Early Usability Evaluation in Model-Driven Architecture Environments

Silvia Abrahão and Emilio Insfran

Department of Information Systems and Computation
Valencia University of Technology
Camino de Vera, s/n, 46022, Valencia, Spain
{sabrahao, einsfran}@dsic.upv.es

Abstract

Due to the increasing interest in the Model Driven Architecture (MDA) paradigm, the conceptual models have become the backbone of the software development process. So far some methods exist to develop a user interface according to a MDA-compliant method, none of them explicitly connects usability to their process activities. In this paper, we present a framework which incorporates usability as part of a MDA development process. In particular, a usability model for early evaluation is proposed. Using this model, the usability of a software system is evaluated and improved at the platform independent model (PIM) level. It focuses on the correspondences between the abstract user interface elements and the final user interface elements in a specific platform (CM). This framework has been successfully applied to an industrial MDA tool.

Keywords: User Interfaces, Conceptual Models, Quality Models, Model-Driven Architecture.

1. Introduction

Many approaches to evaluate the usability of software systems have been proposed in the last years [5] [10] [13] [19] [21] [24]. Most of them focus on defining a set of attributes that explains usability and on developing guidelines and heuristics for testing it. As drawback, they mainly focus on artifacts used in the late stages of software development without connecting usability evaluation results with changes in software design.

Other studies demonstrated that 50% of the implementation stage time is dedicated to user interface construction [18]. This has motivated the development of tools for automatic generation of user interfaces (UIs). Among these tools, those that follow the Model-Driven Architecture approach [16] (MDA) seem to be appropriate for the development of user interfaces. In this approach, models are applied in all steps of development up to a target platform, providing a complete software production process where model transformation at different levels of abstraction becomes the basic strategy to obtain a final software product (including the UI) from conceptual models. So far, several works have been conducted in the area of Software Engineering (SE) as well as in Human-Computer Interaction (HCI) to see how MDA could be effectively applied. However, we are not aware of any existing work that actually performs any explicit usability evaluation of the UI as issued by such a model-based software production method.

In this paper, we introduce a framework that incorporates usability in MDA development processes.

Figure 1. Early usability evaluation in MDA processes

This framework was instantiated in a concrete MDA tool: OlivaNova Model Execution (ONME) [20]. As suggested in Figure 1, the usability of a software application is evaluated and improved at the platform

---

1 This work is financed by the DESTINO project (ref. TIN2004-03534) and developed in a technology transfer project with CARE Technologies S.A.
independent model (PIM) using a usability model (1A). As outcome, a list of usability problems is produced. However, instead of relate a given usability problem with changes in the interface tier (code model level), we relate it to the elements of the PIM that are affected by it. Another benefit, which has no precedents in MDA literature, is the opportunity to have a feedback on the transformation rules itself (1B). This iterative process combines model development and usability evaluation and can be applied until the PIM has the required level of usability. Note that the evaluation is made without users. This allows very early usability evaluation by verifying the PIM properties that are related to usability.

This work is structured as follows. Section 2 discusses the existing approaches to model usability. Section 3 provides an overview of the usability evaluation framework and its instantiation to a specific MDA tool. Section 4 shows the usability model underlying the framework. Section 5 discusses some results obtained with the framework instantiation and their implications to the PIM and PSM levels. Finally, section 6 presents the conclusions and future work.

2. Related Work

Several quality models for usability evaluation have been proposed in the last few years. These models have been defined based on existing standards such as ISO/IEC 9241-11 [12] and ISO/IEC 9126-1 [13] or defined from scratch. The ISO/IEC 9126-1 provides a software product quality model that is intended to be used as a general-purpose standard quality model. The ISO 9241-11 explains how usability can be specified in terms of user performance and satisfaction. Although these models are useful in providing principles and recommendations, they are generic and need to be extended and/or decomposed for their use on different kind of systems.

Quality models defined from scratch include the proposals of Dromey [7], Dix [6], and Nielsen [19]. Dromey [7] uses a constructive strategy to characterize behaviours and uses. Behaviour is something that the software exhibits itself when it executes under the influence of a set of inputs (e.g., usability). Use is something that different users do with or to software. The model of Dromey enumerates specific properties and classifies them as pertaining to certain software characteristics, and further enumerates software characteristics that characterize each behaviour and use.

The model proposed by Dix [6] is defined according to three main factors: learnability, flexibility, and robustness. These factors are subdivided into sub-characteristics that influence the concept that they belong to. The model proposed by Nielsen [19] is more detailed than the Dix model. It focuses on social acceptability and practical acceptability. Usability is considered as a sub-characteristic of usefulness, which is, in turn, a sub-characteristic of practical acceptability. The usability dimension of the model incorporates the following attributes: easy to learn, efficient to use, easy to remember, fewer errors, and subjectively pleasing. The first four attributes represent the quality characteristics of a software product, whereas the last attribute represents end-users’ subjective evaluations of a software system.

A similar model was proposed by Shneiderman [21]. Shneiderman calls his definition of usability “five measurable human factors that are central to the evaluation of human factor goals”. These factors are: speed of performance, time to learn, retention over time, rate of errors by users, and subjective satisfaction.

In addition, there is another category of quality models that have been defined based on existing quality standards. The proposals of Van Welie [22], Abran [2], and Fitzpatrick and Higgins [8] are in this category.

Van Welie [22] proposed a model that is structured in three layers. In the highest layer are the three aspects that appear in the ISO 9241-11 [12]: efficiency, effectiveness, and satisfaction. The second layer contains a set of usage indicators of usability that can be observed in practice when users interact with the system (learnability, errors/safety, satisfaction, performance speed, and memorability). Finally, the lowest layer contains the means, which cannot be observed in user tests, but can be used in heuristics to improve the usage indicators.

Some other proposals try to combine the ISO 9241-11 and the ISO/IEC 9126-1. For example, Abran [2] considers the ISO 9241-11 as the basis for his model and integrates other usability characteristics (learnability and security) from the ISO/IEC 9126 and other sources. The proposed enhanced model follows a three-layer structure composed by characteristics, sub-characteristics, and measures. Some examples of measures for effectiveness are: percentage of tasks accomplished, ratio of failure, and percentage of tasks achieved per unit of time.

Other proposals are based on the quality factors proposed by McCall et al. [15]. For example, Fitzpatrick and Higgins [8] identified problems that are related to the actual definitions of software usability and models such as the lack of consistency of the software attributes and the definitions that do not support a universal set of measures. To solve these problems, they proposed a model with basic quality factors that are taken from the McCall model. It has been refined and extended through an analysis of three strands that influence these factors.

After reviewing the existing quality models approaches, we concluded that no single quality model copes with all the required usability attributes. All the aforementioned proposals include user performance-based criteria. Dromey [7] is the only proposal that does not consider any user perception-based attribute, although there exist well-know questionnaires to measure it.
Overall, what is missing is a usability model that can be applied in the early stages of software development providing recommendation for changes. In addition, a “good” model should address the following concerns:

- Usability is an abstract, non-atomic concept that can be properly decomposed taking into account performance-based and perception-based attributes of the user interaction with the system.
- Inter-relationship among attributes. The usability of a software system is not a direct sum of the values of each one of the model attributes. It is necessary not only to reach a certain level for each attribute but also to consider how the different attributes are positively or negatively affected by each other to identify possible conflicts.
- Usability also depends on the user type for which the system has been developed. For instance, learnability is more important for novices than for expert users.

3. A Framework for Usability Evaluation

Figure 2 presents the framework proposed for early usability evaluation. This framework suggests that the usability of a software system can be evaluated and improved at the PIM level. As outcome, a list of usability problems is produced. These problems can be fixed by changing the interface model (PIM) or the transformation rules (PSM). We analyze these two perspectives below.

![Usability framework](adapted from [9])

3.1 Usability at the Platform Independent Model

Usability at this level is defined through mappings that are implemented in three layers: usability definitions, ergonomic criteria [4], and PIM elements. The first layer is concerned with the usability definitions.

Usability is defined in the ISO/IEC 9126-1 [13] as: “the capability of the software product to be understood, learned, used, and attractive to the user, when used under specified conditions”. On the other hand, the ISO 9241-11 [12] defines usability as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use”. This last definition is in agreement with the definition of quality in use provided by the ISO/IEC 9126-4 [14].

Although both usability views are important and needed as part of a software development process, we believe that the ISO 9126-1 view is more appropriate for early usability evaluation as this standard deal with characteristics of the product itself and can be used to evaluate the intermediate artifacts (PIMs) produced in a MDA development process. Note that the ISO 9241-11 view on usability is focused on the product effect (efficiency, effectiveness, and user satisfaction) at the time of use of the product. In [1] we describe an example of usability evaluation using the ISO 9241-11 definition.

According to ISO/IEC 9126-1, usability is a quality characteristic that includes the following sub-characteristics: understandability, learnability, operability, attractiveness, and compliance. These concepts are abstracts and cannot be directly measured. In order to decompose these sub-characteristics into measurable attributes, we have analyzed several usability and ergonomic criteria from the HCI field [4] [10] [21]. Based on these criteria, we developed a usability model that supports early usability evaluation (see section 4). This model establishes the relationship among usability attributes and the elements of the PIM (i.e., ONME presentation patterns) that are affected to them. The purpose is to describe which element of the model contributes to the satisfaction of certain usability attributes. Such a model is the basis for usability inspection methods based on standards.

3.2 Usability at the Platform Specific Model

Thanks to the model transformations that implement the relationships between the PIM elements (abstract UI) and the software components (widgets in the final UI), the usability of an application can be obtained by construction. Let us illustrate this notion. Depending on the transformation rules that are applied, the abstract UI patterns (AOI - Abstract Object Interface) are instantiated in the corresponding software components in the target platform architecture (COI - Concrete Object Interface). This strategy is called reification [23].

The duality between COIs and AOIs has allowed us to abstract usability attributes and other relevant properties from the user interface elements. We verified that some usability attributes are fulfilled during the reification process. For instance, the following user guidance attributes related to visual feedback are directly ensured:
visual feedback in menus or dialog boxes where choices can be selected or where the cursor is pointing, visual feedback where options are already selected for multiple options, and visual feedback when objects are selected. In addition, other attributes related to the legibility of the UI are also ensured: prompts and error messages always appear in the same place(s), error messages displayed in pop-up windows indicate the field(s) where the error has occurred, etc.

3.3 Applying the Usability Framework to ONME

OlivaNova Model Execution (ONME) is a MDA tool that uses models to separate the conceptual specification (PIM) from its software implementations (CM) and the corresponding transformations rules (PSM).

The Conceptual Model (PIM) is organized in four complementary views: i) the Object Model, which describes the static properties of the system in terms of classes and their relationships; ii) the Dynamic Model, which describes the aspects related to the control, valid lives, and the interaction between objects; iii) the Functional Model, which describes the semantics associated to the state changes of the objects caused by the execution of events; and, iv) the Presentation Model, which allows the abstract specification of the user interface. A Transformation Model establishes specific transformations rules (PSM) for each technological platform allowing to obtain a final software application (CM) from its conceptual model. Currently, the target platforms are: VB, C#, ASP.NET, Cold Fusion and JSP.

In the following, we describe the Presentation Model which is the most important view affected by the usability. This model allows the specification of abstract user interfaces by means of a pattern language called Just-UI [17]. It is structured in three main levels.

3.3.1 Level 1 - Hierarchy of Actions Tree (HAT). This pattern organizes the functionality that is presented to the users. It is a tree containing links to interaction units.

3.3.2 Level 2 - Interaction Units (IUs). These patterns represent abstract interaction units that the user will interact with. Four types of IUs are distinguished: Service IU (SIU) that models a dialogue to execute a service; Instance IU (IIU) that models the data presentation of an instance and supports interaction with it; Population IU (PIU) that models showing sets for instances of a class using filter and ordering mechanisms; and Master/Detail IU (MDIU) which deals with master/slave presentations.

3.3.3 Level 3 - Elementary patterns (EPs). These patterns constitute the primitive building blocks of the UI. Some of them are: Introduction: captures the features of data types that the user will introduce into the system. It includes the definition of edition masks, a value by default, rank of values, help messages and validation messages. Defined Selection: enumerates the possible values for a class attribute, a data-valued argument, or a data-valued filter variable. Argument Grouping: organizes the arguments of a service into different groups to facilitate the introduction of data. Dependency: establishes dependency relationships between the arguments of a service. Supplementary Information: captures the additional information to be shown when a user selects an object. Filter: defines selection criteria and allows obtaining instances that fulfill the constraint expressed by a filter formula. Order Criteria: defined when instances have to be ordered by the value of an attribute or a set of attributes. Display Set: composed by all attributes to be shown in an abstract interaction unit. Actions: allows the user to execute actions. Navigation: allows access to instances of a related class.

4. Usability Model

We present in this section a usability model for early usability evaluation. The development of the model followed several activities. They are described below.

4.1 Define Goals

In this activity, the attributes that are to be covered by the evaluation and other issues of quality modeling such as the artifact and the stakeholder (viewpoint) should be clearly defined. According to the Goal-Question-Metric (GQM) [3], the goal of our usability model is: analyze the PIM (task, dialog, presentation models, etc.) and/or the CM (final UI) for the purpose of evaluate with respect to its usability from the viewpoint of a set of evaluators. The context is the software applications obtained with a MDA development process.

4.2 Specify Sub-characteristics and Attributes

This activity is concerned with the decomposition of the quality focus characteristics (i.e., usability) into a set of sub-characteristics and measurable attributes. To do this, the ISO/IEC 9126-1 quality model is used as the basis for building the proposed usability model. In this model, the following sub-characteristics are identified:

- **Learnability**: the capability of the software product to enable the user to learn its application.
- **Understandability**: the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.
- **Operability**: the capability of the software product to enable the user to operate and control it.
• **Attractiveness**: the capability of the software product to be attractive to the user.

• **Compliance**: the capability of the software product to adhere to standards, conventions, style guides, or regulations related to usability.

All these high-level characteristics can be decomposed into more detailed sub-characteristics and attributes based on the user-interaction or other attributes concerned with help features, user documentation, and installation instructions. Actually, such a decomposition must be done in order to support usability inspection methods and to identify and fix usability problems.

We argue that the first three sub-characteristics are related to the user performance and can be quantified using objective measures. **Learnability** refers to the attributes of a software product that facilitate learning. It also corresponds to the attributes of “suitability for learning” as defined in ISO 9241-10 [11]. In our model, this sub-characteristic consists of help facilities (pre-defined keyboards, wizards, on-line help, and documentation) as well as predictability which refers to the ease with which a user can determine the result of his/her future actions, informative feedback in response to user actions, and memorability as a measure of how quickly and accurately users can remember how to use an application that they have used before.

**Understandability** refers to the attributes that facilitate understanding. We decompose this subcharacteristic to differentiate the own optical legibility of texts and images from readability that refers to the information grouping cohesiveness and density. Familiarity is the degree to which a user recognizes UI components and views their interaction as natural; UI can be familiar by imitating the visual appearance of real-world objects, by relying on standardized commands, or by following other common metaphors. **Brevity** is related to the reduction of user cognitive effort (i.e., limiting the reading and input workload and the number of action steps). Finally, the user guidance attribute is related to the message quality and navigability to guide the user by providing mechanisms such as path and current position.

**Operability** refers to the attributes that facilitate user control and operation. It also corresponds to controllability, error tolerance, and conformity with user expectations as defined in ISO 9241-10 [11]. In our model, this sub-characteristic is concerned with the capacity of the system to provide instalability facilities (e.g., user’s help, documentation and installation procedures), data validation (data validity), the user’s degree of control to services execution (controlability), and the capability of adaptation of the UI or services. This last attribute refers to the system capacity to behave contextually and according to the users’ needs and preferences. We distinguish between adaptability, which is the system capacity to be adapted by the user, and adaptivity, which is the system capacity to adapt to the users’ needs. The difference is in the agent of the adaptation. Other attributes include the operational consistency in the execution of services and controls (consistency), the prevention and error management, and the capacity to monitor the state of the system.

Finally, the last two sub-characteristics are related to the perception of the end-user (attractiveness) or evaluator (compliance) using the system. It can be mainly measured using subjective measures. We think that some aspects of attractiveness related to aesthetic design can be also quantified by measuring the UI uniformity in terms of font color, font style, font size and elements position. Some studies suggest that designs should not use more than 2-3 fully saturated intense colors.

In addition, compliance can be measured by assessing the agreement of the proposed usability model with respect to the following standards: ISO/IEC 9126-1 [13], ISO 9241-11 [12], and the Microsoft and Java style guides. The first two assessments can be done at PIM or CM levels, whereas the last one can only be done at the CM, as it analyses the fulfillment of the UI components to the prescriptions of the style guides.

The complete usability decomposition is described below:

1. **Learnability**
   - 1.1. Help Facilities
      - 1.1.1. Documentation Completeness
      - 1.1.2. Multi-user Documentation
         - 1.1.2.1. Capability profiled
         - 1.1.2.2. Role profiled
   - 1.2. Predictability
      - 1.2.1. Icon Significance
      - 1.2.2. Icon/Link Title Significance
      - 1.2.3. Action Determination
   - 1.3. Informative Feedback
   - 1.4. Memorability
      - 1.4.1. Time to remember
      - 1.4.2. Accuracy

2. **Understandability**
   - 2.1. Legibility
      - 2.1.1. Font Size
      - 2.1.2. Contrasting Text
      - 2.1.3. Disposition
   - 2.2. Readability
      - 2.2.1. Information Grouping Cohesiveness
         - 2.2.1.1. Location Grouping
         - 2.2.1.2. Format Grouping
      - 2.2.2. Information Density
   - 2.3. Familiarity
      - 2.3.1. Labeling significance
      - 2.3.2. Internationalization
      - 2.3.3. Metaphor
   - 2.4. Workload Reduction
      - 2.4.1. Brevity
         - 2.4.1.1. Values Inizialization
         - 2.4.1.1.1. Initial Values Completion
         - 2.4.1.1.2. Initial Values Modifiability
         - 2.4.1.2. Actions Minimization
2.4.2. Self-descriptiveness
2.5. User Guidance
2.5.1. Message Quality
2.5.2. Navigability

3. Operability
3.1. Installability
3.1.1. Ease of Installation
3.1.2. Multiplicity of Installation
3.1.3. Updateability
3.1.4. Update Transparency
3.2. Data Validity
3.3. Controlability
3.3.1. Edition deferral
3.3.2. Cancel Support
3.3.3. Explicit Execution
3.3.4. Interruption Support
3.3.5. Undo Support
3.3.6. Redo Support
3.4. Capability of Adaptation
3.4.1. Adaptability
3.4.2. Adaptivity
3.5. Consistency
3.5.1. Steady Behavior of Controls
3.5.2. Permanence of Controls
3.5.3. Stability of Controls
3.5.4. Order Consistency
3.5.5. Label Consistency
3.6. Error Management
3.6.1. Error Prevention
3.6.2. Error Recovery
3.7. State System Monitoring

4. Attractiveness
4.1. Background Color Uniformity
4.2. Font Color Uniformity
4.3. Font Style Uniformity
4.4. Font Size Uniformity
4.5. UI Position Uniformity
4.6. Subjective Appealing

5. Compliance
5.1. Degree of fulfillment with the ISO/IEC 9126
5.2. Degree of fulfillment with the ISO 9241-10
5.3. Degree of fulfillment with the Microsoft style guide
5.4. Degree of fulfillment with the Java style guide

4.3 Specify Relationships

After defining usability sub-characteristics and attributes, the relationships among them can be defined. The relative importance of the sub-characteristics and attributes can be determined as well as the way in which they positively or negatively affect each other. Asking experts to weigh the relative importance of usability characteristics is one way to identify the set of the most important attributes.

Another important task in this activity is the association of one or more elements of the PIM or CM models to each identified attribute. This is due to the fact that the attributes in our model are inherent to UI modeling and make a product usable. Table 1 shows several examples of the relationships established between a usability attribute and a PIM element of the ONME Presentation Model. For instance, the data validity attribute is related to the following properties of the Introduction pattern (see section 3.3.3): edition mask, value range, and maximum size. Similarly, the labeling significance attribute is related to the existence of the meaningful alias defined as part of the Introduction, Defined Selection, Ordering Criteria, Filter, and Action patterns.

4.4 Operationalize the Model

This step consists of usability model quantification. We associate one or more metrics to each model attribute. The model combines quantitative and qualitative measures. Table 1 shows some examples of potential metrics to measure some attributes of the usability model. For instance, a potential metric for the Message Quality attribute is the Proportion of meaningful error messages. An error message is meaningful if it informs the user of the nature of the error, the probable cause and the way to fix it. For instance, messages such as “Item not found” must be avoided.

On the other hand, the order consistency attribute is measured by verifying the following properties of the PIM (see section 3.3): (i) the order of the arguments in the services of the same class or related classes (e.g., the create and modify services must to have its arguments in the same order); (ii) the order of the attributes in the display set of the same or related classes; (iii) the order of the navigation patterns; (iv) the order of the services in the action pattern of a class or related classes (e.g., the services create, modify and delete must to appear in the same order across user interfaces).

5 Results Obtained and their Effects on the PIM and PSM

The proposed model was applied across several usability inspections carried out in ONME generated applications. This allowed us to validate the model experimentally. In the following, we provide some lessons learnt from the experience of applying the usability model at the PIM level. Some usability problems (UPs) observed as well as recommendations to solve them are also described.

For reasons of simplicity, we explain each usability problem using the following scheme: usability problem is the description of the problem; context is the elements of the PIM (Presentation Model) where the problem was observed; related ergonomic criteria are the ergonomic criteria from the HCI field affected by the problem; usability attributes are the sub-characteristics and attributes in the model that are affected; example is an example that illustrates the problem; and a recommendation to solve the problem.
Table 1. Examples of sub-characteristics, attributes, PIM elements affected and potential metrics

<table>
<thead>
<tr>
<th>Sub-characteristic</th>
<th>Attribute</th>
<th>PIM elements affected</th>
<th>Rationale</th>
<th>Potential metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legibility</td>
<td>Information Grouping Cohesiveness</td>
<td>HAT (actions grouping), Service IU (argument grouping).</td>
<td>Actions can be grouped to build user menus using criteria such as hierarchic organization and sorting. The arguments can also be grouped</td>
<td>Proportion of actions grouped by class Proportion of actions grouped by related classes</td>
</tr>
<tr>
<td>User Guidance</td>
<td>Message Quality</td>
<td>Introduction (help messages), Defined Selection (help messages)</td>
<td>The quality of error messages promotes users’ understandability. The help messages of the patterns must indicate to the users the reasons for their errors, their nature, and the way to solve their errors.</td>
<td>Proportion of meaningful error messages Proportion of meaningful warning messages</td>
</tr>
<tr>
<td>Operability</td>
<td>Data Validity</td>
<td>Introduction (edition mask, value range, and maximum size)</td>
<td>It addresses the problem of detecting and preventing data entry errors or actions. For instance, the value range represents the minimum and maximum values that can be introduced.</td>
<td>Number of defined properties errors per IU Proportion of patterns in an IU that applies data validation</td>
</tr>
<tr>
<td></td>
<td>Labeling Significance</td>
<td>The alias property of the introduction, supplementary information, defined selection, filter, navigation, and action patterns</td>
<td>A label is significant to the users when there is a strong semantic relationship between such labels and the items or actions they refer to.</td>
<td>Proportion of elements with meaningful names</td>
</tr>
<tr>
<td>Order Consistency</td>
<td>Population IU, Navigation, Action, Service IU</td>
<td>The order of the different presentation patterns (i.e., services, actions, navigations) must always be the same across the IUs.</td>
<td></td>
<td>Proportion of actions in the same order across related IUs Proportion of navigations in the same order across related IUs</td>
</tr>
</tbody>
</table>

Some usability problems that we found when applying the usability model are described below. In order to facilitate comprehension, we provide excerpts of the UI that were automatically generated from the PIMs (ONME Presentation Models) that were evaluated.

- **Usability Problem:** Lack of descriptive labels
  **Context:** IU and Supplementary Information.
  **Related ergonomic criteria:** guidance
  **Usability attribute:** labeling significance
  **Example:** Note that the display set of the IU (indicated by the row) has no label.

  In the following example, two composed attributes defined as part of a supplementary information pattern lack a label:

  ![Supplementary Information Pattern]

  **Recommendation:** Apply the proposed metric described in table 1. After that, identify the patterns in the Presentation Model without alias and define them.

- **Usability Problem:** Lack of consistency between a label in the HAT and the label of the interact unit associated to it.
  **Context:** HAT, SIU, IU, PIU, and MDIU.
  **Related ergonomic criteria:** consistency
  **Usability attribute:** label consistency
  **Example:** The list projects label in the HAT has a different label in the title of the associated IU (observation of project).

  ![HAT and IU Label Comparison]

  **Recommendation:** use the same label for a leaf node in the HAT and the title of the interaction unit associated to it.

  Note that the aforementioned UPs were observed at the PIM level and can be solved by changing this model (see 1A in Figure 1). However, during the PIM inspection we also observed some other UPs that can only be solved by changing the transformation rules (see 1B in Figure 1). For instance, the information grouping cohesiveness and information density attributes were poor evaluated because of the lack of grouping mechanisms. This problem is classified as follows:

  - **Usability Problem:** the users’ understanding of a display screen depends on the grouping and density of the objects (images, texts, etc.) that are presented. Currently, in ONME, all the attributes are shown in an abstract interaction unit without any grouping mechanism.
    **Context:** IUs, MDIUs.
    **Related ergonomic criteria:** user explicit control, predictability.
    **Usability attribute:** information grouping cohesiveness and information density.
Recommendation: Change the transformation rules (PSM) to incorporate different representations to the grouping mechanisms such as boxes, tabs, eye fishing, etc.

Finally, we also observed that some attributes of the usability model (i.e., action determination) cannot be evaluated since there is no element in the PIM that is related to them. This corresponds to a lack in the PIM metamodel. We provide the following recommendation in order to incorporate this attribute in the ONME technology:

- **Usability Problem**: The user needs to know possible actions and navigations allowed with the purpose of determining his/her future action. Currently, in ONME, all actions and navigations defined in an Interaction Unit are available independently of the actual object state.

  **Context**: IIUs, MIDUs, Actions, Navigations.

  **Related ergonomic criteria**: User explicit control, predictability.

  **Usability attribute**: Action determination

**Recommendation**: Extend the Presentation Model by including a new property acting as a precondition to activate/deactivate actions and/or navigations depending on the actual object state.

### 6 Conclusions and further work

We have presented a framework for early usability evaluation that fits in MDA development processes. Unlike existing proposals, the usability of a system is evaluated and improved at the PIM level instead of in the final user interface. This framework has been operationalized by means of a usability model which represents the relationships among the ISO/IEC 9126 usability sub-characteristics and the usability patterns and criteria proposed in the HCI field. This model was applied in the inspection of several PIMs (presentation models) specified using the ONME tool to generate web and desktop applications.

The obtained results allowed us to validate the model experimentally. The framework was found to be very useful not only to evaluate PIMs but also to discover deficiencies and/or limitations of the PIM expressiveness and the transformation rules to support some usability attributes.

Further work includes the application of the usability framework to other MDA tools specifically oriented to Web development (i.e., VisualWade). We are concerned, in particular, to explore and evaluate to which extend our usability model can improve the development of Web applications.

### 7 References


