. developed the USER ACTION FRAMEWORK [20]. This framework aims toward a structured knowledge base of usability concepts which provides a means to classify, document, and report usability problems. By contrast, our approach models system properties and their impact on activities.

2.4 Summary

As pointed out, existing quality models generally suffer from one or more of the following shortcomings:

1. *Assessability*. Most quality models contain a number of criteria that are too coarse-grained to be assessed directly. An example is the *attractiveness* criterion defined by the ISO 9126-1 [11]. Although there might be some intuitive understanding of attractiveness, this model clearly lacks a precise definition and hence a means to assess it.
2. *Justification*. Additionally, most existing quality models fail to give a detailed account of the impact that specific criteria (or metrics) have on the user interaction. Again the ISO standard cited above is a good example for this problem, since it does not provide any explanation for the presented metrics. Although consolidated models advance on this by providing a more detailed presentation of the relations between criteria and factors, they still lack the desired degree of detail. An example is the relationship between the criterion *feedback* and the factor *universality* presented in [5]. Although these two items are certainly related, the precise nature of the relation is unclear.
3. *Homogeneity*. Due to a lack of clear separation of different aspect of quality most existing models exhibit inhomogeneous sets of quality criteria. An example is the

set of criteria presented in [5] as it mixes attributes like *consistency* with mechanisms like *feedback* and principles like *minimum memory load*.

3 A 2-Dimensional Approach to Model Quality

To address the problems with those quality models described in the previous section we developed the novel two-dimensional quality meta-model QMM. This meta-model was originally based on our experience with modeling maintainability [6], but now also serves as a formal specification for quality models covering different quality attributes like *usability* and *reliability*. By using an explicit meta-model we ensure the well-structuredness of these model instances and foster their preciseness as well as completeness.

3.1 The 2-Dimensional Quality Meta-Model

This model is based on the general idea of hierarchical models like FCM, i.e. the breaking down of fuzzy criteria like *learnability* into sub-criteria that are tangible enough to be assessed directly. In contrast to other models, it introduces a rigorous separation of system *properties* and *activities* to be able to describe quality attributes and their impact on the usage of a software product precisely.

This approach is based on the finding that numerous criteria typically associated with usability, e.g. *learnability*, *understandability*, and of course *usability* itself, do not actually describe the properties of a system but rather the activities performed on (or with) the system. It might be objected that these activities are merely expressed in the form of adjectives. We argue, by contrast, that this leads precisely to the most prevalent difficulty of most existing quality models, namely to a dangerous mixture of activities and actual system properties. A typical example of this problem can be found in [5] where *time behavior* and *navigability* are presented as the same type of criteria. Where *navigability* clearly refers to the navigation activity carried out by the user of the system, *time behavior* is a property of the system and not an activity. One can imagine that this distinction becomes crucial, if the usability of a system is to be evaluated regarding different types of users: The way a user navigates is surely influenced by the system, but is also determined by the individuality of the user. In contrast, the response times of systems are absolutely independent of the user. A simplified visualization of the system property and activity decompositions as well as their interrelations is shown in Fig. 1. The activities are based on Norman's action model [19]. The whole model is described in detail in Sec. 4.

The final goal of usability engineering is to improve the *usage* of a system, i.e. to create systems that support the activities that the user performs on the system. Therefore, we claim that usability quality models must not only feature these activities as first-class citizens, but also precisely describe how properties of the system influence them and therewith ultimately determine the usability of the system.

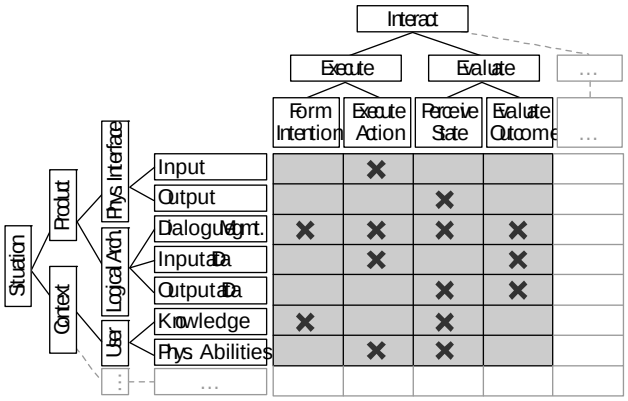

Fig. 1. Simplified quality model

3.2 Facts, Activities, Attributes, & Impacts

Our usability model does not only describe the product, i.e. the user interface, itself, but also comprises all relevant information about the situation of use (incl. the user). To render this description more precisely the model distinguishes between *facts* and *attributes*. Facts serve as a means to describe the situation of use in a hierarchical manner but do not contain quality criteria. For example, they merely model that the fact *user interface* consists of the sub-facts *visual interface* and *aural interface*.

Attributes are used to equip the facts with desired or undesired low-level quality criteria like *consistency*, *ambiguousness*, or even the simple attribute *existence*. Thus, tuples of facts and attributes express system properties. An example is the tuple [Font Face | CONSISTENCY] that describes the consistent usage of font faces throughout the user interface. Please note, that for clarity's sake the attributes are not shown in Fig. 1.

The other part of the model consists of a hierarchical decomposition of the activities performed by a user as part of the interaction with the system. Accordingly, the root node of this tree is the activity *interact* that is subdivided into activities like *execute* and *evaluate* which in turn are broken down into more specific sub-activities.

Similar to facts, activities are equipped with attributes. This allows us to distinguish between different properties of the activities and thereby fosters model preciseness. Attributes typically used for activities are *duration* and *probability of error*. The complete list of attributes is described in Sec. 4.

The combination of these three concepts enables us to pinpoint the impact that properties of the user interface (plus further aspects of the situation of use) have on the user interaction. Here impacts are always expressed as a relation between fact-attribute-tuples and activity-attribute-tuples and qualified with the direction of the impact (positive or negative):

$$[\text{Fact f} \mid \text{ATTRIBUTE } A_1] \rightarrow +/- [\text{Activity a} \mid \text{ATTRIBUTE } A_2]$$

For example, one would use the following impact description

[Font Face | CONSISTENCY] → – [Reading | DURATION]

to express that the consistent usage of font faces has a positive impact on the time needed to read the text. Similarly the impact

[Input Validity Checks | EXISTENCE] → – [Data Input | PROBABILITY OF ERROR]

is used to explain that the existence of validity checks for the input reduces the likelihood of an error.

3.3 Tool Support

Our quality models are of substantial size (e.g. the current model for maintainability has > 800 model elements) due to the high level of detail. We see this as a necessity and not a problem, since these models describe very complex circumstances. However, we are well aware that models of this size can only be managed with proper tool support. We have therefore developed a graphical editor, based on the ECLIPSE platform[1] that supports quality engineers in creating models and in adapting these models to changing quality needs by refactoring functionality[2]. Additionally, the editor provides quality checks on the quality models themselves, e.g. it warns about facts that do not have an impact on any activity.

For the distribution of quality models the editor provides an export mechanism that facilitates exporting models (or parts thereof) to different target formats. Supported formats are, e.g., simple graphs that illustrate the activity and system decomposition, but also full-fledged quality guideline documents that serve as the basis for quality reviews. This export functionality can be extended via a plug-in interface.

4 Usability Quality Model

Based on the critique of existing usability models described in Sec. 2 and using the quality modeling approach based on the meta-model from Sec. 3, we propose a 2-dimensional quality model for usability. The complete model is too large to be described in total, but we will highlight specific core parts of the model to show the main ideas.

Our approach to quality modeling includes *high-level* and *specific* models. The aim of the high-level model is to define a basic set of facts, attributes, and activities that are independent of specific processes and domains. It is simultaneously abstract and general enough to be reusable in various companies and for various products. In order to fit to specific projects and situations the high-level models are refined and tailored into specific models.

[1] http://www.eclipse.org

[2] A beta version of the editor can be downloaded from http://www4.cs.tum.edu/~ccsm/qmm.

4.1 Goals

In accordance with existing standards [21], we see four basic principles needed for defining usability:

- *Efficiency*. The utilization of resources.
- *Effectiveness*. The sharing of successful tasks.
- *Satisfaction*. The enjoyment of product use.
- *Safety*. The assurance of non-harmful behavior.

Frøkjær, Hertzum, and Hornbæk [22] support the importance of these aspects: "Unless domain specific studies suggest otherwise, effectiveness, efficiency, and satisfaction should be considered independent aspects of usability and all be included in usability testing." However, we do not use these principles directly for analysis, but rather to define the usability goals of the system. The goals are split into several attributes of the activities inside the model. For example, the effectiveness of the user interface depends on the probability of error for all activities of usage. Therefore, all impacts on the attribute *probability of error* of activities are impacts on the effectiveness and efficiency. We describe more examples below after first presenting the most important facts, activity trees, and attributes.

4.2 The Activity Subtree "Interacting with the Product"

The activity tree in the usability model has the root node *use* that denotes any kind of usage of the software-based system under consideration. It has two children, namely *execution of secondary tasks* and *interacting with the product*. The former stands for all additional tasks a user has that are not directly related to the software product. The latter is more interesting in our context because it describes the interaction with the software itself. We provide a more detailed explanation of this subtree in the following.

Activities. The activity *interacting with the product* is further decomposed, based on the seven stages of action from Norman [19] that we arranged in a tree structure (Fig. 2). We believe that this decomposition is the key for a better understanding of the relationships in usability engineering. Different system properties can have very different influences on different aspects of the use of the system. Only if these are clearly separated will we be able to derive well-founded analyses. The three activities, *forming the goal*, *executing*, and *evaluating*, comprise the first layer of decomposition. The first activity is the mental activity of deciding which goal the user wants to achieve. The second activity refers to the actual action of planning and realizing the task. Finally, the third activity stands for the gathering of information about the world's state and understanding the outcome.

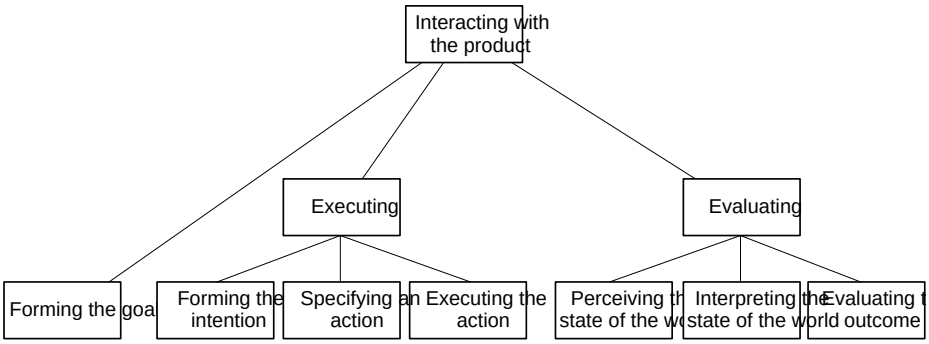

Fig. 2. The subtree for "Interacting with the Product" (adapted from [19])

The *executing* node has again three children: First, the user forms his intention to do a specific action. Secondly, the action is specified, i.e. it is determined what is to be done. Thirdly, the action is executed. The *evaluating* node is decomposed into three mental activities: The user perceives the state of the world that exists after executing the action. This observation is then interpreted by the user and, based on this, the outcome of the performed action is evaluated. Scholars often use and adapt this model of action. For example, Sutcliffe [23] linked error types to the different stages of action and Andre et al. [20] developed the USER ACTION FRAMEWORK based on this model.

Attributes. To be able to define the relation of the facts and activities to the general usability goals defined above, such as *efficiency* or *effectiveness*, we need to describe additional properties of the activities. This is done by a simple set of attributes that is associated with the activities:

- *Frequency*. The number of occurrences of a task.
- *Duration*. The amount of time a task requires.
- *Physical stress*. The amount of physical requirements necessary to perform a task.
- *Cognitive load*. The amount of mental requirements necessary to perform a task.
- *Probability of error*. The distribution of successful and erroneous performances of a task.

As discussed in Sec. 4.1, these activity attributes can be used to analyze the usability goals defined during requirements engineering. We already argued that the effectiveness of a user interface is actually determined by the probability of error of the user tasks. In our model, we can explicitly model which facts and situations have an impact on that. The efficiency sets the frequency of an activity into relation to a type of resources: time (duration), physical stress, or cognitive load. We can explicitly model the impacts on the efficiency of these resources. Further attributes can be used to assess other goals.

4.3 The Fact Subtree "Logical User Interface"

The fact tree in the usability model contains several areas that need to be considered in usability engineering, such as the physical user interface or the usage context. By means of the *user* component, important properties of the user can be described. Together with the *application* it forms the context of use. The *physical output devices* and the *physical input devices* are assumed to be part of the physical user interface. However, we concentrate on a part we consider very important: the *logical user interface*. The decomposition follows mainly the logical architecture of a user interface as shown in Fig. 3.

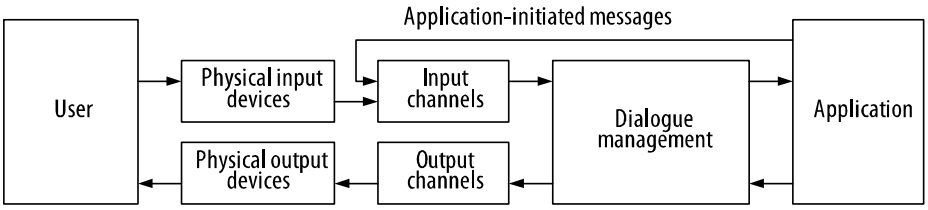

Fig. 3. The user interface architecture

Facts. The logical user interface contains *input channels*, *output channels*, and *dialogue management*. In addition to the architecture, we also add data that is sent via the channels explicitly: *input data* and *output data*. The architecture in Fig. 3 also contains a specialization of input data, *application-initiated messages*. These messages, which are sent by the *application*, report interrupts of the environment or the application itself to the *dialogue management* outside the normal response to inputs.

Attributes. The attributes play an important role in the quality model because they are the properties of the facts that can actually be assessed manually or automatically. It is interesting to note that it is a rather small set of attributes that is capable of describing the important properties of the facts. These attributes are also one main building block that can be reused in company- or domain-specific usability models. Moreover, we observe that the attributes used in the usability model differ only slightly from the ones contained in the maintainability model of [6]. Hence, there seems to be a common basic set of those attributes that is sufficient – in combination with facts – for quality modeling.

- *Existence*. The most basic attribute that we use is whether a fact exists or not. The pure existence of a fact can have a positive or negative impact on some activities.
- *Relevance*. When a fact is relevant, it means that it is appropriate and important in the context in which it is described.

- *Unambiguousness*. An unambiguous fact is precise and clear. This is often important for information or user interface elements that need to be clearly interpreted.
- *Simplicity*. For various facts it is important that in some contexts they are simple. This often means something similar to small and straightforward.
- *Conformity*. There are two kinds of conformity: conformity to existing standards and guidelines, and conformity to the expectations of the user. In both cases the fact conforms to something else, i.e. it respects and follows the rules or models that exist.
- *Consistency*. There are also two kinds of consistency: internal consistency and external consistency. The internal consistency means that the entire product follows the same rules and logic. The external consistency aims at correspondence with external facts, such as analogies, or a common understanding of things. In both cases it describes a kind of homogeneous behavior.
- *Controllability*. A controllable fact is a fact which relates to behavior that can be strongly influenced by the actions of the user. The user can control its behavior.
- *Customizability*. A customizable fact is similar to a controllable fact in the sense that the user can change it. However, a customizable fact can be preset and fixed to the needs and preferences of the user.
- *Guardedness*. In contrast to customizability and controllability, a guarded fact cannot be adjusted by the user. This is a desirable property for some critical parts of the system.
- *Adaptability*. An adaptive fact is able to adjust to the user's needs or to its context dependent on the context information. The main difference to customizability is that an adaptive fact functions without the explicit input of the user.

4.4 Examples

The entire model is composed of the activities with attributes, the facts with the corresponding attributes and the impacts between attributed facts and attributed activities. The model with all these details is too large to be described in detail, but we present some interesting examples: triplets of an attributed fact, an attributed activity, and a corresponding impact. These examples aim to demonstrate the structuring that can be achieved by using the quality meta-model as described in Sec. 3.

Consistent Dialogue Management. A central component in the logical user interface concept proposed in Sec. 4.3 is the *dialogue management*. It controls the dynamic exchange of information between the product and the user. In the activities tree, the important activity is carried out by the user by interpreting the information given by the user interface. One attribute of the dialogue management that has an impact on the interpretation is its *internal consistency*. This means that its usage concepts are similar in the entire dialogue management component. The corresponding impact description:

[Dialogue Management | INTERNAL CONSISTENCY] → – [Interpretation | PROB. OF ERROR]

Obviously, this is still too abstract to be easily assessed. This is the point where company-specific usability models come in. This general relationship needs to be refined for the specific context. For example, menus in a graphical user interface should always open the same way.

Guarded Physical Interface. The usability model does not only contain the logical user interface concept, but also the physical user interface. The *physical interface* refers to all the hardware parts that the user interacts with in order to communicate with the software-based system. One important attribute of such a physical interface is *guardedness*. This means that the parts of the interface must be guarded against unintentional activation. Hence, the guardedness of a physical interface has a positive impact on the *executing* activity:

[Physical Interface | GUARDEDNESS] → – [Executing | PROBABILITY OF ERROR]

A physical interface that is not often guarded is the touchpad of a notebook computer. Due to its nearness to the location of the hands while typing, the cursor might move unintentionally. Therefore, a usability model of a notebook computer should contain the triplet that describes the impact of whether the touchpad is guarded against unintentional operation or not.

5 Case Study: Modeling the ISO 15005

To evaluate our usability modeling approach we refine the high-level model described in Sec. 4 into a specific model based on the ISO 15005 [7]. This standard describes ergonomic principles for the design of *transport information and control systems* (TICS). Examples for TICS are driver information systems (e.g. navigation systems) and driver assistance systems (e.g. cruise control). In particular, principles related to dialogues are provided, since the design of TICS must take into consideration that a TICS is used in addition to the driving activity itself.

The standard describes three main *principles* which are further subdivided into eight *sub-principles*. Each sub-principle is motivated and consists of a number of *requirements* and/or *recommendations*. For each requirement or recommendation a number of examples are given.

For example, the main principle *suitability for use while driving* is decomposed among others into the sub-principle *simplicity*, i.e. the need to limit the amount of information to the task-dependent minimum. This sub-principle consists, among others, of the recommendation to optimize the driver's mental and physical effort. All in all the standard consists of 13 requirements, 16 recommendations, and 80 examples.

5.1 Approach

We follow two goals when applying our method to the standard: First, we want to prove that our high-level usability model can be refined to model such principles. Secondly, we want to discover inconsistencies, ill-structuredness, and implicitness of important information.

Our approach models every element of the standard (e.g. high-level principles, requirements, etc.) by refinement of the high-level model. For this, the meta-model elements (e.g. facts, attributes, impacts, etc.) are used. We develop the specific model by means of the tool described in Sec. 3.3. The final specific model consists of 41 facts, 12 activities, 15 attributes, 48 attributed facts, and 51 impacts.

5.2 Examples

To illustrate how the elements of the standard are represented in our specific model, we present the following examples.

Representation of Output Data. An element in the logical user interface concept proposed in Sec. 4.3 is the *output data*, i.e. the information sent to the driver. A central aspect is the representation of the data. One attribute of the representation that has an impact on the interpretation of the state of the system is its *unambiguousness*, i.e. that the representation is precise and clear. This is especially important so that the driver can identify the exact priority of the data. For example, warning messages are represented in a way that they are clearly distinguishable from status messages.

[Output Data | UNAMBIGUOUSNESS] → – [Interpretation | PROBABILITY OF ERROR]

Another attribute of the representation that has an impact on the interpretation is the *internal consistency*. If the representations of the output data follow the same rules and logic, it is easier for the driver to create a mental model of the system. The ease of creating a mental model has a strong impact on the ease of interpreting the state of the system:

[Output Data | INTERNAL CONSISTENCY] → – [Interpretation | DURATION]

One attribute of the representation that has an impact on the perception is *simplicity*. It is important for the representation to be simple, since this makes it easier for the driver to perceive the information:

[Output Data | SIMPLICITY] → – [Perception | COGNITIVE LOAD]

Guarded Feature. A TICS consists of several features which must not be used while driving the vehicle. This is determined by the manufacturer as well as by regulations. One important attribute of such features is its *guardedness*. This means that the feature is inoperable while the vehicle is moving. This protects the driver from

becoming distracted while using the feature. The guardedness of certain features has a positive impact on the *driving* activity:

[Television | GUARDEDNESS] → – [Driving | PROBABILITY OF ERROR]

5.3 Observations & Improvements

As a result of the meta-model-based analysis, we found the following inconsistencies and omissions:

Inconsistent Main Principles. One of the three main principles, namely *suitability for the driver*, does not describe any activity. The other two principles use the activities to define the high-level usability goals of the system. For example, one important high-level goal is that the TICS dialogues do not interfere with the driving activity. Hence, we suggest that every main principle should describe an activity and the high-level goals of usability should be defined by means of the attributes of the user's activities.

Mixed Sub-Principles. The aspects described by the sub-principles are mixed: Three sub-principles describe activities without impacts, three describe facts without impacts, and the remaining two describe impacts of attributes on activities. This mix-up of the aspects described by the sub-principles must be resolved.

We believe that in order to make a design decision it is crucial for the software engineer to know which high-level goals will be influenced by it. Sub-principles which only describe attributes of system entities do not contribute toward design decisions. The same holds true for sub-principles which only describe activities, since they are not related to system entities. For this reason we suggest that all sub-principles that only describe activities should be situated at the main principle level, while those sub-principles that describe software entities should be situated at the requirement level.

Requirements with Implicit Impacts. 9 out of 13 requirements do not explicitly describe impacts on activities. Requirements serve to define the properties which the system entities should fulfill. If a requirement does not explicitly describe its impacts on activities, the impact could be misunderstood by the software engineer. Hence, we suggest that requirements should be described by attributed facts and their impacts on activities.

Incomplete Examples. 14 out of 80 examples only describe facts and their attributes, leaving the impacts and activities implicit. To provide complete examples we suggest that the examples should be described with explicit impacts and activities.

6 Discussion

The usability model acts as a central knowledge base for the usability-related relationships in the product and process. It documents in a structured manner how the properties of the system, team, and organization influence different usage activities. Therefore, it is a well-suited basis for quality assurance (QA). It can be used in several ways for constructive as well as analytical QA. Some of these have been shown to be useful in an industrial context w.r.t. maintainability models.

Constructive QA. The knowledge documented in the quality model aids all developers and designers in acquiring a common understanding of the domain, techniques, and influences. This common understanding helps to avoid misunderstandings, and improvements to the quality model become part of a continuous learning process for all developers. For example, by describing the properties of the system artifacts, a glossary or terminology is built and can be easily generated into a document. This glossary is a living artifact of the development process, not only because it is a materiality itself, but also because it is inside and part of a structured model. Hence, by learning and improving the way developers work, it is possible to avoid the introduction of usability defects into the product.

Analytical QA. The identified relationships in the usability model can also be used for analytical QA. With our quality model we aim to break down the properties and attributes to a level where we can measure them and, therefore, are easily able to give concrete instructions in analytical QA. In particular, we are able to generate guidelines and checklists for reviews from the model. The properties and attributes are there and subsets can easily be selected and exported in different formats so that developers and reviewers always have the appropriate guidelines at hand. Moreover, we annotate the attributed properties in the model, whether they are automatically, semi-automatically, or only manually assessable. Hence, we can identify quality aspects that can be analyzed automatically straightforwardly. Thus, we are able to use all potential benefits of automation.

Analyses and Predictions. Finally, more general analysis and predictions are possible based on the quality model. One reason to organize the properties and activities in a tree structure is to be able to aggregate analysis to higher levels. This is important to get concise information about the quality of the system. To be able to do this, the impacts of properties on activities must be quantified. For example, the usability model is a suitable basis for cost/benefit analysis because the identified relationships can be quantified and set into relation to costs similar to the model in [24]. In summary, we are able to aid analytical QA in several ways by utilizing the knowledge coded into the model.

7 Conclusion

Usability is a key criterion in the quality of software systems, especially for its user.

It can be decisive for its success on the market. However, the notion of usability and its measurement and analysis are still not fully understood. Although there have been interesting advances by consolidated models, e.g. [5], these models suffer from various shortcomings, such as inconsistencies in the dimensions used. An approach based on an explicit meta-model has proven to be useful for the quality attribute *maintainability*. Hence, we propose a comprehensive usability model that is based on the same meta-model.

Using the meta-model and constructing such a usability model allows us to describe completely the usability of a system by its facts and their relationship with (or impact on) the activities of the user. We support the consistent and unambiguous compilation of the usability knowledge available. The general model still needs to be refined for specific contexts that cannot be included in a general model. By utilizing a specific usability model, we have several benefits, such as the ability to generate guidelines and glossaries or to derive analyses and predictions.

The usefulness of this approach is demonstrated by a case study in which an ISO standard is modeled and several omissions are identified. For example, the standard contains three sub-principles which describe activities, but no impacts on them, as well as nine requirements that have no described impacts. This hampers the justification of the guideline: A rule that is not explicitly justified will not be followed.

For future work we plan to improve further the general usability model and to carry out more case studies in order to validate further the findings of our current research. Furthermore, other quality attributes, e.g. *reliability*, will also be modeled by means of the meta-model to investigate whether this approach works for all attributes. If this be the case, the different models can be combined, since they are all based on a common meta-model.

References

1. Bevan, N.: International standards for HCI and usability. Int. J. Hum.-Comput. Stud. 55 (2001) 533–552
2. Seffah, A., Metzker, E.: The obstacles and myths of usability and software engineering. Commun. ACM 47(12) (2004) 71–76
3. Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., Macleod, G.J., Merrit, M.J.: Characteristics of Software Quality. North-Holland (1978)
4. Dromey, R.G.: A model for software product quality. IEEE Trans. Software Eng. 21(2) (1995) 146–162
5. Seffah, A., Donyaee, M., Kline, R.B., Padda, H.K.: Usability measurement and metrics: A consolidated model. Software Quality Control 14(2) (2006) 159–178
6. Broy, M., Deissenboeck, F., Pizka, M.: Demystifying maintainability. In: Proc. 4th Workshop on Software Quality (WoSQ ’06), ACM Press (2006)
7. ISO 15005: Road vehicles – Ergonomic aspects of transport information and control systems – Dialogue management principles and compliance procedures (2002)

8. Cavano, J.P., McCall, J.A.: A framework for the measurement of software quality. In: Proc. Software quality assurance workshop on functional and performance issues. (1978) 133–139
9. Shackel, B., Richardson, S., eds.: Human Factors for Informatics Usability. Cambridge University Press (1991)
10. Nielsen, J.: Usability Engineering. AP Professional (1993)
11. ISO 9126-1: Software engineering – Product quality – Part 1: Quality model (2001)
12. ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability (1998)
13. Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. 3 edn. Addison-Wesley (1998)
14. Grudin, J.: The case against user interface consistency. Commun. ACM 32(10) (1989) 1164–1173
15. Dix, A., Finley, J., Abowd, G., Beale, R.: Human-Computer Interaction. 2 edn. Prentice-Hall (1998)
16. ISO 9241-110: Ergonomics of human-system interaction – Part 110: Dialogue principles (2006)
17. Folmer, E., Bosch, J.: Architecting for usability: A survey. The Journal of Systems and Software 70 (2004) 61–78
18. van Welie, M., van der Veer, G.C., Eliëns, A.: Breaking down usability. In: Proc. International Conference on Human-Computer Interaction (INTERACT '99), IOS Press (1999) 613–620
19. Norman, D.A.: Cognitive engineering. In Norman, D.A., Draper, S.W., eds.: User Centered System Design: New Perspectives on Human-Computer Interaction. Lawrence Erlbaum Associates (1986) 31–61
20. Andre, T.S., Hartson, H.R., Belz, S.M., McCreary, F.A.: The user action framework: A reliable foundation for usability engineering support tools. Int. J. Hum.-Comput. Stud. 54(1) (2001) 107–136
21. ISO 9126-4: Software engineering – Product quality – Part 4: Quality in use metrics (2004)
22. Frøkjær, E., Hertzum, M., Hornbæk, K.: Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated? In: Proc. Conference on Human Factors in Computing Systems (CHI '00), ACM Press (2000) 345–352
23. Sutcliffe, A.: User-Centered Requirements Engineering: Theory and Practice. Springer-Verlag (2002)
24. Wagner, S.: A model and sensitivity analysis of the quality economics of defect-detection techniques. In: Proc. International Symposium on Software Testing and Analysis (ISSTA '06), ACM Press (2006) 73–83

Questions

Laurence Nigay:

Question: You describe the product using two models but there are a lot of usability models, why only the two? Task models can be used to describe properties such as reachability.

Answer: Factors and activity can capture all this information in these models and then relate it to activities.

Michael Harrison:

Question: Much is said at the moment about the need to consider the features of complex systems that cannot be characterized by a decompositional approach – so-called emergent properties. So for example a high reliability organization is one for reasons that cannot easily be understood using the probing style techniques that you have described. What is your opinion of this perspective and do you agree that there is a need to explore alternatives to the style of analysis that you describe?

Answer: This technique is better than other techniques that exist and none of them handle these emergent properties of complex systems.

Thomas Memmel:

Question: If you say you are building a model-based system to understand design would you say that simulation is not also a good idea?

Answer: Of course both are required. I have described just one aid for the developer.