EARLY USABILITY MEASUREMENT IN MODEL-DRIVEN DEVELOPMENT: DEFINITION AND EMPIRICAL EVALUATION

JOSE IGNACIO PANACH[1], NELLY CONDORI-FERNÁNDEZ[2], TANJA VOS[1], NATHALIE AQUINO[1], FRANCISCO VALVERDE[1]

[1]*Centro de Investigación en Métodos de Producción de Software,*
Universidad Politécnica de Valencia,
Camino de Vera s/n, Edificio 1F, 46022 Valencia, Spain
{jpanach, tvos, naquino, fvalverde}@pros.upv.es

[2]*Faculty of Electrical Engineering, Mathematics and Computer Science*
University of Twente
Information Systems Group
7500 AE Enschede, P.O. Box 217, The Netherlands
nelly@pros.upv.es

Usability is currently a key feature for developing quality systems. A system that satisfies all the functional requirements can be strongly rejected by end-users if it presents usability problems. End-users demand intuitive interfaces and an easy interaction in order to simplify their work. The first step in developing usable systems is to determine whether a system is or is not usable. To do this, there are several proposals for measuring the system usability. Most of these proposals are focused on the final system and require a large amount of resources to perform the evaluation (end-users, video cameras, questionnaires, etc.). Usability problems that are detected once the system has been developed involve a lot of reworking by the analyst since these changes can affect the analysis, design, and implementation phases. This paper proposes a method to minimize the resources needed for the evaluation and reworking of usability problems. We propose an early usability evaluation that is based on conceptual models. The analyst can measure the usability of attributes that depend on conceptual primitives. This evaluation can be automated taking as input the conceptual models that represent the system abstractly

Keywords: usability, conceptual modeling, model-driven development.

1. Introduction

Techniques to evaluate and ensure the usability of a software system are becoming more important every day since the success of a software product increasingly depends on the ease of use it offers. Common techniques to evaluate usability usually involve activities such as interviewing users or recording their use of the system with a video camera. These activities require a lot of resources such as the final system implementation, several end-users, recording systems, a usability lab, etc. Moreover, these techniques can only be applied once the final system has been implemented. At this stage, it is very expensive to go back and make major changes to the design [1].

To mellow these problems, we propose an early usability evaluation using conceptual models. In a Model-Driven Development (MDD) process [2], the conceptual model is composed of conceptual primitives that represent all the features of a system. Conceptual primitives are modeling elements that have the capability of abstractly representing a feature of the system, including usability. Examples of conceptual primitives are classes, attributes, and services of class diagrams. These primitives are the input for a model compiler that automatically (or semi-automatically) generates the code that implements the system. Apart from modeling the system, we state that these conceptual primitives can be used to evaluate the usability of the system before implementing it.

As the standard ISO/IEC 9126-1[3] states, usability is the capability of the software product to be understood, learned, and used as well as to be attractive to the user, when used under specified conditions. Providing a mechanism to measure the system usability is the first step to know whether or not a system is usable. According to this standard, it is possible to measure the usability through attributes. There are two types of attributes: a) **external attributes**, which can be measured once the system has been built by means of testing, operation, and observation of the executable software; and b) **internal attributes**, which can be measured before the implementation of the system, during the design of the software. All these attributes are grouped by different sub-characteristics: understandability, learnability, operability, attractiveness, and compliance.

This paper aims to delineate a method for evaluating internal usability by defining metrics for conceptual primitives that constitute conceptual models. This method can be applied automatically taking the conceptual model, that represents a system, as input. However, it is important to note that internal usability is only a portion of the measurable usability, since there are many external (subjective) attributes that can only be measured in the final system with end-users.

The usability evaluation method presented in this paper is based on a usability model defined by Fernández [4] and the measurement meta-model proposed by Cachero [5]. Fernández has defined a usability model composed of sub-characteristics and attributes. Sub-characteristics represent a general aspect that is related to usability, while attributes are measurable artifacts inside a sub-characteristic. From Fernández's usability model, we have identified a set of internal attributes than can be measurable from conceptual models. With regard to the work of Cachero, we have used her meta-model to define all the elements needed in our measurement method.

The proposed evaluation method is made up of three steps: 1) The definition of **metrics** to provide a numerical value for each usability attribute; 2) The application of **indicators** to interpret the numerical value for each usability attribute; 3) **Indicator grouping** to group the indicators obtained in the previous step by sub-characteristics. An initial version of this proposal can be consulted in [6]. This article extends that initial version and provides two main contributions: (1) the definition of a method for evaluating a system's usability through conceptual primitives; (2) an empirical evaluation of our method with 18 users.

The method proposed in this paper can be used in any software development proposal based on conceptual models. Although the metrics depend on a specific software development environment with its specific conceptual models and particular primitives, it should not be too hard to translate them to another environment. Metrics defined for a specific MDD environment are similar to other environments since elements involved in the metrics are the same. The only difference between two MDD environments is the conceptual primitive used to represent each element.

This paper focuses on a model-based environment for software development that complies with the MDD paradigm called OO-Method [7]. OO-Method has been successfully implemented in an industrial tool called OLIVANOVA [8]. The evaluation method has been defined for the OO-Method Conceptual Model, and the empirical evaluation has been done with systems developed with the OLIVANOVA tool. We have selected OO-Method because OLIVANOVA generates fully working systems, which facilitates the evaluation of our proposal in an industrial MDD tool. The OO-Method Conceptual Model is composed of four complementary models (or views):

- **The Object Model**: This specifies the system structure in terms of classes of objects and their relationships.
- **The Dynamic Model**: This represents the valid sequences of events for a class of objects and the interaction between object classes.
- **The Functional Model**: This specifies how events change object states.
- **The Presentation Model**: This specifies the graphical user interface and the interaction between the system and the user.

The remainder of the paper is structured as follows. Section 2 reviews the literature on early usability evaluation. Section 3 explains our usability evaluation method. Section 4 shows a design of an empirical experiment to evaluate our method. Section 5 analyzes the results of the experiment with statistics. Section 6 interprets the outcomes of the statistical study. Finally, section 7 presents the conclusions of this work.

2. State of the Art

Defining metrics using conceptual models is currently used in several works such as Genero [9], who defines metrics and indicators to measure the usability of UML class diagrams. Our proposal goes one step further, instead of measuring the usability of UML class diagrams, it measures the usability of the system represented within a conceptual model. To the authors' knowledge, there are few proposals for measuring usability by means of a conceptual model. Automatic evaluation of internal usability attributes are usually performed with metrics based on the generated code [10]. Next, we comment the most relevant works that propose measuring usability using conceptual models. These works have been aggregated by the type of measurement: (1) based on end-user evaluation; (2) based on conceptual primitives exclusively; (3) based on patterns; (4) and based on usability standards.

Firstly, we focus on works that propose measuring the usability with conceptual models involving end-users. In this group there are authors that deal with usability measurement using a log, such as Fraternali [11]. Fraternali defines a set of conceptual logs using meta-data derived from system conceptual specifications. These conceptual logs are used in a model-driven software development process. The logs store how the user interacts with the system, and the usability is analyzed using these logs. Another author that proposes an evaluation with logs is Lecerof [12]. This author has defined a method to evaluate user interfaces using task models and logs generated from a user test. The outcome of this evaluation is a combination of empirical testing with the information from the task model. Chatley [13] presents a technique for modeling the system at the architectural level. This author includes the interaction with the end-user and connects the architecture to a realistic mock-up of the user interface. Animation or simulation of the model allows the user to interact with the model through an interface which is very close to the one they would use in the final system.

Fraternali, Lecerof, and Chatley need the end-user to perform the evaluation. This requirement is a disadvantage since it reduces the easiness of performing the usability evaluation quickly.

Secondly, we are going to explain measurement methods based on conceptual primitives exclusively. In this group, it is important to mention the work of Bajaj [14]. Bajaj proposes a method (called CMU-WEB) to model systems as an abstraction of the real world. CMU-WEB models can measure the usability of the represented system before generating the code. The main limitation of the Bajaj's proposal is that the analyst must use the CMU-WEB conceptual model to represent the system since these models are needed to perform the early evaluation. Therefore, CMU-WEB metrics cannot be applied to other MDD methods, which is a disadvantage.

The third type of usability measurement is composed of proposals based on design patterns. In this group we find the work of Fraternali [15], who defines an XSL-based framework that is able to automatically analyze the XML specification of web applications defined with a model-driven method. That proposal aims to identify the occurrence of design patterns and to calculate metrics, revealing whether or not they are used consistently throughout the system. Fraternali's proposal can be automated and does not need the end-user. However, it has the disadvantage that it can only measure the usability of features that are included as patterns.

The fourth group of usability measurement gathers proposals based on usability standards (ISO/IEC 9126-1 [3] or ISO 9241 [16]), such as the works of Sorokin [17], Abran [18], and Seffah [19]. Sorokin uses these usability standards to evaluate usability from the beginning of an MDD development. The author uses four layers to represent the system: the Task Model, the Abstract Interaction Model, the Concrete Interaction Model, and a Rich Internet Application specification. Abran has performed an evaluation of both ISO standards and a proposal for their integration into an enhanced model. Both Sorokin and Abran define metrics to evaluate usability, but these metrics are based on experiments with end-users. This dependency is incompatible with an automatic early evaluation. With regard to Seffah, he has reviewed existing usability standards to detect limitations and complementarities. Moreover, the author unifies all these standards into a single consolidated, hierarchical model of usability measurement called Quality in Use Integrated Measurement (QUIM). QUIM includes the definition of many metrics, most of which do not need the end-user involvement. These metrics are based on the code of the system and on generated interfaces, not on conceptual primitives, making the application to MDD difficult.

Considering the state of the art, it becomes clear that early usability evaluation in MDD environments is still an immature area and more research work is needed. In order to cover this need, we propose an early usability measurement based on conceptual primitives. The goals of our proposal are the following:

- The evaluation must be independent of end-users.
- The evaluation must be performed quickly.
- The proposed evaluation can be applied to any MDD method.
- The evaluation method must be independent of the format in which usability attributes are represented in the conceptual model
- The evaluation must be based on conceptual primitives.

3. A Method for Early Usability Measurement

This section presents a method for measuring internal usability by means of conceptual primitives in MDD environments. The measurement method using conceptual primitives is based on attributes of the usability model designed by Fernández [4], which are measurable entities that were extracted from ISO/IEC 9126-1 [3] and ergonomic criteria [20]. From the work of Fernández, we have used usability attributes and the groups in which they are classified (sub-characteristics). More specifically, we have defined a metric for each internal attribute defined in the usability model.

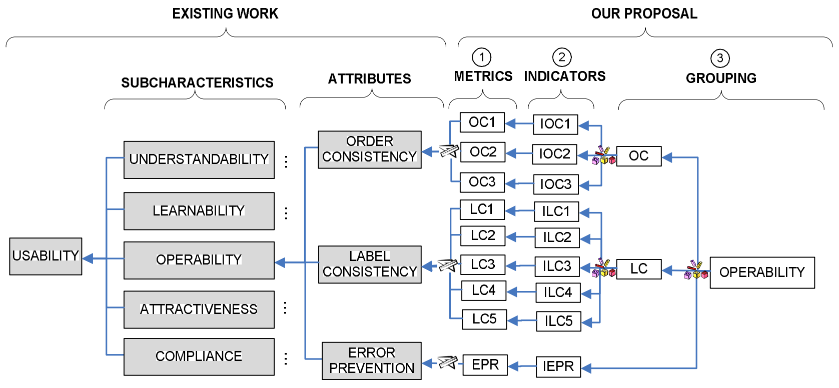

Fig. 1 Summary of the method to measure internal usability with conceptual primitives

Fig. 1 summarizes the whole method. Grey boxes represent existing elements (Fernández's work), and white boxes represent new elements introduced through our proposal. The aim of the evaluation is to determine whether or not the system is usable. According to ISO/IEC 9126-1, the concept of usability is divided into 5 sub-characteristics, each of which is composed of several attributes. From all the existing attributes [4], we focus on those that can be measured before generating the system (i.e., internal attributes). The first step of the method is to define metrics for each internal attribute. Fig. 1 illustrates the defined metrics for three attributes of the *Operability* sub-characteristic: Order Consistency, which has three metrics (OC1, OC2, OC3); Label Consistency that has 5 metrics (LC1, LC2, LC3, LC4, LC5); and Error Prevention that has only one metric (EPR). The metrics are numeric values and are defined such a way that they can be calculated using only conceptual primitives. The second step of the method consists of finding a meaning for the numeric values that the metrics have obtained. This meaning is obtained by means of indicators that transform numerical values into ordinal values. This transformation classifies each number according to a numerical range. As a result, we have the usability level by indicator, but we need the usability level for each attribute. Therefore, in the third step, we have to group each indicator by usability attribute to determine the degree of usability of this attribute. In Fig. 1, we have grouped the metrics of Order Consistency (OC) and Label Consistency (LC). Moreover, we can group the degree of usability of each of the attributes by sub-characteristic in order to determine the degree of usability of each sub-characteristic (Operability in Fig. 1).

In order to formalize our proposal, we have adapted the measurement meta-model defined by Cachero [5] (which is based on an ontology meta-model [21]) to fit our proposal. We have slightly modified some classes of the original meta-model to formalize the early usability evaluation. This new version of the meta-model provides a set of concepts with clear semantics and relationships, as shown in Fig. 2:

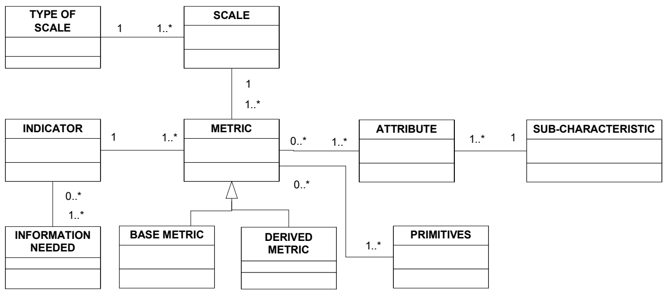

Fig. 2. Measurement meta-model adapted from the work of Cachero [5].

- **Metric**: This class represents all the metrics applied in step 1. There are two types of metrics: Base Metrics, which do not depend on other metrics, or Derived Metrics, which do depend on other metrics.
- **Primitives**: This class represents the conceptual primitives used to define the metrics.
- **Scale**: This class represents a set of values used to measure or compare the level of something. In our proposal, we work with positive integer numbers.
- **Type of scale**: This class is instantiated as ratio (quotient between two numbers).
- **Attribute**: This class represents the attributes used to derive metrics.
- **Sub-characteristic:** This class represents the sub-characteristic that groups several attributes.
- **Indicator**: This corresponds to the indicators that are defined to assign a meaning for each metric.
- **Information needed**: This corresponds to the information used to define the indicators. We have focused on existing literature in the area of human-computer interaction that defines how to develop usable systems.

Our proposal for an early usability measurement can be applied to any MDD method. As a proof of concept, this paper focuses on a specific method: OO-Method [7]. Next, we are going to introduce OO-Method and to explain how to apply our measurement proposal to it.

3.1. *The OO-Method Presentation Model*

We have selected OO-Method because it has been implemented in an industrial tool (OLIVANOVA [8]), and this facilitates empirical evaluation considerably. OLIVANOVA generates full functional systems taking a conceptual model that abstractly represents a system as input. Metrics are based on the Presentation Model of OO-Method, which is composed of interaction conceptual primitives divided into three levels:

1. **Level 1**: Hierarchical Action Tree (HAT). This primitive helps the designer to abstractly define how the end user can access the system's functionality.
2. **Level 2**: Interaction Unit (IU). This primitive allows particular scenarios of the user interface to be described. Three types of Interaction Units can be defined:
    - Instance Interaction Unit (IIU): This represents a particular instance from an object class.
    - Population Interaction Unit (PIU): This represents a set of different instances of the same class.
    - Service Interaction Unit (SIU): This represents a dialog in which the end user can launch a service. The user can insert parameters for the service in this IU.
3. **Level 3**: Elementary Patterns (EP). Level 3 defines those primitives that make up and restrict the Interaction Units. There are 9 EPs:
    - Display Set: This specifies which attributes of an object can be shown. It is associated to a PIU or to an IIU.
    - Action: This specifies which services can be launched when an instance of an object is selected.
    - Navigation: This specifies which related objects can be accessed when an instance of an object is selected.
    - Supplementary Information: This specifies more information apart from the object identifier. This information aims to help the user to identify an object.
    - Filter: By specifying this primitive, different values can be entered in order to list a group of objects with some common criteria.
    - Order Criteria: Whenever this primitive applies to a list of objects, the user can list them using different criteria and order.
    - Introduction: This captures the relevant aspects of data to be entered by the end-user.
    - Defined Selection: This enables the definition (by a list) of a set of valid values for an associated model element.
    - Argument Grouping: This defines the way in which input elements are presented to the end-user, allowing these input arguments to be arranged in groups and subgroups.

- Dependency: This enables dependency relationships between the values of two input elements from the same service.
- Preload: This allows us to specify that the selection of an object as an input element will be carried out without changing the interaction context.

3.2. *Metrics definition*

In order to be able to measure the usability of internal attributes, we first need to define the metrics. The metrics are specific to an MDD method since they are defined using conceptual primitives of the underlying MDD models. In Table 1, we list the definition of the metrics that we have defined for the conceptual primitives of OO-Method. As we indicated above, even though these metrics are specific to OO-Method, the concept of each one can be applied to any MDD method with similar conceptual primitives. Table 1 shows the metrics grouped by sub-characteristics and attributes extracted from the work of Fernández [4].

Table 1. List of proposed metrics for OO-Method

Uundestarndability	**Information density**: The degree in which the system will display/demand the information to/from the user in each interface. 1. The average of input elements per group of elements: $\forall x \in$ Group of input elements: $\frac{\sum_{i=1}^{n} Input\ Elements(x_i)}{n} = ID1$ The *Input Elements* function returns the number of input elements
	2. The average of actions per IU: $\forall x \in$ (PIU ∪ IIU): $\frac{\sum_{i=1}^{n} Action(x_i)}{n} = ID2$ The *Action* function returns the number of actions.
	3. The average of elements in a visualization display per IU: $\forall x \in$ (PIU ∪ IIU): $\frac{\sum_{i=1}^{n} Attributes\ of\ visualization\ display(x_i)}{n} = ID3$ The *Attributes of visualization display* function returns the number of attributes in a visualization display.
	4. The average of navigation elements per IU: $\forall x \in$ Navigation element, $\forall y \in$ PIU: $\frac{\sum_{i=1}^{n} x_i}{\sum_{j=1}^{m} y_i} = ID4$
	5. The average of filters per IU: $\forall x \in$ PIU with at least 1 filter, $\forall y \in$ PIU: $\frac{\sum_{i=1}^{n} x_i}{\sum_{j=1}^{m} y_i} = ID5$
	6. The average of input elements per filter:

	$\forall x \in$ Input element in a filter, $\forall y \in$ Filter: $\frac{\sum_{i=1}^{n} x_i}{\sum_{j=1}^{m} y_i} = ID6$ 7. The average of order criteria per IU: $\forall x \in$ PIU with at least 1 order criteria, $\forall y \in$ PIU: $\frac{\sum_{i=1}^{n} x_i}{\sum_{j=1}^{m} y_i} = ID7$ 8. The maximum number of elements per IU Pagination cardinality= *ID8*
	Labeling meaning: The level of meaning that the labels will have for the user. 1. The percentage of attributes defined with an alias: $\forall x \in$ Attribute with defined alias, $\forall y \in$ Attribute: $\frac{\sum_{i=1}^{n} x_i}{\sum_{j=1}^{m} y_i} LME$
	Brevity: The level of cognitive effort of the user. This attribute measures the amount of information that the user must read or write in each interface. 1. The minimum distance between two IUs: $\forall x \forall y \in$ (PIU ∪ IIU): $Minimum\ Distance\ (x, y) = BR1$ The *Minimum Distance* function returns the minimum distance between two IUs. 2. Elements that avoid the navigation to other IU to get information by means of ListBoxes with preloaded elements: $\forall x \in$ (object valued input element ∪ object valued filter element): $\frac{\sum_{i=1}^{n} Preload(x_i)}{n} = BR2$ The *Preload* function returns the number of input elements that will be shown in a preloaded list.
	Initial values completion: The percentage of input elements that will display a default value: 1. The percentage of input elements with a default value definition: $\forall x \in$ Input element with default value, $\forall y \in$ Input element: $\frac{\sum_{i=1}^{n} x_i}{\sum_{j=1}^{m} y_i} = IVC$
	Message quality: The average message quality. A good message informs the user about the reasons for the error and suggests possible solutions. 1. The average of words per error message: $\forall x \in$ Error message: $\frac{\sum_{i=1}^{n} Length(x_i)}{n} = MEQ$ The *Length* function returns the number of words in a text.
	Navigability: The level of facilities that the system will provide to navigate throughout several interfaces.

	1. The maximum distance between two IUs: $\forall x \forall y \in (\text{PIU} \cup \text{IIU})\text{: } Maximum\ Distance\ (x,y) = NV1$ The *Maximum Distance* function returns the maximum distance between two IUs. 2. Number of elements of the main menu: $\forall x \in \text{Element of the HAT: } \sum_{i=1}^{n} x_{\text{i}} = NV2$ 3. The average of navigations per PIU and IIU: $\forall x \in \text{Navigation, } \forall y \in (\text{PIU} \cup \text{IIU})\text{: } \frac{\sum_{i=1}^{n} x_{\text{i}}}{\sum_{j=1}^{m} y_{\text{i}}} = NV3$
Learnability	**Action determination**: The level of help that the user will receive before triggering an action. 1. Percentage of non-default labels: $\forall x \in (\text{Service } \cup \text{ Filter } \cup \text{ Order criteria})\text{: } \frac{\sum_{i=1}^{n} Non-default\ label(x_{\text{i}})}{n} = AD1$ The *Non-default label* function returns the number of non-default labels. 2. Percentage of object valued input elements with supplementary information: $\forall x \in \text{Supplementary information, } \forall y \in \text{Object valued input element:}$ $\frac{\sum_{i=1}^{n} x_{\text{i}}}{\sum_{j=1}^{m} y_{\text{i}}} = AD2$
Operability	**Order consistency**: The percentage of visual elements that will be displayed with the same order. 1. The percentage of IUs to create and modify an object that have the same order for input elements: $\forall x \in \text{Input fields in the same order (create, modify),} \forall y \in \text{Classes (create, modify):}$ $\frac{\sum_{i=1}^{n} x_{\text{i}}}{\sum_{j=1}^{m} y_{\text{i}}} = OC1$ 2. The percentage of IUs with the same order in common services (create, modify, delete): $\forall x \in \text{Common action (create, modify, delete): } \frac{\sum_{i=1}^{n} Ordered(x_{\text{i}})}{n} = OC2$ The *Ordered* function returns the number of elements that are ordered. 3. The percentage of IUs that ask for data and show them in the same order: $\forall x \in \text{Fields related to a IU,} \forall y \in (\text{PIU} \cup \text{IIU} \cup \text{UIS})\text{: } \frac{\sum_{i=1}^{n} Ordered(x_{\text{i}})}{\sum_{j=1}^{m} y_{\text{i}}} = OC3$
	Label consistency: The percentage of visual elements that will use the same label

	to identify a field or an action in several interfaces. 1. The percentage of navigations that appear in several IUs with the same label $\forall x \in$ Navigations with repeated target: $\frac{\sum_{i=1}^{n} Same\ alias(x_i)}{n} = LC1$ The *Same alias* function returns the number of elements with the same alias 2. The percentage of services that appear in several IUs with the same label: $\forall x \in$ Services repeated in different IUs: $\frac{\sum_{i=1}^{n} Same\ alias(x_i)}{n} = LC2$ 3. The percentage of services to create, modify and delete that appear with a similar notation: $\forall x \in$ Services to create, modify and delete : $\frac{\sum_{i=1}^{n} Similar\ notation(x_i)}{n} = LC3$ The *Similar notation* function returns the number of elements with a similar notation. 4. The percentage of attributes that appear in several IUs with the same label: $\forall x \in$ Attributes repeated in different IUs: $\frac{\sum_{i=1}^{n} Same\ alias(x_i)}{n} = LC4$ 5. The percentage of filter variables that appear in several IUs with the same label: $\forall x \in$ Filter variable repeated in different IUs: $\frac{\sum_{i=1}^{n} Same\ alias(x_i)}{n} = LC5$
	Error prevention: The percentage of input elements with limited possible values that will be displayed in a ListBox. 1. The percentage of enumerated input elements that use the primitive which represents a list: $\forall x \in Input\ element\ with\ limited\ values: \frac{\sum_{i=1}^{n} List(x_i)}{n} = ERP$ The *List* function returns the number of input elements that will be displayed in a list.

3.3. *Indicator definition*

The metrics defined in the previous section provide a numerical value that must be interpreted. Therefore, these metrics must be normalized since each metric is measured in a different scale. The use of indicators is a mechanism to normalize numerical values obtained by metrics [22]. The normalization process consists of assigning a categorical value to each numerical value. The possible categorical values are: Very Good (VG), Good (G), Medium (M), Bad (B), and Very Bad (VB). We have defined five numerical ranges associated to each metric in order to assign an ordinal value to each range.

The ranges used to define the indicators have been built from the usability guidelines and heuristics described in existing studies. These studies give hints for defining the ranges to interpret the numerical values of each metric. Next, we detail the numeric values that each metric must present to be considered with a "Very Good" usability value according to previous studies. This information will be used later to define the indicator ranges.

- **Understandability**:
  - **Information density**: Several usability guidelines recommend interfaces that are not too dense [23][24]. We have defined the maximum number of elements to maintain a good balance between information density and the use of white space: 15 input elements per group of widgets (ID1); 10 actions per interface (ID2); 7 elements in a display set (ID3); 9 navigations per interface (ID4); 90% of interfaces should have at least 1 filter (ID5); 3 filter variables per filter (ID6); 85% of interfaces should include order criteria (ID7); 20 instances at most for interface (ID8).
  - **Labeling meaning**: Guidelines recommend using clear, descriptive and meaningful labels [24] [25]. These features do not usually appear in default labels of display sets, only in labels defined by the analyst. Therefore, we state that more than 95% of display set labels should be non-default labels in order to have very good meaning (LME).
  - **Brevity**: Some studies have demonstrated that the human memory has the capacity to retain a maximum number of 3 different scenarios [26]. Therefore, each reachable context requiring more than three contexts decreases usability (BR1). Moreover, at least 90% of object-valued input elements should include the Preload Elementary Pattern (BR2).
  - **Initial values completion**: Some guidelines such as [27][26] recommend using default values as much as possible. Without feedback from the end-user, it is difficult to predict whether an argument should have a default value in the analysis step. Nevertheless, we can state that if the analyst defines some default values, system usability increases even though the number of default values is small. Therefore, we state that at least 20% of input elements should have a default value (IVC).
  - **Message quality:** Error messages should not contain more than 15 words, according to [27] (MEQ).
  - **Navigability:** According to [27], using more than three navigations to reach a specific context decreases usability (NV1). Moreover, system functionality should be organized into submenus so that no more than seven menu items [28] are shown to the user (NV2). Finally, if a context has many navigations, the user must use the scroll [27] because in OO-Method systems, no more than seven navigation buttons fit in an interface (NV3).
- **Learnability:**
  - **Action determination**: Action labels should be clear, descriptive and meaningful like display set labels [24][25]. Therefore, we state that more than 95% of action

labels should be non-default labels to get an excellent usability (AD1). Moreover, at least 95% of object valued input elements should have the Elementary Pattern called Supplementary Information to help the user to know which information will be used by the input elements to execute the action (AD2).

- **Operability**:
  - **Order consistency:** Each page of the system must share a consistent layout according to [25]. We state that at least 95% of the interfaces for creating and modifying information should have the input elements in the same order (OC1). Moreover, 95% of interfaces should display the actions create, modify, and delete in the same order (OC2). Finally, 95% of interfaces that ask for data should show their input elements in the same order as they are shown in interfaces that display the value (OC3).
  - **Label consistency**: The same word or phrase must be used consistently to describe the same item throughout the entire system [23]. Therefore, we state that at least 90% of repeated items must have the same label in the entire system to get very good usability. The items considered in this rule are: navigations (LC1), services (LC2), attributes (LC4), and filter variables (LC5). Moreover, common services like create, modify, and delete should have a similar label in at least 90% of the interfaces (LC3).
  - **Error prevention**: The system must provide mechanisms to keep the user from making mistakes [20][23]. One way to avoid mistakes is the use of ListBoxes for enumerated values. We state that at least 90% of enumerated values must be shown in a ListBox to improve usability (ERP).

According to usability guidelines, Table 2 shows the list of indicators that we have defined. Using the usability values that we have considered as “Very Good”, we have estimated the value to consider the usability as “Very Bad”. Once we have both extremes, we have distributed the values “Good”, “Medium” and “Bad” equitably between these extremes. Table 2 defines the range to consider the metric outcome as Very Good (VG), Good (G), Medium (M), Bad (B), or Very Bad (VB) for each metric.

Table 2. List of proposed indicators

Metric	VG	G	M	B	VB
ID1	ID1 ≤ 15	15< ID1 ≤20	20<ID1 ≤25	25< ID1 ≤30	ID1 >35
ID2	ID2 ≤ 10	10< ID2 ≤13	13<ID2 ≤16	16< ID2 ≤19	ID2 >19
ID3	ID3 ≤ 7	7< ID3 ≤10	10<ID3 ≤13	13< ID3 ≤16	ID3 >16
ID4	ID4 ≤ 9	9< ID4 ≤12	12<ID4 ≤15	15< ID4 ≤18	ID4 >18
ID5	ID5 ≥ .90	.90>ID5 ≥.80	.80>ID5 ≥.70	.70> ID5 ≥.60	ID5<.60
ID6	ID6 ≤ 3	3< ID6 ≤5	5<ID6 ≤7	7< ID6 ≤9	ID6 >9
ID7	ID7 ≥ .85	.85 >ID7 ≥.70	.70>ID7 ≥.55	.55>ID7 ≥.40	ID7<.40
DI8	DI8 ≤ 20	20< DI8 ≤30	30< DI8 ≤40	40< DI8 ≤50	DI8 >50
LME	LME ≥ .95	.95>LME ≥.85	.85>LME ≥.75	.75>LME ≥.65	LME <.65
BR1	BR1 ≤ 2	2< BR1 ≤4	4< BR1 ≤5	5< BR1 ≤6	BR1 >6

Metric	VG	G	M	B	VB
BR2	BR2 ≥ .90	.90>BR2≥.80	.80>BR2≥.70	.70>BR2≥.60	BR2 <.60
IVC	IVC ≥ .20	.20>IVC≥.15	.15>IVC≥.10	.10>IVC≥.5	IVC <.5
MEQ	MEQ≤ 15	15< MEQ≤25	25< MEQ≤35	35< MEQ≤45	MEQ >45
NV1	NV1 ≤ 3	4< NV1≤6	6< NV1≤8	8< NV1 ≤10	NV1 >10
NV2	NV2 ≤ 7	7< NV2≤10	10< NV2≤13	13< NV2≤16	NV2 >16
NV3	NV3≥ .90	.90 >NV3≥.80	.80>NV3≥.70	.70>NV3≥.60	NV3 <.60
DA1	DA1≥ .95	.95> DA1≥.85	.85> DA1≥.75	.75> DA1≥.65	DA1<.65
DA2	DA2≥ .95	.95> DA2≥.85	.85> DA2≥.75	.75> DA2≥.65	DA2<.65
OC1	OC1 ≥ .95	.95 >OC1≥.90	.90>OC1≥.85	.85>OC1≥.80	OC1 <.75
OC2	OC2 ≥ .95	.95 >OC2≥.90	.90>OC2≥.85	.85>OC2≥.80	OC2 <.75
OC3	OC3 ≥ .95	.95 >OC3≥.90	.90>OC3≥.85	.85>OC3≥.80	OC3 <.75
LC1	LC1 ≥ .90	.90>LC1≥.80	.80>LC1≥.70	.70>LC1≥.60	LC1 <.60
LC2	LC2 ≥ .95	.95>LC2≥.90	.90>LC2≥.85	.85>LC2≥.80	LC2 <.80
LC3	LC3 ≥ .90	.90>LC3≥.80	.80>LC3≥.70	.70>LC3≥.60	LC3 <.60
LC4	LC4 ≥ .90	.90>LC4≥.80	.80>LC4≥.70	.70>LC4≥.60	LC4 <.60
LC5	LC5 ≥ .90	.90>LC5≥.80	.80>LC5≥.70	.70>LC5≥.60	LC5 <.60
ERP	ERP ≥ .90	.90>ERP≥.80	.80>ERP≥.70	.70>ERP≥.60	ERP <.60

3.4. *Grouping definition*

In this step, we obtain a usability level for each attribute, each sub-characteristic and the overall usability of the system. Therefore, the grouping method is applied three times: to group the indicators by attribute; to group the attributes by sub-characteristic; to group the sub-characteristics to obtain the overall usability level. Next, we are going to describe how to perform these groupings. Each grouping consists of three steps:

1. **Convert ordinal values into numeric values**: We turn the ordinal values obtained after applying indicators into numbers. Ordinal values are values that can be ordered, and, therefore, they can be transformed into numbers within a limited range [29]. The value VG turns into 5, G into 4, M into 3, B into 2, and VB into 1.
2. **Calculate the average**: Each grouping consists of calculating the average of the items to be grouped. We calculate the average of the elements to be grouped.
3. **Convert numeric values into ordinal values**: We turn the numeric value obtained in the average into ordinal values. To do this, we divide the possible numbers from 1 to 5 into 5 ranges. From 1 to 1.80 we assign the value VB to the outcome of the aggregation; from 1.81 to 2.6 we assign the value B; from 2.61 to 3.4 we assign the value M; from 3.41 to 4.2 we assign the value G; and from 3.41 to 5 we assign the value VG.

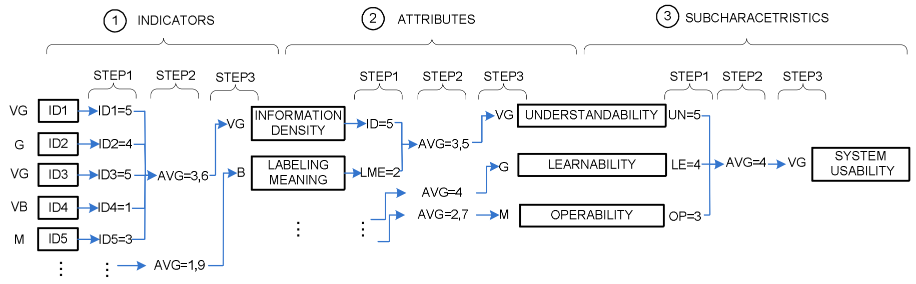

Fig. 3 Example of grouping

These three steps are applied for each grouping. Groupings are performed bottom-up from indicators until the overall system usability is reached. Fig. 3 shows an example of groupings from Information Density indicators until the overall system usability is reached. The first grouping consists on grouping by attribute. To do this, we apply the three steps defined above. We turn each indicator into a numerical value, calculate the average from all the indicators, and convert the result of the average into an ordinal value (VG in Fig. 3). This outcome is the value for the Information Density attribute. Secondly, once we have the level of usability for each attribute, we group attributes by sub-characteristic. To perform this grouping, the ordinal values of the attributes are converted into numbers, the average of these numbers is calculated, and, finally, this average is turned into ordinal values. In the example, Understandability has the value VG. Finally, we group the usability value of each sub-characteristic to obtain the usability of the overall system. To do this, we convert the ordinal values of the sub-characteristics into numbers, calculate their average, and convert this result into an ordinal value. In Fig. 3, the overall usability is VG.

It is important to note that the level of aggregation depends on the analyst's preferences. In some cases, the degree of usability per attribute can be enough and the aggregation by sub-characteristics is not required.

4. An Experiment to Evaluate the Proposal

This section explains the experiment design that we used to evaluate our method for an early usability evaluation. The evaluation consisted of a usability test performed by 18 users who interacted with two systems developed with OLIVANOVA. The aim of the experiment was to compare the usability measure obtained by our proposed method and the level of usability perceived by the end-user.

4.1. *Objectives*

According to the Goal/Question/Metric template [30], the objective of the experiment was to:

Analyze	internal measures of usability
For the purpose of	evaluating the internal measures of usability

With respect to	the degree of coherence with regard to users' perception
From the viewpoint	of the researcher
In the context of	end-users evaluating web applications that were automatically generated from conceptual models

From this objective, the following research question was derived:

- RQ1: Is there significant coherence between the users' perception about the usability of the final applications and the value obtained with the early evaluation method?

We identified the following hypotheses related to RQ1:

- H_0: There is not a significant difference between the usability obtained with the early evaluation (EE) method and the usability perceived by the end-user (PU).

$$H_0: \mu_{EE} = \mu_{PU}$$

- H_1: There is a significant difference between the usability obtained with the early evaluation (EE) method and the usability perceived by the end-user (PU).

$$H_1: \mu_{EE} \neq \mu_{PU}$$

4.2. *Subjects and objects*

The subjects were 18 undergraduate students from the Technical University of Valencia. There were differences in the application domains commonly used by subjects. Some subjects were more used to interact with e-commerce applications, others were more used to interact with social applications, and some others were more used to interact with travel agency applications. Moreover, their ages ranged between 20 and 29 years old. These subjects were volunteers recruited from the "Web Applications" course of the Department of Computer Science. The subjects had a high level of knowledge in the web application domain used in this study; however, they did not have experience in conceptual modeling, and more specifically, in OO-Method.

The objects used were two case studies with the same level of complexity: Rent-a-Car and Internet Movie DataBase (a web application based on the IMDB web site). These two web applications and instruments used in the empirical study can be found in [31]. The Rent-a-Car web application is a system for renting cars on Internet, and IMDB is a huge database front-end with information about movies. Each case study was specified using OO-Method conceptual models, and the final system was automatically generated from these models. To minimize the influence of subjective aspects, both web applications had the same visual appearance (e.g., color background, font type, etc.). However, the IMDB system was developed to get a good level of usability, whereas the Rent-a-Car system was designed to get poor usability.

4.3. *Identification of variables*

We identified two types of variables:

- **Response variables**: Variables that correspond to the outcomes of the experiment [32]. In this work, usability was the target of the study, which was measured in terms of learnability, operability, and understandability. Each one of these sub-characteristics was measured by means of a set of metrics listed in Table 1.
- **Factors**: Variables that affect the response variables. The **usability evaluation method** was identified as a factor that affects the response variable. This variable had two alternatives: 1) early evaluation of usability from conceptual models without end-users; 2) usability evaluation with end-users. Moreover, there was another factor to represent the **level of usability** of the system used in the experiment, since IMDB was designed with better usability than Rent-a-Car.

4.4. *Instrumentation*

The instruments used to carry out the experiment were [31]:

- **A demographic questionnaire**: A set of questions to know the level of the users' experience in web applications similar to IMDB and Rent-a-Car.
- **Tasks**: A list of tasks for each web application that the user must carry out. Tasks definition is a mechanism to guarantee that the user interacts with the most significant contexts of the web application and that all the users interact with the same contexts.
- **A survey**: A list of 27 questions defined to capture the users' perceptions in a 5-Likert scale format. Each question refers to a metric defined to measure a usability internal attribute (one question per metric). This survey was made to obtain end-users' impressions for each metric. In this way, we could compare the usability values obtained by means of metrics with the users' perception. We could not use an existing questionnaire since we needed a specific question for each attribute.
- **A spreadsheet**: To measure the system usability using metrics and indicators with the conceptual model, according to the early usability evaluation method we propose. The spreadsheet was used to accelerate the metrics calculation based on conceptual primitives. This calculus was carried out by two experts in OO-Method and measurement.

4.5. *Design process*

Fig. 4 depicts the design process that represents the strategy used for empirically evaluating our approach. The process started with a demographic questionnaire to capture the user background in the domain. Next, the user interacted with the IMDB or with the Rent-a-Car system. Half of the users started the interaction with IMDB and the other half started with Rent-a-Car. The interaction consisted of performing six tasks, and we only considered users that finished the six tasks successfully. Once the users finished all the tasks, they filled in a survey to capture their usability perception with regard to the system. Next, the process was repeated again with the other system. The tasks for each system were different, but the questions of the survey were the same for both of them. Once we captured the users' perception, two usability experts applied our proposal to measure the usability studying the IMDB and Rent-a-Car conceptual models. For this aim, they used the spreadsheet to facilitate the calculus. The outcomes of the surveys were compared with the outcomes of the evaluation based on conceptual models. The comparison was done with a statistical analysis.

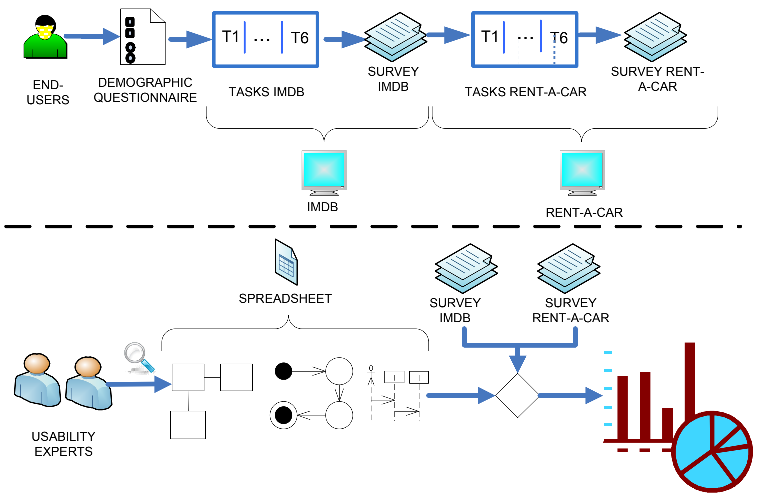

Fig. 4. The strategy used for empirically evaluating the proposed usability evaluation method.

4.6. *Validity evaluation*

It is important to consider a validity evaluation in order to ensure that the experimental results are valid for the target population. In this section, we discuss the threats [33] that were identified in our experiment and how to minimize them.

4.2.1. Conclusion Validity

This type of validity concern is related with the relationship between the treatment and the outcomes of the experiment. Our evaluation was threatened by **Random heterogeneity of subjects**. This threat appears when, within each user group, some users have more experience than others. In our experiment, the experience is related to the use of web applications. This threat was resolved with a demographic questionnaire that allowed us to evaluate the knowledge and experience of each participant beforehand. The demographic questionnaire revealed that most users had experience with systems of this type.

4.2.2. Internal validity

This type of validity concern is related with the influences that can affect the factors with respect to causality, without the researcher's knowledge. Our evaluation had the threat called **Maturation**: the effect that users react differently as time passes (because of boredom or tiredness). We solved this threat by establishing a limit of one hour for the whole test. Another internal validity threat that our evaluation had was **Instrumentation**: even though tasks and questionnaires are the same for all subjects, a wrong interpretation of the task may affect the results. This threat was minimized by the collaboration of a lecturer (not involved in this paper) who answered any question throughout the experiment. This lecturer made sure that the user correctly understood what was being asked before starting the task. Moreover, the instruments had been verified in advance with a small group of people.

4.2.3. Construct validity

Threats to construct validity generalize the results of the experiment to the concept or theory behind the experiment. We considered the threat called **Inadequate preoperational explanation of constructs**. This threat means that the constructs are not sufficiently defined and, hence, the experiment cannot be sufficiently clear. We used an inter-item correlation analysis to evaluate the construct validity of the response variables. To do this, we used two criteria proposed by Campbell and Fiske [34]: *Convergent validity (CV),* which refers to the convergence among different indicators used to measure a particular construct; and *Discriminant validity (DV),* which refers to the divergence of indicators used to measure different constructs. The average of DV should be lower than the average of CV. The results of the validity analysis for each construct show that the CV value was higher than the DV value (see [31]), except for ID2 and ID7, which were not included in the analysis for this reason. In addition, we also conducted a **reliability analysis** on the survey. The reliability was conducted using the Chronbach alpha for every question of the survey. The value obtained for the whole questionnaire was 0.904, which is a very good value for reliability. The reliability for the response variables were: 0.403 for learnability, 0.839 for understandability, and 0.756 for operability. These values are also very good for an academic experiment.

4.2.4. External validity

This type of validity concern is related to conditions that limit our ability to generalize the results of the experiment to industrial practice. Our evaluation might suffer from **Interaction of selection and treatment**: the subject population might not be representative of the population we want to generalize. We used a confidence interval where conclusions were 95% representative. This means that if conclusions followed a normal distribution, results would be true for 95% of the times the evaluation would be repeated. Moreover, the demographic questionnaire indicated that most of the users were familiar with both systems, which reduces the generality of the results to other systems.

5. Data Analysis

This section compares the users' usability perception with the outcomes of the early usability evaluation for each metric. Firstly, we compare the average of users' usability perception with regard to the level of usability extracted from conceptual models. Fig. 5

depicts this comparison for the IMDB system. We can state that the users' perception is more constant (less peaks) than the early evaluation. However, the trend is similar for most metrics. For example, there are similar values for ID8, NV3, AD1, AD2, OC3, LC1, LC2, LC3, LC4,and LC5.

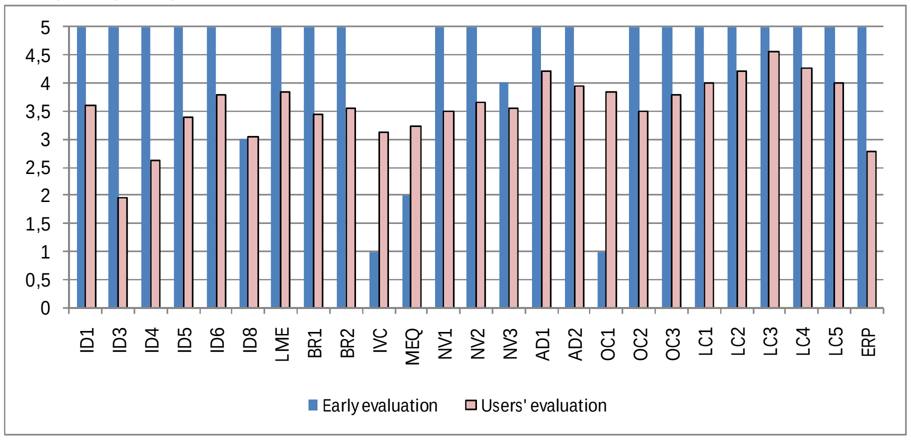

Fig. 5. Comparison of users' perception with the early evaluation for the IMDB system

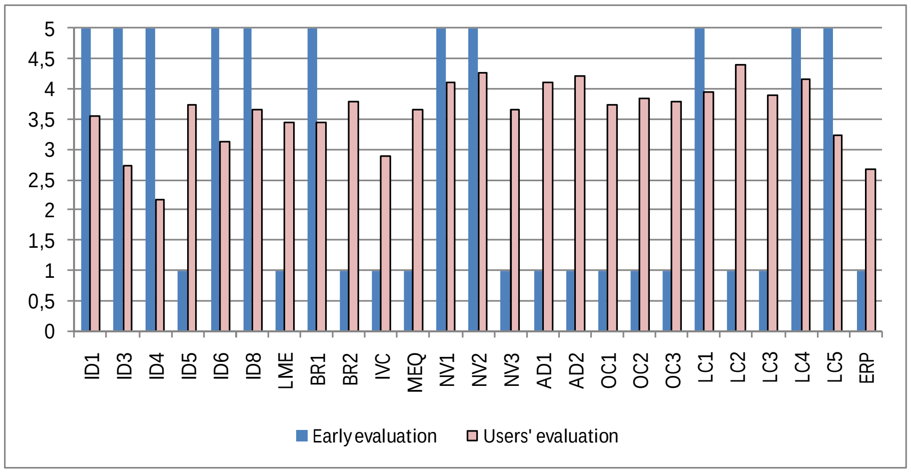

Fig. 6 Comparison of users' perception with the early evaluation for the Rent-a-Car system

Fig. 6 compares the early usability evaluation with the user's perception for the Rent-a-Car system. It shows how the level of usability using the early evaluation method is lower than the values obtained with the same method in IMDB (Fig. 5). As with the IMDB analysis, we can state that the users' usability perception does not fluctuate as much as the early usability evaluation. For this case, in general, the tendency of the early usability evaluation is not the same as the users' perception. For example, there are important differences for ID4, ID5, LME, BR2, IVC, MEQ, NV3, AD1, AD2, OC1, OC2

OC3, LC2, LC3 and ERP. There are some metrics with similar values, such as ID8, NV1, NV2, LC1 and LC4.

In order to study the comparison of factors in depth, we performed a statistical study called *One sample T-test*. This test is a statistical procedure that is used to determine the mean difference between a sample and the value of a population mean. For our study, the sample was composed of the evaluation performed with 18 subjects (the experiment) and the population mean was the level of usability obtained from metrics and indicators (early evaluation). The One sample T-test is a parametric technique; therefore, the first step is to verify whether or not the response variables follow a normal distribution. To achieve this task, we applied a one-sample Kolmogorov-Smirnov (K-S) test. The results of the K-S test show that all the response variables were normally distributed.

The One sample T-test works with a null hypothesis that states that there are no significant differences between the population mean and the sample. We verified this hypothesis for IMDB and for the Rent-a-Car system. Table 3 shows, for each metric, the levels of significance obtained in the IMDB system. When the significance is higher than 0.005, we can accept the null hypothesis (there is no difference between the early evaluation and the users' perception). Therefore, we can state that the early evaluation fits in with the users' perception for ID8, NV3 and LC3.

Table 3. Significance of the One sample T-test for IMDB with a 5-point scale

ID1	ID3	ID4	ID5	ID6	ID8	LME	BR1	BR2	IVC	MEQ
.000	.000	.000	.000	.003	,868	.000	.000	.000	.000	.002
NV1	NV2	NV3	AD1	AD2	OC1	OC2	OC3	LC1	LC2	LC3
.000	.000	.104	.003	.001	.000	.000	.000	.000	.003	.007
LC4	LC5	ERP								
.001	.000	.000								

Table 4 shows the levels of significance of the One sample T-test for the Rent-a-Car system. In this case, only two metrics allow us not rejecting the null hypothesis (there is no difference between the early evaluation and the users' perception): ID8 and NV2. For the other metrics, there is not a correspondence between the early evaluation and the users' perception.

Table 4. Significance of the One sample T-test for Rent-a-Car with a 5-point scale

ID1	ID3	ID4	ID5	ID6	ID8	LME	BR1	BR2	IVC	MEQ
.000	.000	.000	.000	.000	.035	.000	.000	.000	.000	.000
NV1	NV2	NV3	AD1	AD2	OC1	OC2	OC3	LC1	LC2	LC3
.001	.008	.000	.000	.000	.000	.000	.000	.001	.000	.000
LC4	LC5	ERP								
.003	.000	.000								

The outcomes of the One sample T-test indicate that the level of usability extracted from each metric does not correspond with the users' perception. One reason could be that the indicators do not work properly to differentiate similar usability levels, such as

VG and G, B and VB. In order to verify this statement, we converted the 5-point Likert scale of the indicators into a 3-point Likert scale. To do this, we grouped the values VG and G, and, the values B and VB into the same group. Since the value M did not undergo any change, we now have three different levels of usability: Good (G), Medium (M) and Bad (B).

Table 5 shows the outcomes of applying the One sample T-test to the indicators converted into a 3-point Likert scale for the IMDB system. According to the level of significance, we can state that the values provided by the metrics correspond with the users' perception for: ID1, ID6, ID8, LME, NV2, NV3, AD1, AD2, OC3, LC1, LC2, LC3, LC4 and LC5.

Table 5. Significance of the One sample T-test for IMDB with a 3-point scale

ID1	ID3	ID4	ID5	ID6	ID8	LME	BR1	BR2	IVC	MEQ
.024	.000	.000	.003	.008	.805	.055	.002	.004	.000	.000
NV1	NV2	NV3	AD1	AD2	OC1	OC2	OC3	LC1	LC2	LC3
.003	.008	.008	.163	.016	.000	.004	.016	.015	.056	.331
LC4	LC5	ERP								
.083	.029	.000								

Table 6 shows the significance for the Rent-a-Car system. According to the level of significance, we can state that the values provided by the metrics correspond with the users' perception for: ID1, NV1, NV2, LC1 and LC4.

Table 6. Significance of the One sample T-test for Rent-a-Car with a 3-point scale

ID1	ID3	ID4	ID5	ID6	ID8	LME	BR1	BR2	IVC	MEQ
.006	.000	.000	.000	.000	.004	.000	.004	.000	.000	.000
NV1	NV2	NV3	AD1	AD2	OC1	OC2	OC3	LC1	LC2	LC3
.056	.187	.000	.000	.000	.000	.000	.000	.030	.000	.000
LC4	LC5	ERP								
.104	.001	.004								

Next, we apply the grouping process to analyze the usability level for sub-characteristics and the overall system. This aggregation is based on the indicators with a 5-point Likert scale. Table 7 shows the outcomes after applying the grouping to the result of the indicators in the IMDB system. The same information is displayed in Table 8 for the Rent-a-Car system:

Table 7. Usability level for sub-characteristics and the overall system in IMDB

Learnability	**Understandability**	**Operability**	**Overall Usability**
VG	VG	VG	VG

Table 8. Usability level for sub-characteristics and the overall system in Rent-a-Car

Learnability	**Understandability**	**Operability**	**Overall Usability**
VB	VG	B	B

The grouping by sub-characteristic is also applied to the users' perception in order to compare these results with the outcomes of the early evaluation method (Table 7 and Table 8). This comparison was performed using the One sample T-test. The results of this test show that there is no relation between the values obtained from the early evaluation and the values extracted from users' perception.

In order to study the usability level by sub-characteristic in depth, we aggregated the indicators and the users' perception converted into a 3-point Likert scale. This conversion aims to mellow problems to differentiate between the values VG and G, and the values B and VB, such as we have done with the indicators previously (Table 5 and Table 6). Applying the One sample T-test to the results of the aggregation with this 3-point scale in the IMDB system, we found a significant relation between the users' perception and the early evaluation for: Learnability, Operability, and the overall system. For the Rent-a-Car system, we found a significant relation only for Understandability.

6. Results Interpretation and Discussion

The goal of our evaluation was to test the null hypothesis H_0: There is not a significant difference between the usability obtained with the early evaluation method and the usability perceived by the user. According to the statistical study detailed in the section above, we can conclude that H_0 is true for half of the metrics when we deal with indicators in a 3-point Likert scale.

Comparing the average of the users' perception with the early evaluation (Fig. 5 and Fig. 6), we can conclude that our proposed indicators do not differentiate correctly between the values VG and G, and the values B and VB. That is the reason why the usability level for users' perception is more constant than the early evaluation. Therefore, the indicators for VG and VB should be more restrictive than they currently are. This theory is also supported by the One sample T-test applied to a 3-point Likert scale. If we put together values of VG and G, and values of B and VB, we can conclude that the early usability predicts the users' perception in many cases. In addition, from Table 5 and Table 6, we can conclude that indicators work better to measure a good value of usability than to measure a bad value of usability. The IMDB system was designed with a good value of usability, and 56% of metrics matched the users' perception. However, the result of the early evaluation in the Rent-a-Car system, which was designed with poor usability, only fits in with 20% of users' perception. The theory that indicators work better with good usability levels is also supported by the aggregations performed in Table 7 and Table 8. This is because the values of VG were defined using usability guidelines and heuristics but the values of VB were estimated by the authors. These estimations must be corrected, such a way the indicators for the VB level will be even more restrictive than for the VG level.

The results of this experiment show that it is possible to predict the system usability using conceptual models. This early evaluation minimizes resources since the process can be automated and the evaluation results are obtained in few seconds. In the experiment, two usability experts performed the early evaluation using conceptual models manually with the help of a spreadsheet. However, once indicators are adjusted, we can define an automatic evaluator that takes as input conceptual models and determines the usability of the system without the participation of any expert. This is a clear advantage with regard to traditional evaluations that require investing time and material resources [35].

7. Conclusions

This paper presents a method for an early usability evaluation based on conceptual models. This evaluation method has the following properties: the evaluation can be performed automatically without user intervention, can be applied to any MDD method, and is based on conceptual primitives. The analyst can model the system, evaluate its usability in a short time, and modify the model in order to improve usability. This evaluation can be done without recording devices, or users, and it takes very little time since it is automatic. Moreover, the evaluation can be repeated infinite times, helping the incremental software development.

Before developing the tool to support the automatic evaluation, this paper has tested the proposal with end-users using OO-Method as example of MDD method. To do this, we have defined a set of metrics based on OO-Method conceptual primitives. The metrics are specific for OO-Method since they use particular conceptual primitives of OO-Method. However, the goal of each metric can be used in any other MDD method with enough expressiveness to represent the interaction with the user (such as the Presentation Model of OO-Method). Moreover, the indicators and the grouping process are also applicable to any other model-driven method.

The empirical evaluation of metrics and indicators conclude that indicators must be adjusted. The range to determine a value of very good usability and very bad usability should be more restrictive. The experiment has been a key factor to guide the improvement of our proposal. As future work, we plan to perform more experiments with other systems in order to define the ranges taking users' perception as input. These ranges will be more realistic than current ranges, which have been extracted from the literature for good usability values and have been estimated by the authors for bad usability values. In addition, we plan to build a repository of users' perceptions and the results of early usability evaluations in order to define a prediction model that determines usability in the early steps of software development. This prediction will be based on linear regressions. Once indicators will be improved, we also plan to automate the early evaluation process developing a tool that taking as input conceptual models, determine the usability of the system represented in these conceptual models.

It is important to mention that the early usability evaluation does not replace the evaluation with end-users, rather than complement each other. Early usability focuses on internal attributes, while external attributes can only be measured by end-users. The idea is to perform the early evaluation before generating the system. Once the usability problems for internal attributes have been solved, the analyst can carry out an evaluation with end-users. The evaluation of external attributes is needed to measure subjective attributes related to attractiveness and visual appearance.

Acknowledgments

We would like to thank Professor Joan Fons and his students from the Technical University of Valencia for their collaboration in the empirical evaluation. This work has been developed with the support of MITYC, MICINN, and GVA under the projects MyMobileWeb (TSI-020400-2010-118), PROS-Req (TIN2010-19130-C02-02), and ORCA (PROMETEO/2009/015), and co-financed with ERDF. We also acknowledge the support of the ITEA2 Call 3 UsiXML project under reference 20080026.

References

[1] B. W. Boehm, *Software Engineering Economics*. Upper Saddle River, NJ: Prentice Hall PTR, 1981.

[2] S. J. Mellor, A. N. Clark, and T. Futagami, Guest Editors' Introduction: Model-Driven Development, in *IEEE Software*, vol. 20, 2003, 14-18.

[3] ISO/IEC 9126-1 (2001), Software engineering - Product quality - 1: Quality model.

[4] A. Fernández, E. Insfrán, and S. Abrahão, Integrating a Usability Model into Model-Driven Web Development Process, in *Proc. of Web Information Systems Engineering (WISE)*, Vol. 5802, 2009, 497-510.

[5] C. Cachero, C. Calero, G. Poels, (2007). Metamodelling the Quality of the Web Development Process' Intermediate Artifacts. *Proc. of the 7th International Conference on Web Engineering (ICWE 2007)*, Como (Italy), Springer-Verlag. 2007, 74-89.

[6] I. Panach, N. Condori, F. Valverde, N. Aquino, O. Pastor, Towards an Early Usability Evaluation for Web Applications, in *Proc. of International Conference on Software Process and Product Measurement (MENSURA),* 2007, 67-76.

[7] O. Pastor, J. Molina, *Model-Driven Architecture in Practice*. (Valencia, Springer, 2007).

[8] CARE Technologies S.A. www.care-t.com .

[9] M. Genero, M. Piatini, E. Manso, Finding "early" indicators of UML class diagrams understandability and modifiability, in *Proc. of International Symposium on Empirical Software Engineering (ISESE'04)*, 2004, 207-216.

[10] G. Brajnik, Automatic web usability evaluation: what needs to be done?, in *Proc. of 6th Conference on Human Factors and the Web (HFWEB)*, Austin, TX, USA, 2000.

[11] P. Fraternali, P. L. Lanzi, M. Matera, and A. Maurino, Model-driven Web usage analysis for the evaluation of Web application quality, *Journal of Web Engineering*, vol. 3, 2004, 124 - 152.

[12] A. Lecerof, F. Paternò, Automatic Support for Usability Evaluation, *IEEE Transactions on Software Engineering*, vol. 24, 1998, 863-888.

[13] R. Chatley, J. Kramer, J. Magee, and S. Uchitel, Model-based Simulation of Web Applications for Usability Assessment, *Proc. of the International Conference of Software Engineering (ICSE)*, Portland, Oregon, 2003.

[14] A. Bajaj and R. Krishnan, CMU-WEB: a conceptual model for designing usable web applications, *Human computer interaction development and management: Idea Group Inc (IGI)*, IRM Press, 2002.

[15] P. Fraternali, M. Matera, and A. Maurino, WQA: an XSL Framework for Analyzing the Quality of Web Applications, *in Proc. of the 2nd International Workshop on Web Oriented Software Technology (IWWOST)*, Málaga, Spain, 2002.

[16] ISO 9241-11 (1998) Ergonomic requirements for office work with visual display terminals - Part 11: Guidance on Usability.

[17] L. Sorokin, F. Montero, and C. Märtin, Flex RIA development and usability evaluation, *in Proc. of the 1st International workshop on web usability and accessibility (IWWUA)*, Nancy (France), 2007, 447-452.

[18] A. Abran, A. Khelifi, W. Suryn, Usability Meanings and Interpretations in ISO Standards, *Software Quality Journal*, vol. 11, 2003. 325-338.

[19] A. Seffah, M. Donyaee, R. B. Kline, H. K. Padda, Usability measurement and metrics: A consolidated model, *Software Quality Journal*, vol. 14, 2006. 159-178.

[20] J. M. Bastien, Scapin, D., *Ergonomic Criteria for the Evaluation of Human-Computer Interfaces*, (Rapport technique de l'INRIA, France, 1993).

[21] F. García, F. Ruiz, M. F. Bertoa, C. Calero, M. Genero, L. Olsina, M. Martín, C. Quer, N. Condori, S. Abrahão, A. Vallecillo, M. Piattini, *An Ontology for Software Measurement*, 2004

[22] N. E. Fenton, *Software Metrics*: Pws Pub Co; 2nd edition, 1996.

[23] Information services & technology: http://ist.mit.edu/services/consulting/usability/guidelines

[24] Usability.gov: http://www.usability.gov/guidelines/index.html

[25] Userfocus: http://www.userfocus.co.uk/resources/guidelines.html

[26] M. E. Lacob, Readability and Usability Guidelines.Web: https://doc.telin.nl/dsweb /Get/Document-35439/ArchiMate%20D2.3%20Readability%20and%20Usability%20 Guidelines.pdf , 2003.

[27] M. Leavit, B. Shneiderman, Research-Based Web Design & Usability Guidelines, U.S. Government Printing Office. Web: http://www.usability.gov/pdfs/guidelines.html , 2006

[28] G. A. Miller, The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review,* Vol. 63, 1956, 81-97.

[29] Tutorial of SPSS: http://www.uni.edu/its/support/article/604

[30] Goal/Question/Measures: http://www.gqm.nl/

[31] Web of the experiment: http://hci.dsic.upv.es/earlyevaluation

[32] N. Juristo, A. Moreno, *Basics of Software Engineering Experimentation* (Kluwer Academic Publishers, Boston, 2001).

[33] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, *Experimentation in Software Engineering: An Introduction*, (Kluwer Academic Publishers, Sweden, 2000).

[34] D. T. Campbell and D. W. Fiske, Convergent and Discriminant Validation by the Multitrait-Multimethod Matrix, in *Psychological Bulletin*, vol. 56, 1959, 81-105.

[35] A. Dix, J. E. Finlay, G. D. Abowd, and R. Beale, *Human-Computer Interaction*: Prentice Hall, 2003.