A Quality Model in a Quality Evaluation Framework for MDWE Methodologies

F. J. Domínguez-Mayo, M. J. Escalona, M. Mejías, A. H. Torres
Departamento de Lenguajes y Sistemas Informáticos
University of Seville
Seville, Spain
fjdominguez@us.es, mjescalona@us.es, risoto@us.es, arturohtz@gmail.com

Abstract— **Nowadays, diverse development methodologies exist in the field of Model-Driven Web Engineering (MDWE), each of which covers different levels of abstraction on Model-Driven Architecture (MDA): CIM, PIM, PSM and Code. Given the high number of methodologies available, it is necessary to evaluate the quality of existing methodologies and provide helpful information to the developers. Furthermore, proposals are constantly appearing and the need may arise not only to evaluate the quality but also to find out how it can be improved. In this context, QuEF (Quality Evaluation Framework) can be employed to assess the quality of MDWE methodologies. This article presents the work being carried out and describes tasks to define a Quality Model component for QuEF. This component would be responsible for providing the basis for specifying quality requirements with the purpose of evaluating quality.**

Keywords: Model-Driven Web Engineering; Quality; Quality Model; Software Metrics; Methodologies; Model-Driven Engineering;

I. INTRODUCTION

Model-Driven Engineering (MDE)[35] is a paradigm of software development which consists of the creation of models closer to a particular domain rather than concepts or a specific syntax. The domain environment specific to MDE for web engineering is called Model-Driven Web Engineering (MDWE)[11]. The Object Management Group (OMG) has developed the standard Model-Driven Architecture (MDA) which defines an architecture platform for proposals based on the Model-Driven paradigm[1].

According to the OMG [29], the goals of MDA are portability, interoperability and reusability through architectural separation. The concept of platform independence appears frequently in MDA. Models may have the quality of being independent from the characteristics of any technological platform. By applying this paradigm, the lifecycle of a software system is completely covered, starting from requirements capture, passing through the generation of code, and up to the system maintenance.

MDA determines a minimum number of stages or levels of abstraction: Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM), and Code. However, research in this field is mainly oriented towards the CIM and PIM levels of abstraction.

In recent years, the growing interest in the internet has led to the generation of a high number of MDWE approaches which offer a frame of reference for the Web environment [11]. On the other hand, there are a high number of approaches without standard consensus [13][19][33], a lack in the use of standards, and scarcity of both practical experience and tool support. In the face of this situation, an important need to assess the quality of existing methodologies arises. In this paper, QuEF (Quality Evaluation Framework), an environment for the quality evaluation of Model-Driven Web methodologies based on MDA is proposed.

The paper is organized into the following sections. In Section II a general analysis of the situation is presented. Section III presents the problem, motivation and goal, and is intended to lay the basis of a framework that allows the evaluation of the quality of different methodological proposals. In Section IV concepts such as MDWE methodology and framework are explained and a short description of the components of the framework is given. In Section V, the Quality Model component for QuEF is defined and the stages for the definition of the Quality Model component for QuEF and a description of every component, structure and process to achieve the Quality Model are shown. In Section VI, an example of applying the Quality Model proposed with the NDT methodology is performed. Finally, in Section VII, a set of conclusions and contributions is laid out, and possible future work is given.

II. RELATED WORK

A. *Surveys*

There are many methodological approaches in the area of MDWE and numerous comparative studies [33], [31], [11],

1. http://www.omg.org

[19]. Along these lines, [33] must be considered, which specifically considers modelling concepts for their ubiquitous nature, together with an investigation of available support for Model-Driven Development in a comprehensive way, using a well-defined as well as fine-grained catalogue of more than 30 evaluation criteria.

B. *Quality Evaluation*

In [5], an approach is proposed to evaluate Web quality that provides all the elements which, according to the ISO/IEC 14598, are essential parts of a software quality evaluation, namely: (1) quality model, (2) a method of evaluation, (3) a software measurement process, and (4) supporting tools. The idea of developing an MDE framework for evaluating quality has been applied in various studies [25], [26], where it is stated that the quality of models is affected by the quality of modelling languages, tools, modelling processes, the knowledge and experience of modellers, and the quality assurance techniques applied. In other papers by the same authors, some related work on quality frameworks and requirements for their evaluation is presented. Furthermore, a 7-step process is proposed on how to define a quality framework adapted to MDE, which integrates quality engineering with quality evaluation. Existing quality models are discussed in their other papers before a metamodel is proposed for specifying quality models in the context of MDE. Along these lines, in yet another paper, these authors analyze the existing literature in order to extract model quality properties and to build a quality model with focus on the quality of models.

C. *Software Metrics*

In the literature there are numerous references to metrics [7], [20], [21], [10], [3], according to which, software measurement integration could be achieved by adopting the MDA approach. To this end, an approach is described in [14] for the management of measurement of software processes. From the methodological perspective, software measurement is supported by a wide variety of proposals, with the Goal Question Metric (GQM) method (Basili and Victor), the Practical Software & systems Measurement (PSM) methodology [23], and the ISO 15539 and IEEE 1061-1998 standards all deserving special attention. As far as web metrics quality is concerned, in [6] some important metrics proposed for web information systems are classified, with the aim of offering the user a global vision of the state of the research within this area. Studies on the quality of products and processes for the Web are rather recent and there are still no widely used metrics and models for different assessment and prediction purposes [1]. Although a few heuristics and metrics currently exist for the evaluation of a few quality characteristics such as usability [28], accessibility [18], user traffic and performance, most of them lack a sound and rigorous definition and validation framework.

With regards to the metrics model, an important study has been revealed in [1], which proposes a set of metrics for navigational models to analyze the web application quality in terms of size and structural complexity.

D. *Quality Models*

The term quality model is often used to refer to a set of quality attributes (also known as quality characteristics) and relations between them, with the aim of evaluating quality. Research on quality models has been going on for decades and different quality models have emerged. In [27] some of the best-known quality models are identified, one of which is McCall's hierarchical quality model which focuses on product quality, dividing it into the external view as seen by users (quality factors to specify) and the internal view as seen by the developers (quality criteria to build) [22]. By answering "yes" and "no" to questions related to quality criteria, one may measure to what extent a quality criterion is achieved. Another is Boehm's hierarchical quality model with three levels of quality characteristics: high-level characteristics form the users' perspective, intermediate characteristics which are software characteristics needed to achieve the high-level characteristics, and primitive characteristics which are the foundation for evaluation and defining metrics [2]. ISO standards are set out in [17], especially the ISO-9126 series with the hierarchical model of six quality factors and subcharacteristics related to each factor. The standard divides metrics into internal, external and quality-in-use metrics. Dromey's model, which has three main principles: quality attributes, product properties that are important for achieving quality attributes, and links between product properties and quality attributes. Dromey defines a five-step process for building product-specific quality models. Along these lines, P. Mohagheghi and V. Dehlen [9] propose a five-step approach in constructing a quality model.

III. PROBLEM, MOTIVATION AND GOAL

The main goal of this research is to lay the basis of a quality evaluation framework that facilitates the quality assessment of different methodological approaches under some specific criteria. The ability to measure these methodologies, facilities the assessment. The problem of measurement not only entails understanding the worth of a proposal, but also requires an objective criterion for improvement or the possibility of unifying criteria when designing new proposals in the future.

Today's modern web information systems are called to manage a huge amount of information which is difficult to develop and maintain. In this sense, there is a need for the suitable design of MDWE methodologies and effective tools. In this way, our work concentrates on evaluating and comparing existing proposals. The assessment is based on quality models which ensure the quality of proposals. As a result, a goal for the future could be to unify the criteria to decide on the use of a particular proposal in MDWE, or to improve the design of new proposals and the use of standards.

The use of an MDWE methodology and its influence on the final product quality is a crucial aspect under consideration. Nowadays, it is essential in the software industry to produce faster, cheaper software of higher quality. The use of a methodology based on MDA is fundamental to achieve this objective.

IV. QuEF (Quality Evaluation Framework), a Quality Evaluation Framework for MDWE methodologies

Methodology and *framework* are some words commonly used in software engineering literature and sometimes their meanings are not clear. Therefore, a brief definition of what these words mean in this work is given together with a short explanation for each QuEF component.

- *"Methodology":* We refer to an approach or methodology as a Model-Driven proposal for the development of web applications. It may provide a set of guidelines, techniques, processes and/or tools for the structuring of specifications, which are expressed as models. In this sense, only web modelling approaches which are based on MDA (Model-driven architecture) in the framework are considered.
- *"Framework":* Numerous definitions of the framework concept exist. In addition to this, a very broad definition has allowed the term to be used as a buzzword, especially in a software context. For example, the Java collection framework is not a software framework, but a library[2]. On the other hand, a software framework is a re-usable design for a software system (or subsystem). A software framework may include support programs, code libraries, a scripting language, or other software to help develop and glue together the different components of a software project. Various parts of the framework may be exposed through an API (Application Programming Interface). However, in this work, a quality evaluation framework is a basic conceptual structure composed of a set of components used to evaluate Model-Driven Web Engineering (MDWE) methodologies.

Therefore, a quality evaluation framework with a set of elements based on existing literature is proposed as shown in Figure 1, where four components for the evaluation of the quality of MDWE methodologies can be seen:

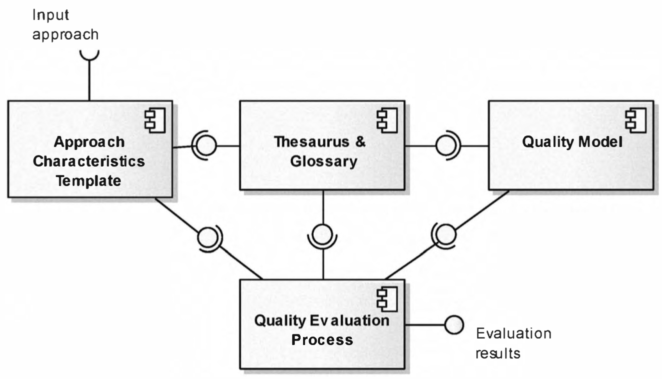

Figure 1. Component diagram of QuEF for MDWE methodologies.

- *Approach Characteristics Template:* This component has the responsibility for describing the input methodology characteristics to be evaluated.
- *Thesaurus & Glossary:* This component is responsible for improving the standardization of the access channel and communication between users of different MDWE methodologies.
- *Quality Model:* This component is responsible for providing the basis for the specification of quality requirements with the purpose of evaluating quality.
- *Quality Evaluation Process:* This component has the responsibility for carrying out the quality evaluation process.

V. The Quality Model component in QuEF

First of all, the meaning of the term Quality Model in this work is described, since various definitions in the literature can be found. Furthermore, the elements and relations between the elements of the Quality Model are explained.

According to (IEEE 610), the word "quality" has two definitions: "(1) The degree to which a system, component or process meets specified requirements. (2) The degree to which a system, component, or process meets customer or user needs or expectations." In this work, a Quality Model is a set of characteristics, subcharacteristics and metrics, quality factors, quality attributes and the relationships between them, which provides the basis for specifying quality requirements and evaluating quality. It may be defined as "conformance to requirements" and/or "fitness of use". In simple terms all the stakeholders must be well-informed of what is expected, what the subcharacteristics to be achieved are, which impact should be achieved on quality attributes, what the evaluation criteria are and how these criteria can contribute towards achieving the goal. In Figure 2, the Quality Model metamodel with the relations between the different elements in the Quality model are shown, and the elements are described and explained.

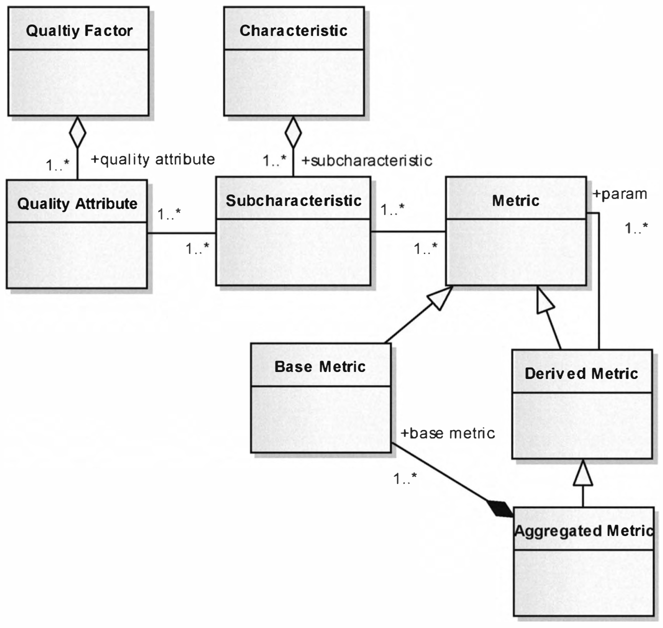

Figure 2. Quality Model Metamodel

- *Quality Factor:* This is a higher-level feature that affects an item's quality. For example, a quality factor could be Usability, Functionality, Portability, etc. In ISO 9126 the quality factors are classified according to three different views:
  - *External Quality:* which measures the software itself (ISO 9126-2)
  - *Internal Quality:* which measures the system behaviour (ISO 9126-3)
  - *Quality in Use:* which measures the effect of using the software in a specific context (ISO 9126-4)

  ISO 9126 describes ten quality factors, six that are common to internal and external quality and four that are specific to the quality in use. Each quality factor and attribute in ISO 9126 is described in relation with a software product but in our particular case all quality factors and attributes are described in relation with approach characteristics. Quality factors could be classified into two categories:

  - *External/Internal Quality*: which measures the approach characteristics themselves.
  - *Quality in Use:* which measures the effect of using the approach characteristics in a specific context.

- *Quality Attribute:* According to (IEEE 610), a quality attribute is "A feature or characteristic that affects an item's quality (Syn: quality factor). In a hierarchy of quality attributes, higher-level attributes may be called quality factors, lower-level attributes called quality attributes". For example, Usability is defined for various quality attributes as Learnability, Understandability, Operability, etc.
- *Characteristic:* This is a higher-level concept of an approach. It may be, for example, the software development process, models, metamodels, languages, tools, transformations or the quality assurance techniques.
- *Subcharacteristic:* This is a lower-level concept of an approach. For example, the Model-Driven Engineering characteristic may have various subcharacteristics such as, the Language Definition, Transformations, Trace Generation, etc.
- *Metric:* In the Quality Model, metrics should indicate which quality attribute is affected by subcharacteristics and also the degree to which it is affected. For each subcharacteristic, a specification of its evaluation is necessary. For example, the evaluation may be via measuring quantitatively by metrics or subjective evaluation, inspections using checklists or interviewing the users. Links that validate that the right item is measured are also identified. In terms of metrics, our aim is to look for a series of qualitative and quantitative metrics based on their nature, although it might be interesting to have standard metrics on MDWE which are all, somehow, centralized. In the literature, numerous references to metrics can be found, but a standardization has yet to be carried out. Furthermore, the metrics used must be validated theoretically or empirically. A metric is used for measuring subcharacteristics.

  A metric may be classified according to their nature as a base metric, aggregated metric and/or derived metric.

  - *Base Metric:* This is obtained directly from analyzing a subcharacteristic.
  - *Aggregated Metric:* This is the composition consisting of a metric from a defined set of basic metrics, usually by means of a weighted sum.
  - *Derived Metric:* This is a mathematical function whose input is the value of other metrics.

Therefore, for our purposes, a Quality Model contains a minimal amount of characteristics and subcharacteristics through which any kind of MDWE approach can be evaluated. In order to define a Quality Model, it contains association links between the subcharacteristics and the quality attributes.

These association links represent the dependencies between *subcharacteristics* and *quality attributes*. They show *quality attributes* which are affected by *subcharacteristics* or the areas of the methodology that will be significantly affected if the approach is changed. Association links may be based on proven and real-world experience. The impact of each *subcharacteristic* on *quality attributes* must be demonstrated and the requirements determined by real case study applications to a number of real projects. This should be supplemented by reference to published literature. Furthermore, *subcharacteristics* have to define quantitative or qualitative *metrics* which may be used to measure each *subcharacteristic*. Otherwise it would be necessary to define a set of indicators from reference values which may be set to a prescribed state based on the results of measuring or on the occurrence of a specified condition. This option will be studied in future work. Hence, a *quality factor* has various *quality attributes* and a *characteristic* has various *subcharacteristics* as is shown in Figure 2. The relations between *quality attributes* and *subcharacteristics* are also presented. Moreover, a *subcharacteristic* may have various *metric*. *Metrics* are used for measuring *subcharacteristics*.

For example, one characteristic of the Quality Model component in QuEF is the Model-Driven Engineering characteristic which focuses on modelling language definitions, model transformations, trace generation, test-case generation, and rule-generation models. We have identified various subcharacteristics and defined some metrics which can be used for evaluating the degree to which the quality attributes are affected. The GQM templates are a structured way of specifying goals as is shown in Table 1. The GQM template contains the following fields:

2. http://java.sun.com

TABLE I. AN EXAMPLE OF A GQM TEMPLATE

Field	Examples
Object of study – Analyze	Model-Driven Engineering Characteristic
For the purpose of	Improving Usability,
Focus - With respect to their	Learnability, Understandability, Operability, Simplicity, Interpretability, Attractiveness,
Stakeholder - From the point of view of the	User of one approach
Context factors - In the context of	MDWE methodologies

This study can be for the characterization of the effect of using the Model-Driven Engineering Characteristic for improving Usability from the point of view of Users in the context of MDWE methodologies. However, existing methods could be used to define and theoretically validate all metrics in the framework. Although the ISO / IEC 9126 and IEEE specify the quality factors and quality attributes of a software product, they fail to indicate what quality measures determine a quality attribute. Therefore, the metrics for measuring subcharacteristics have to be identified. For example, and as shown in Figure 3, according to the Usability quality factor, including transformations can be a subcharacteristic that may have an impact on Attractiveness but may have no influence on the Understandability and Learnability quality attributes.

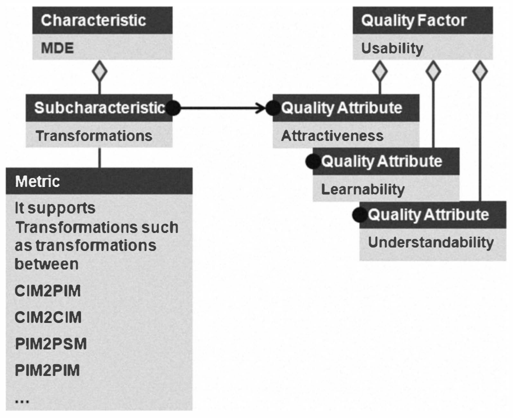

Figure 3. Brief example of association links in the Quality Model

This subcharacteristic may increase the degree to which an approach is more attractive for users in the sense of supporting a high number of transformations. This should be validated: for example by analyzing the number of transformations that support the approach. A set of qualitative metrics have to be defined to measure this subcharacteristic. Finally, indicators may describe one state of Attractiveness. In this brief example, it could be the number of transformations that are supported. In this context two types of points are defined:

- Tradeoff points are defined as the dependencies between subcharacteristics and quality factors.
- Sensitivity points are the areas of the methodology that will be significantly affected if the approach is changed.

The tradeoff points are the breeding ground for the sensitivity points, when a methodology changes, then the connections between different subcharacteristics also change.

Therefore, two new framework components can be defined as a consequence of the Quality Model concept in QuEF: Quality Model component and Approach Characteristics Template component. In this paper, the definition of the Quality Model component is described. Although the Approach Characteristics Template will be described in future work. In this paper we present a process for defining a Quality Model component for QuEF for MDWE methodologies which is inspired by the work of [9]. The steps are defined as shown in the Business Process Modeling Notation (BPMN) of Figure 4. Concepts, tasks to be performed for each step, and the Quality Model component structure which results for each step are described.

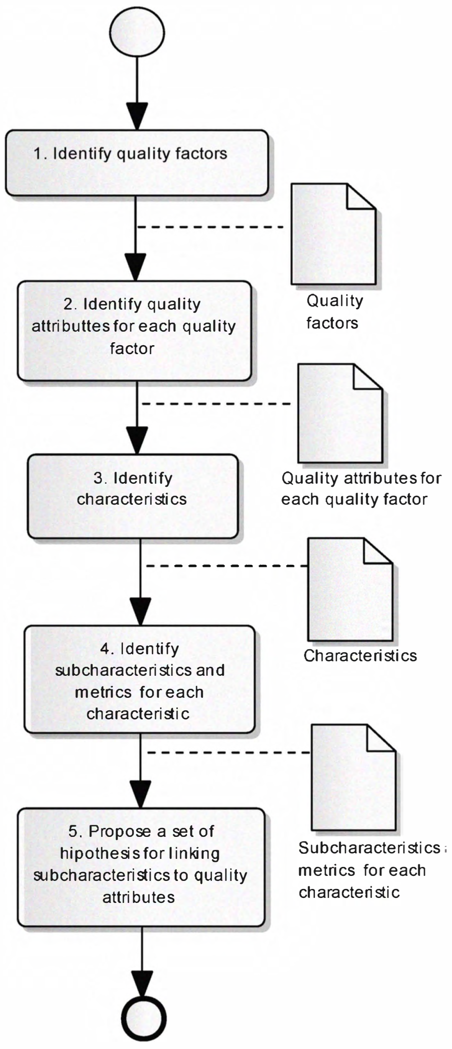
1. Identify quality factors
Quality factors
2. Identify quality attributtes for each quality factor
Quality attributes for each quality factor
3. Identify characteristics
Characteristics
4. Identify subcharacteristics and metrics for each characteristic
Subcharacteristics metrics for each characteristic
5. Propose a set of hipothesis for linking subcharacteristics to quality attributes

Figure 4. BPMN for the definition of the Quality Model Component

1) Identify quality factors

Identifying quality factors should involve all stakeholders and reflect the purposes and priorities of using a MDWE approach. To this ends, a set of quality factors based on current literature such as ISO/IEC 9126, IEEE and other standards which are adapted to MDWE methodologies has to be identified, classified and hierarchical. The Quality Factors of an approach may be:

- *External/Internal Quality:* Usability, Functionality, Reliability, Maintainability and Portability.
- *Quality in Use:* Effectiveness, Productivity and Satisfaction.

Each quality factor and attribute in ISO 9126 is described in relation with a software product but in this case all quality factors and attributes would be described in relation with approach characteristics.

In this work, Usability is taken as an example of the quality factor. In ISO 9126, Usability is a quality factor which is defined as: *"The capability of the software to be understood, learned, used and attractive to the user when used under specified conditions".* This definition could be adapted to more closely fit our specific domain: *"The capability of an approach characteristic to be understood, learned, used and attractive to the user when used under specified conditions"* or in a general way could be described as: *"A set of attributes that bear influence on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users".*

2) Identify quality attributes for each quality factor

For each quality factor, a set of quality attributes have to be identified. For example, quality attributes related with Usability are described in the same way by adapting other definitions from ISO, IEEE, other standards and work already published. These quality attributes may be described as:

- Learnability: *The capability of an approach characteristic to enable the user to learn how to use it.* [adapted from ISO 9126]
- Understandability / Comprehensibility: *The capability of being understood and the extent to which an approach characteristic is clear, without ambiguity, and easily comprehensible.* [adapted from ISO 9126]
- Simplicity: *The degree to which an approach characteristic has a design that is straightforward and easy to understand* [adapted from IEEE].
- Interpretability: *The extent to which an approach characteristic is in units of information appropriate for the capability of the developer.*
- Operability / Ease of Operation: *The capability of an approach characteristic to enable the user to operate and control it.* [adapted from ISO 9126].
- Attractiveness: *The extent to which an approach characteristic is attractive for developers to use.*

3) Identify characteristics

Approach characteristics can be the software development process, models, metamodels, languages, tools, transformations or the quality assurance techniques. In MDWE, models are refined progressively and transformed into new models or code. To this end, tools may also be used to test, verify or validate the models. Moreover, each methodology may define its development process and/or techniques. In this context, our belief coincides with that of other authors in that there is a direct relationship between the quality of the final software product and the quality of the methodologies used, such as the qualities of consistency and completeness. The quality of methodologies in turn depends on the following Characteristics:

- *The Model-Driven Engineering*: the modelling language definition used, such as their appropriateness for the MDWE domain and complexity, transformations, traces, test cases, and rule generation models as a prerequisite for successfully employing MDE in the style of the MDA of the OMG.
- The *knowledge of MDWE methodology users* of the problem in hand and their experience of web modelling languages and tool support in use.
- The *web modelling* which covers criteria for evaluating the web application development process, conceptual levels, features and levels of abstraction.
- The *customization modelling* which explicitly deals with characteristics related with the methodology adaptations in a web application development.
- The *maturity* of a methodology.
- The *tool support* used for modelling and transformations, such as compliance with the web modelling languages and capability of combining information.
- The *quality assurance techniques* applied to discover faults or weaknesses.

Methodology users and developers use the available modelling languages, tools and processes and develop models based on their knowledge of the problem and their experience. Another problem is that relations are often based on judgment. For example, ISO and IEEE have different hierarchies of quality factors and attributes. Therefore, a set of general MDWE approaches, characteristics and subcharacteristics have to be identified, classified and described based on work and current literature. The idea is to characterize the whole MDWE process.

4) Identify subcharacteristics and metrics for each characteristic

A characteristic or subcharacteristic is a feature assigned to a product, process or technique of a methodology, and hence is generally a set of user needs or expectations of a methodology. In this sense, evaluating the degree to which the quality attributes would be affected is not an easy task, and for this reason, most of the metrics defined so far are qualitative metrics which indicate if the subcharacteristic is *Supported (S)*, *Partly Supported (PS)* or *Not Supported (NS)*.

In this work, subcharacteristics and metrics for the MDE characteristics are described. Furthermore, a table for each subcharacteristic is shown with the metric and its possible values.

- **Language Definition:** This subcharacteristic is for the evaluation of whether a web modelling language has been defined explicitly in terms of a metamodel (including UML profiles), a grammar, a semantic description in terms of semantic web technologies, or if such a definition is absent.

TABLE II. LANGUAGE DEFINITION SUBCHARACTERISTICS AND METRICS

Metamodel, Schema, Grammar or Ontology	
It provides a metamodel based on the Meta Object Facility (MOF).	*S, PS or NS*
It provides a UML-profile (a metamodel extended from the standard UML metamodel)	*S, PS or NS*
Visual Syntax	
It provides a standard visual syntax for Model-Driven Web modelling similar to UML	*S, PS or NS*
Semantic	
It provides an standard semantic specification using the W3C (World Wide Web Consortium) recommendations such as OWL (the Web Ontology Language) and RDF (the Resource Description Framework)	*S, PS or NS*

- **Transformations:** This subcharacteristic for the evaluation of whether approaches might support or not support various types of model transformations. For example, an approach might support transformations between platform-independent models (PIM2PIM), and transformations between platform-independent and platform-specific models (PIM2PSM), transformations between platform-specific models and code (PSM2Code). To this end, different kinds of model transformation languages, such as imperative, declarative, or hybrid languages, can be used. In addition [19], following the classification of McNeile there are two interpretations of the MDE vision named "elaborationist" and "translationist" approaches [24]. Following the "elaborationist" approach, the specification of the application is built up step by step by alternating automatic generation and manual elaboration steps. Today, most approaches based on MDA are "elaborationist" approaches, which have to deal with the problem of model and/or code synchronization and reverse-engineering. In a "translationist" approach the platform independent design models of an application are automatically transformed to platform specific models, which are then automatically serialized to code. These models and the generated code must not be modified by the developer because roundtrip engineering is neither necessary nor allowed. In fact, the OMG has launched a new working group on Architecture Driven modernization (ADM), whose aims is the integration of Reverse Engineering and MDA.

TABLE III. TRANSFORMATION SUBCHARACTERISTICS AND METRICS

Transformations Types	
It uses a standard language for defining transformations (i.e. providing ATL and QVT transformations)	*S, PS or NS*
It provides mapping functions or transformations such as:	*S, PS or NS*

CIM2CIM	*S, PS or NS*	PIM2PIM	*S, PS or NS*
CIM2PIM	*S, PS or NS*	PIM2PSM	*S, PS or NS*
CIM2PSM	*S, PS or NS*	PIM2Code	*S, PS or NS*
CIM2Code	*S, PS or NS*	PSM2Code	*S, PS or NS*

Model-Driven Reverse Engineering or Synchronization (elaboratonist approach)	
It uses standard languages for defining synchronization methods or reverse engineering techniques such as ADM, XMI, MOF; GXL, JMI, EMF, MDR, QVT, etc.	*S, PS or NS*
It provides a synchronization method or a reverse engineering technique between transformations such as:	*S, PS or NS*

PIM2CIM	*S, PS or NS*	Code2CIM	*S, PS or NS*
PSM2PIM	*S, PS or NS*	Code2PIM	*S, PS or NS*
PSM2CIM	*S, PS or NS*	Code2PSM	*S, PS or NS*

- **Traces:** MDE processes must consider traceability practices for its successful application. They help the understanding, capturing, tracking and verification of software artefacts and their relationships and dependencies with other artefacts during the software life-cycle. This subcharacteristic evaluates if a generation of traces has been defined from transformations or between models. Regarding MDE, the traceability mechanism links elements of different models in order to specify elements useful in generating others. Those links can also be used to analyze impacts of model evolutions onto other models in the transformation chain.

TABLE IV. TABLE 1. TRACE SUBCHARACTERISTICS AND METRICS

Trace Generation Language	
It uses a standard language for defining trace generation from transformations (i.e. providing ATL and QVT transformations).	*S, PS or NS*
Horizontal Trace Generation	
It provides a local trace of relationships and dependencies of models with other models (CIM2CIM, PIM2PIM, etc.).	*S, PS or NS*
It provides a general trace of relationships and dependencies of models with other models (CIM2CIM, PIM2PIM, etc.).	*S, PS or NS*
Vertical Trace Generation	
It provides a local trace of relationships and dependencies of models with other models (CIM2PIM, PIM2PSM, etc.).	*S, PS or NS*
It provides a general trace of relationships and dependencies of models with other models (CIM2PIM, PIM2PSM, etc.).	*S, PS or NS*

- **Test Cases:** This subcharacteristic evaluates whether the approach offers effective processes for the systematic generation of test products in order to consume the shortest time possible and to cover a high number of tests. The test phase is one of the most important phases in the software development process. However, delays in development may be caused by incorrect execution. For this reason, several research teams are working on test cases generated directly from requirements.

TABLE V. TEST CASE SUBCHARACTERISTICS AND METRICS

Metamodel, Schema, Grammar or Ontology for Test Cases	
It provides a metamodel based on the Meta Object Facility (MOF) for Test Cases.	*S, PS or NS*
It provides a UML-profile (a metamodel extended from the standard UML metamodel) for Test Cases.	*S, PS or NS*
Visual Syntax for Test Cases	
It propose a standard visual syntax for Model-Driven Web modelling similar to UML for Test Cases	*S, PS or NS*
Semantic Description for Test Cases	
It provides an standard semantic especification using for example, the W3C (World Wide Web Consortium) recommendations, such as OWL(the Web Ontology Language) and RDF (the Resource Description Framework) for the definition of a Test Case Semantic Description	*S, PS or NS*
Transformations for Test Cases	
It uses a standard language for defining transformations (i.e. providing ATL and QVT transformations) for Test Cases.	*S, PS or NS*

- **Rule Generation Model:** This subcharacteristic is for the evaluation of whether the information about a model is represented separately within a rule generation model (a translationist approach) or if this information is captured within the transformation rules and later generating the model is elaborated by hand. (elaborationist approach)

TABLE VI. RULE GENERATION MODEL SUBCHARACTERISTICS AND METRICS

Rules Generation Model		
It uses a standard language for rule generation (i.e. providing ATL and QVT).		*S, PS or NS*
It supports a rule generation model such as		*S, PS or NS*
CIM2PIM	*S, PS or NS*	
PIM2PSM	*S, PS or NS*	
PSM2Code	*S, PS or NS*	

5) Propose a set of hypotheses for linking subcharacteristics to quality attributes.

In this step, the association links between subcharacteristics and quality attributes have to be defined. On one hand, a set of characteristics, subcharacteristics and metrics, quality factors and quality attributes have been defined. On the other hand, a set of hypotheses have to be proposed for indicating which quality attribute is affected for each subcharacteristic.

For example, usability is described as a set of quality attributes. These quality attributes could be affected by one of various subcharacteristics as shown in table VII.

TABLE VII. ASSOCIATION LINKS BETWEEN MDE SUBCHARACTERISTICS AND USABILITY QUALITY ATTRIBUTES

		Quality Attributes of Usability					
		L.	U.	S.	I.	O.	A.
Subcharacteristics of MDE	Language Definition	X	X	X	X	X	X
	Transformations					X	X
	Traces					X	X
	Test Cases					X	X
	Rule Generation Model	X	X	X	X	X	X

L.: Learnability	I.: Interpretability
U.: Understandability	O.: Operability
S.: Simplicity	A.: Attractiveness

The explanation for each subcharacteristic of the MDE characteristics and the relations between quality attributes is described below.

- **Language Definition:** Our initial hypothesis is that the language definition could bear influence on all quality attributes of Usability in general because:
  - **Learnability**: It is easier to learn an approach if it uses standard languages.
  - **Understandability**: It is easier to understand an approach for users if it uses standard languages.
  - **Simplicity**: An approach is simpler if a standard language used since its design is straightforward and easy to understand.
  - **Interpretability**: It is easier to interpret an approach for users if it uses standard languages.
  - **Operability**: An approach is more operable for its users if it uses standard languages.
  - **Attractiveness**: We think that this quality attribute is consequence of the others.

- **Transformations:** Our initial hypothesis is that the transformations could bear influence on only these quality attributes of Usability:
  - **Operability**: Defining transformations may make it easier to operate with an approach since it could reduce the number of operations or actions that the users have to carry on. In this sense, it is easier to control the work.
  - **Attractiveness:** It is more attractive for the users of an approach if it defines transformations since the transformations may reduce the amount of work.

- **Traces:** During the development phase of a complex system using an MDWE approach various types of errors can be encountered: those concerning the compiler and those concerning the system itself. Furthermore, as systems may evolve, the implied changes in different subparts can lead to a new stable configuration. Even if these issues are common to any system, they require specific management when a model-driven development approach is used. Our initial hypothesis is that it could bear influence on:

- **Operability**: Defining trace generation may make it easier to operate with an approach since changes and mistakes are controlled.
- **Attractiveness**: It is more attractive for the users of an approach if it defines trace generation since the trace generation may reduce the amount of work.

- **Test Cases:** Our initial hypothesis is that the transformations could bear influence on only these quality attributes of Usability:

  - **Operability**: Defining test generation models may make it easier to operate with an approach since it could reduce the number of operations or actions that the users have to carry out.
  - **Attractiveness**: it is more attractive for the users of an approach if it defines test case generation since the detection of mistakes in a shorter period of time may reduce the amount of work.

- **Rule Generation Model:** The employment of rule generation models together with transformation techniques could broaden the base of application platforms to be employed and if realized within the accompanying tool, could give the approaches a broader application base. Our initial hypothesis is that it could bear influence on:

  - **Learnability**: It is easier to learn an approach if it is translationist than if it is an elaborationist approach since each model or final code has to be elaborated after transforming from other models. Users have to learn how elaborate this model or code after transformation.
  - **Understandability**: It is easier to understand an approach if it is translationist than if it is an elaborationist approach since each model or final code has to be elaborated after transforming from other models. Users have to understand how to elaborate this model or code after transformation.
  - **Simplicity**: A translationist approach has a design that is straightforward and easier to understand.
  - **Interpretability**: It is easier to interpret a translationist approach than an elaborationist approach. Users have to interpret how to elaborate this model or code after transformation.
  - **Operability**: Defining Rule Generation Models may make it easier to operate with an approach since it could reduce the number of operations or actions that the users have to carry out. It is easier to control the work with this subcharacteristic.
  - **Attractiveness**: It is more attractive for the users of an approach if it defines a Rule Generation Model since a model for describing every new model may reduce the amount of work. Everything is more centralized.

The Quality Model component would be refined and improved based on results, experience or current literature. Other subcharacteristics have to be proposed and they have to be associated with quality attributes. In this work, a set of Model-Driven Engineering subcharacteristics and a set of hypotheses for linking these subcharacteristics to quality attributes of Usability are proposed as an example.

VI. EXAMPLE: NDT METHODOLOGY

A. General Description of NDT (Navigational Development Techniques)

NDT (Navigational Development Techniques) is a methodological approach oriented to Web Engineering. Web Engineering is a specific line in Software Engineering that offers specific models and techniques to deal with the special characteristics of Web systems. In recent years, numerous web approaches have been defined [12].

However, comparative studies concluded that these approaches are mainly focused on analysis and design phases and there is a major gap in the treatment of Web requirements. NDT is oriented to cover this gap. Thus, it is mainly focused on the requirements and the analysis phases. It is an approach defined in the Model Driven paradigm and it offers a suitable and easy methodological environment. The most important characteristics of this approach are:

- It offers an easy interface for the final user in the requirements phase.
- It is based on a set of MOF metamodels of which the development team need no previous knowledge. These metamodels are the base of the NDT development process.
- It follows the requirements traceability from the capture to the analysis, offering a systematic process based on formal transformations defined by QVT.
- NDT is completely based on UML, thus it can be compatible with other approaches.
- NDT is being applied in several real projects. It has been a widely applied methodology in real environments and is yielding very good results.

Nowadays, NDT has evolved in the enterprise environment and now covers the complete life cycle of a software project. With the use of NDT-Suite, NDT offers a tool support for each phase of the life cycle. In the next evaluation of NDT the extended revision supported by NDT-Suite is considered.

B. Applying the Quality Model in the NDT methodology for the Model-Driven Engineering Characteristic.

In the use of QuEF, this step would be similar to applying the Approach Characteristic Template component for the framework input because that component is based on the Quality Model. However, the Approach Characteristic Template component has not yet been fully developed, and a brief application of the approach is used as an example. On the other hand, this set of metrics could be quantified in order to give an idea of quantity for every quality attribute.

- **Language Definition**.

TABLE VIII. THE LANGUAGE DEFINITION SUBCHARACTERISTIC ON NDT

Metamodel, Schema, Grammar or Ontology	
It provides a metamodel based on the Meta Object Facility (MOF).	*S*
It provides a UML-profile (a metamodel extended from the standard UML metamodel).	*S*
Visual Syntax	
It provides a standard visual syntax for Model-Driven Web modelling similar to UML.	*S*
Semantics	
It provides a standard semantic specification using the W3C (World Wide Web Consortium) recommendations such as OWL(the Web Ontology Language) and RDF (the Resource Description Framework).	*NS*

With respect to the Language Definition of NDT, it only tails to support a Semantic specification language. In this way and according to the Quality Model component, NDT reaches good scores in some quality attributes, such as Learnability, Understandability, Operability, Simplicity, Interpretability and Attractiveness.

- **Transformations**

TABLE IX. THE TRANSFORMATION SUBCHARACTERISTICS ON NDT

Transformation Types				
It uses a standard language for defining transformations (i.e. providing ATL and QVT transformations).				*S*
It provides mapping functions or transformations such as:				*S*
CIM2CIM	*S*	PIM2PIM	*S*	
CIM2PIM	*S*	PIM2PSM	*S*	
CIM2PSM	*S*	PIM2Code	*S*	
CIM2Code	*S*	PSM2Code	*S*	
Model-Driven Reverse Engineering or Synchronization (elaborationist approach)				
It uses standard languages for defining synchronization methods or reverse engineering techniques such as ADM, XMI, MOF; GXL, JMI, EMF, MDR, QVT, etc.				*NS*
It provides a synchronization method or a reverse engineering technique between transformations such as:				*PS*
PIM2CIM	*S*	Code2CIM	*NS*	
PSM2PIM	*NS*	Code2PIM	*NS*	
PSM2CIM	*NS*	Code2PSM	*S*	

NDT reaches good scores in transformations. This means that is easier to operate with NDT, and attractive for users. In this way, NDT only provides synchronization between PIM2CIM and Code2PSM levels of abstraction.

- **Traces:**

TABLE X. THE TRACE SUBCHARACTERISTICS ON NDT

Trace Generation Language	
It uses a standard language for defining trace generation from transformations (i.e. providing ATL and QVT transformations).	*S*
Horizontal Trace Generation	
It provides a local trace of relationships and dependencies of models with other models (CIM2CIM, PIM2PIM, etc.).	*S*
It provides a general trace of relationships and dependencies of models with other models (CIM2CIM, PIM2PIM, etc.).	*S*
Vertical Trace Generation	
It provides a local trace of relationships and dependencies of models with other models (CIM2PIM, PIM2PSM, etc.).	*S*
It provides a general trace of relationships and dependencies of models with other models (CIM2PIM, PIM2PSM, etc.).	*S*

In the same way as for Transformations, NDT reaches good scores in trace generation. This means that is easier to operate with NDT, and is very attractive for users.

- **Test Cases:**

TABLE XI. THE TEST CASE SUBCHARACTERISTICS ON NDT

Metamodel, Schema, Grammar or Ontology for Test Cases	
It provides a metamodel based on the Meta Object Facility (MOF) for Test Cases.	*S*
It provides a UML-profile (a metamodel extended from the standard UML metamodel) for Test Cases.	*S*
Visual Syntax for Test Cases	
It proposes a standard visual syntax for Model-Driven Web modelling similar to UML for Test Cases.	*S*
Semantic Description for Test Cases	
It provides a standard semantic specification using, for example the W3C (World Wide Web Consortium) recommendations such as OWL(the Web Ontology Language) and RDF (the Resource Description Framework) for the definition of Test Case Semantic Description.	*NS*
Transformations for Test Cases	
It uses a standard language for defining transformations (i.e. providing ATL and QVT transformations) for Test Cases.	*S*

With respect to the Test Cases of NDT, it only fails to suuport a Semantic specification language. In this way and according to the Quality Model component, NDT yields good results for quality attributes such as Learnability, Understandability, Operability, Simplicity, Interpretability and Attractiveness.

- **Rule Generation Model**

TABLE XII. THE RULE GENERATION MODEL SUBCHARACTERISTICS ON NDT

Rule Generation Model		
It uses a standard language for rule generation (i.e. providing ATL and QVT).		*NS*
It supports a rule generation model such as		
CIM2PIM	*NS*	

	PIM2PSM	*NS*	*NS*
	PSM2Code	*NS*	

On the other hand, NDT has no Rule Generation Model. This is comprehensible since NDT is mainly focused on the requirements and the analysis phases. Furthermore, NDT methodology is an "elaborationist" approach. And hence remains unattractive for the users.

C. *Using the Quality Model for the evaluation of NDT methodology*

The Quality Model Component is needed for the development of the Quality Evaluation Process component. To illustrate this, for every quality attribute, an example of a quantitative value for each subcharacteristic is calculated. The total value for the quality attribute could be, for example, the number of values divided by the total metrics in the subcharacteristic. The metric value in the example is 1 if it is *supported,* 1/2 of the arithmetic mean of supported elements from among the total elements (for example in transformations) if it is *partly supported* and 0 if it is *not supported.*

TABLE XIII. VALUES ON APPLYING THE QUALITY MODEL TO NDT

	Subcharacteristics	**Quantified Value**	**Total Value**
Learnability	Language Definition	3 / 4	3/8
	Rule Generation Model	0	
Understandability	Language Definition	3 / 4	3/8
	Rule Generation Model	0	
Simplicity	Language Definition	3 / 4	3/8
	Rule Generation Model	0	
Interpretability	Language Definition	3 / 4	3/8
	Rule Generation Model	0	
Operability	Language Definition	3 / 4	5/8
	Transformations	(1+1+0+2/6) /4 = 3/5	
	Traces	1	
	Test Cases	4/5	
	Rule Generation Model	0	
Attractiveness	Language Definition	3 / 4	5/8
	Transformations	(1+1+0+2/6) /4 = 3/5	
	Traces	1	
	Test Cases	4/5	
	Rule Generation Model	0	

Finally, as is shown in Figure 5, as a graph to make Usability may be represented the evaluation of each quality factor or some aspect of quality easier. In the figure, the grey line represents Usability on the NDT methodology and the black line represents the ideal usability in an ideal approach according to the subcharacteristics under consideration. These types of graphs may be very useful in the evaluation. According to the results of the evaluation of the NDT methodology with the Quality Model, their graphs are very similar and regular but this is due to the fact that only one characteristic has been considered in the example. Hence, the same subcharacteristics have been considered for each quality attribute of Usability. This is the cause of results. If we would consider other characteristics and subcharacteristics the results could have been very different and the line which represents the NDT methodology would have been more irregular but more representative for the usability quality factor.

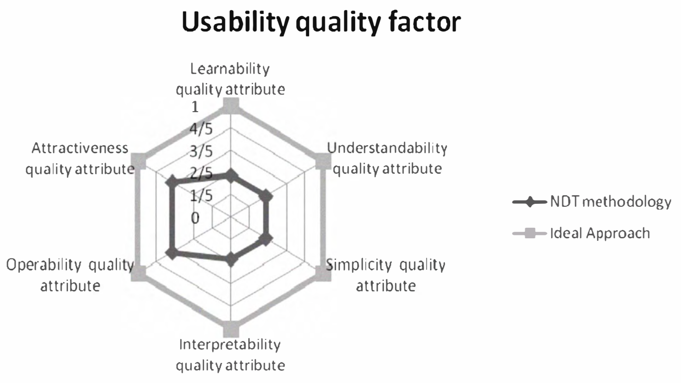

Figure 5. An example of NDT methodology evaluation

VII. CONCLUSIONS & FUTURE WORK

The Quality Model component in QuEF for MDWE methodologies is proposed in this paper and an example of its use in QuEF is described. With regards to the contributions obtained from this research, a Quality Model is proposed and a set of quality attributes are proposed for the Usability quality factor. Furthermore, subcharacteristics related with the MDE characteristic have been described which are required for the measurement of the value of MDWE methodologies in order to be able to assess and improve their Usability. In order to achieve a fully developed Quality Model, other quality factors and attributes, characteristics and subcharacteristics and associations links have to be studied in future work.

In this way, criteria can be unified when developing a new methodology or improving current proposals. We think that the use of QuEF would enhance the quality of products, processes and techniques of approaches. Therefore the use of QuEF may improve the efficiency and effectiveness of MDWE methodologies, and in turn may make their use more widespread. Since, "You can't control what you can't measure" (Tom DeMarco), we consider that QuEF is needed in MDWE to guide the way in which methodologies are able to assure the quality of the different MDWE development processes, techniques and the quality of the MDWE intermediate artifacts.

The principal benefit of QuEF is the ability to see if a proposed MDWE methodology will live up to user expectations. The approach evaluation helps one understand the strengths and weaknesses of a methodology. QuEF would not only help evaluate the current input approach, but would also help with the design of a new approach. This framework would help designers to ask the right questions and solve critical issues. Furthermore, it would be necessary to carry out a standardization of terminology to improve the access channel for communication in MDWE.

ACKNOWLEDGMENT

This research has been supported by the project QSimTest (TIN2007-67843-C06_03) and by the RePRIS project of the Ministerio de Educación y Ciencia (TIN2007-30391-E), Spain.

REFERENCES

[1] S. Abrahão, L. Olsina, and O. Pastor, "A Methodology for Evaluating Quality and Functional Size of Operative WebApps". Proc. of 2nd Int. Workshop on Web Oriented Software Technology (ECOOP'02) Workshops, pp. 1-20, Spain, 2002.

[2] S. Abrahão, N. Condori-Fernández, L. Olsina and O. Pastor, "Defining and Validating Metrics for Navigational Models". IEEE Computer Society. Proceedings of the Ninth International Software Metrics Symposium (METRICS'03), pp. 200, Australia, 2003.

[3] L. C. Briand, W. L. Melo. and J. Wüst. "Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects". ISERN Report No. ISERN-00-06, V. 2, 2006.

[4] C. Cachero, G. Poels, C. Calero, "Towards a Quality-Aware Web Engineering Process". Twelfth International Workshop on Exploring Modelling Methods in Systems Analysis and Design. Vol. 1, pp 7-16. Held in conjunction with CAISE'07Trondheim, Norway, 2007.

[5] C. Cachero, C. Calero, Y. Marhuenda, "A Quality-Aware Engineering Process for Web Applications". Handbook of research on Web Information Systems Quality. 2008.

[6] C. Calero, J. Ruiz, M. Piattini, "Classifying web metrics using the web quality model". Vol. 29, No. 3, pp. 227-248, 2005.

[7] S.R. Chidamber, C.F. Kemerer, "A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering", Vol. 20, No. 6, pp. 476-493, 1994.

[8] Y. Deshpande, S. Marugesan, A. Ginige, S. Hanse, D. Schawabe, M. Gaedke, B. White, "Web Engineering", J. Web Eng., Vol. 1, No.1, pp. 3-17, 2002.

[9] R. G. Dromey, "Concerning the Chimera". IEEE Software 13(1), pp. 33–43, 1996.

[10] A. Etien and C. Rolland, "A Process for Generating Fitness Measures". (CAiSE 2005), LNCS 3520, pp. 227-292. 2005.

[11] M.J. Escalona, G. Aragón, "NDT. A Model-Driven Approach for Web Requirements". IEEE Transactions on software engineering, Vol. 34, No. 3, pp. 377-390, 2008.

[12] M.J. Escalona, J. Torres, M. Mejías, J.J. Gutiérrez, D. Villadiego. "The treatment of navigation in Web Engineering". Advances in Engineering Software Elsevier Ld.; England. Vol. 38, pp. 267-282, 2007.

[13] M.J. Escalona, N. Koch, "Requirements Engineering for Web Applications – A comparative study". Journal of Web Engineering. Vol. 2, No. 3, pp. 193-212, 2004.

[14] F. García, M. Serrano, J. Cruz-Lemus, F. Ruiz, M. Piattini, "Managing software process measurement: A metamodel-based approach", Information Sciences.Vol. 177, No. 12, pp. 2570-2586, 2007.

[15] J.J. Gutierrez, M.J. Escalona, M. Mejías, I. Ramos, J. Torres, "An approach for Model Driven test generation", Proceeding of the IEEE International Conference on Research Challenges in Information Science. IEEE Morocco (RCIS 2009), pp. 337-346, Morocco. 2009.

[16] IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.

[17] ISO- International Organization for Standardization, ISO/IEC 9126-1, http://www.iso.org.

[18] J. Kirakowski, R. Whitehead, N. Claridge, "Human Centered Measures of Success in Web Site Design", Proc. of 4th Conference on Human Factors & the Web, Basking Ridge, New Jersey, USA, 1998.

[19] C. Kroiβ, N. Koch, "UWE Metamodel and Profile, User Guide and Reference". Technical Report 0802. Programming and Software Engineering Unit (PST), Institute for Informatics. Ludwig-Maximilians-Universität München, Germany, 2008.

[20] A. Lake, C. Cook, "Use of factor analysis to develop OOP software complexity metrics". Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon, 1994.

[21] Y.-S. Lee, B.-S. Liang, S.-F. Wu, F.-J. Wang, "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow", in Proc. International Conference on Software Quality, Maribor, Slovenia, 1995.

[22] J. A. McCall, P. K. Richards, G. F. Walters, "Factors in Software Quality", Nat'l Tech. Information Service, Vol. 1, 2 and 3. 1977.

[23] J. Mcgarry, D. Card, C. Jones, B. Layman, E. Clark, J. Dean, F. Hall, "Practical Software Measurement. Objective Information for Decision Makers", Addison-Wesley, 2002.

[24] A. McNeile. MDA: The vision with the Hole . http://www.metamaxim.com/download/documents/MDAv1.pdf, 2003.

[25] P. Mohagheghi, V. Dehlen, "Developing a Quality Framework for Model-Driven Engineering". Models in Software Engineering: Workshops and Symposia at MoDELS 2007, pp. 275–286, 2008.

[26] P. Mohagheghi, J. Aagedal, "Evaluating Quality in Model-Driven Engineering". International Workshop on Modeling in Software Engineering. (MISE'07), IEEE Computer Society, 2007.

[27] P. Mohagheghi, V. Dehlen, "A Metamodel for Specifying Quality Models in Model-Driven Engineering." 2007

[28] J. Nielsen, "Designing Web Usability: The Practice of Simplicity", New Riders Publishing, 2000.

[29] OMG: MDA. http://www.omg.org/mda/faq_mda.htm

[30] OMG: MDA Guide, http://www.omg.org/docs/omg/03-06-01.pdf. 2005.

[31] J. M. Pérez, F. Ruiz, M. Piattini, "Model Driven Engineering Aplicado a Business Process Management", Informe Técnico. UCLM-TSI-002, 2007.

[32] J. Ralyté, X. Lamielle, N. Arni-Bloch, M. Lèonard, "A Framework for Supporting Management in Distributed Information Systems Development". 2nd In. Conference on Research Challenges in Information Science (RCIS 2008), pp. 381-392, Morocco, 2008.

[33] A. Schauerhuber, M. Wimmer, E. Kapsammer, "Bridging existing Web Modeling Languages to Model-Driven Engineering: A Metamodel for WebML". II Int. Workshop on Model-Driven Web Engineering, International Conference On Web Engineering, Vol. 155, No. 5, USA, 2006.

[34] W. Schwinger,W. Retschitzegger, A. Schauerhuber, G. Kappel, M. Wimmer, B. Pröll, C. Cachero Castro, S. Casteleyn, O. De Troyer, P. Fraternali, I. Garrigos, F. Garzotto, A. Ginige, G-J. Houben, N. Koch, N. Moreno, O. Pastor, P. Paolini, V. Pelechano Ferragud, G. Rossi, D. Schwabe, M. Tisi, A. Vallecillo, van der Sluijs and G. Zhang, "A survey on web modeling approaches for ubiquitous web applications". International Journal of web Information Systems Vol. 4 No. 3, pp. 234-305, 2008.

[35] A. Vallecillo, N. Koch, C. Cachero, S. Comai, P. Fraternali, I. Garrigós, J. Gómez, G. Kappel, A. Knapp, M. Matera, S. Meliá, N. Moreno, B. Pröll, T. Reiter, W. Retschitzegger, J. E. Rivera, W. Schwinger, M. Wimmer, and G. Zhang, "MDWEnet: A Practical Approach to Achieving Interoperability of Model-Driven Web Engineering Methods", Proc. Third Int'l Workshop Model-Driven Web Eng., pp. 246-254, 2007.

[36] World Wide Web Consortium (W3C): www.w3.org