Applying ISO 9126 Metrics to MDD Projects

Ricardo Muñoz-Riesle, Beatriz Marín
Facultad de Ingeniería
Universidad Diego Portales
Santiago, Chile
e-mail: rm.riesle@gmail.com; beatriz.marin@mail.udp.cl

Lidia López
Software and Service Engineering Group
Universitat Politècnica de Catalunya
Barcelona, Spain
e-mail: llopez@essi.upc.edu

***Abstract*— The Model Driven Development (MDD) paradigm uses conceptual models to automatically generate software products by means of model transformations. This paradigm is strongly positioned in industry due to the quickly time to market of software products. Nevertheless, quality evaluation of software products is needed in order to obtain suitable products. Currently, there are several quality models to be applied in software products but they are not specific for conceptual models used in MDD projects. For this reason, it is important to propose a set of metrics to ensure the quality of models used in MDD approaches in order to avoid error propagation and the high cost of correction of final software applications. This paper analyzes the characteristics and sub-characteristics defined in the ISO/IEC 9126 quality model in order to reveal their applicability to MDD conceptual models.**

Keywords-Quality Model; Model-Driven Development; Metrics; ISO 9126; Conceptual models

I. INTRODUCTION

Software production processes based on Model Driven Development (MDD) generate software products automatically or semi-automatically from conceptual models by means of model transformations [1][2]. To do this, well-defined modeling constructs, model-to-model transformations and model-to-code transformations are needed. Therefore, MDD approaches uses as input conceptual models and models transformations in order to generate the programming code of software products. This type of development is strongly positioned in the software development industry [3] due to the automatic generation of code, which speed the time to market and avoids error propagation and the high cost of correction of human errors in manual programming.

The MDD software process is supported by the Model Driven Architecture (MDA) [4] standard. MDA defines four abstraction levels for the models used to generate a software product that goes from the higher abstraction level to the lower abstraction level. These levels correspond to the computation independent model (CIM), the platform independent model (PIM), the platform specific model (PSM), and the implementation model (IM). Therefore, the conceptual models used by MDD approaches at any level become an essential resource in the process of software generation due to they are the main input for code generation. In other words, CIM models are used to generate PIM models, PIM models are used to generate PSM models, and PSM models are used to generate the IM model, which corresponds to the code in a specific programming language.

The quality evaluation of these conceptual models is of paramount importance since it allows an early verification of final software products. However, there is no standard defined to evaluate the quality of conceptual models used at MDD environments. The ISO 9126 standard [5] presents a set of characteristics and sub-characteristics that allows the evaluation of the quality of a software product by using different quality metrics proposed for each characteristic. However, these metrics are applied to measure artifacts obtained in later stages of software development cycles, increasing the cost of detecting and correcting defects.

We advocate that it is possible to apply the standard ISO 9126 to evaluate software products that have been developed under an MDD approach. Thus, this paper analyzes the ISO 9126 characteristics, sub-characteristics and their metrics in order to fit an MDD development process, and therefore, evaluate the quality of MDD projects using the metrics defined by ISO 9126. To do this, an exploratory study about the applicability of the defined metrics to conceptual models at different abstraction levels of MDD approaches has been performed. Thus, the main contribution of this work is the selected set of metrics that can be applied to the conceptual models that are specified at the different abstraction levels regarding MDA. This set of metrics composes, therefore, a quality model that allows an early evaluation of software generated in MDD environments.

This contribution is useful for both practitioners and researchers. Practitioners can use this set of metrics in order to evaluate early the quality of models used for the generation of their software products, aligning this evaluation with the standard ISO 9126. Researchers can use this set of metrics in order to integrate it to other quality evaluation techniques used at early phases in the software development cycle.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 presents an exploratory study about the ISO 9126 quality model in order to evaluate the applicability of the defined metrics at MDD projects. Section 4 presents the application of the set of selected metrics of the quality model to a case study. Finally, Section 5 presents an overall analysis and some conclusions from the results obtained.

II. BACKGROUND AND RELATED WORK

This section introduced the ISO 9126 standard in order to facilitate the understanding of later sections. Afterwards, a discussion about relevant related works is presented.

 ISBN: 978-1-61208-438-1

A. ISO/IEC 9126 standard

The ISO/IEC 9126 standard consists of four parts: the quality model [5], the external metrics [6], the internal metrics [7] and the metrics for quality in use [8]. The first part describes in detail six quality characteristics for software products (functionality, reliability, usability, efficiency, maintainability, and portability), and their corresponding sub-characteristics (see Figure 1).

Functionality	Reliability	Usability	Efficiency	Maintainability	Portability
Suitability	Maturity	Understandability	Time Behaviour	Analizability	Adaptability
Accuracy	Fault Tolerance	Learnability	Resource	Changeability	Installability
Interoperability	Recoverability	Operability	Utilization	Stability	Co-Existence
Security	Compliance	Attractiveness	Compliance	Testability	Replaceability
Compliance		Compliance		Compliance	Compliance

Figure 1. ISO 9126 characteristics and sub-characteristics.

In order to determine the level of quality of software products, it is necessary to evaluate these characteristics by applying a set of well-defined metrics. Thus, the second, third and fourth part of ISO 9126 describes metrics to assess the software quality. These metrics are focused on measuring artifacts obtained in later phases of the software development cycle, complicating the detection and correction of problems at early stages, which are propagated to later stages.

B. Related Work

The quality evaluation of software products is the paramount importance. For software products that are developed using an MDD approach, the quality evaluation can be performed at the conceptual models that are used as input for the automatic code generation.

There are several works that are focused in the quality evaluation of software products generated in an MDD approach. A mapping study of works that are focused in quality at Model-Driven Engineering (MDE) was presented in [9]. The main concerns presented in this study are (1) the studies do not provide an explicit definition of quality in model-driven contexts; (2) studies that are focused in UML models are not aligned with MDD approaches; and (3) there is a lack of analysis that indicates when a metric may or may not be applied to an MDD approach.

An approach to integrate usability evaluations of ISO 9126 into Model-Driven Web Development was presented at [10]. This study shows how the final user interface can provide feedback about changes in order to improve usability issues at intermediate artifacts of MDD projects (PIM and PSM models). This paper presents a similar way to us to analyze if the ISO 9126 metrics can fit into the MDD approach.

In [11], a quality model for MDD projects was presented. This model allows the verification of conceptual models by using a set of rules to detect defects related to data, process and interaction perspectives.

As summary, even though there are several model-driven proposals defined, and also there are several works that are1 focused on quality evaluation of MDD projects, there is a lack of a standard method to evaluate the quality at the different abstraction levels in model-driven approaches. For this reason, we decided to analyze the overall software quality framework presented by the ISO 9126 in order to identify if the metrics presented can be applied to the conceptual models defined at the different abstraction levels of MDD approaches.

III. EXPLORATORY ANALYSIS OF ISO 9126

This section presents an analysis of each characteristic of ISO 9126 and their corresponding sub-characteristics, considering if they are or not suitable to be applied to software products developed by using an MDD approach. In addition, the abstraction level of each presented metric has been identified.

A. Functionality

Functionality has been defined as the capability of a software product to provide the functions that meet the stated and implied needs when the software is used under specific conditions [5]. Requirements specifications are used to define the functions that meet the needs of users. Thus, to evaluate functionality it is necessary to focus in this kind of specifications.

In MDD projects, the requirements specifications can be performed by using CIM models, for instance i* models [12] [13], use-case diagrams [14], or BPMN diagrams [15]. Later, these models are transformed into PIM models (such as class diagrams) in order to continue with the process of code generation by using an MDD approach. These transformations are performed by different MDD approaches with their supporting tools, such as [16][17]. Thus, it is possible to evaluate the functionality of a software product generated in an MDD environment by using the CIM models that correspond to the requirements models. In other words, it is possible to evaluate if the software provide the functions stated in requirements by using CIM models and the traceability [18] of these models to the final programming code.

Functionality is comprised of the following sub-characteristics: suitability, accuracy, interoperability, security and conformance. For all the metrics, the closer to 1, the better:

1) Suitability: It corresponds to the capability to provide an appropriate set of functions for specific objectives [5]. In this context, the appropriateness is understood as the ability to correctly select a set of functions that meet the user needs. This verification process can be performed comparing the CIM with the PIM, or the CIM with the IM generated. The following metrics has been defined to evaluate the suitability [6]:

- Functional Adequacy (FA): This metric evaluates how adequate are the functions by using the formula presented in (1). This metric is useful for MDD approaches since it can be used at CIM or PIM. To do this, it is necessary to apply inspection techniques to find problems in the functions specified at CIM or PIM. For instance, if an MDD approach uses a class diagram as PIM, it is possible to identify defects in the defined functions by using for example a list of possible defects [19][20].

FA = 1-(Number of functions in which problems are detected/Number of functions evaluated) (1)

- Functional Implementation Completeness (FIC): This metric evaluates how complete is the implementation according to requirement specifications using the

formula presented in (2). This metric allows the identification of missing functions, based on the requirement specifications. Note that in MDD approaches the requirements are specified at CIM. Thus, the verification process can be performed by evaluating if the functions specified at CIM are present at PIM after the transformation from CIM to PIM, and also, it can be performed by evaluating if the functions specified at PIM are present at IM after the transformation process. For example, if an MDD approach uses i* models as CIM, it is possible to apply rules to verify that all the elements specified in the i* models are in the class diagram by using [21].

FIC = 1-(Number of missing functions detected/Number of functions described in Req Spec) (2)

- Functional Implementation Coverage (FICo): This metric evaluates how correct is the functional implementation using the formula presented in (3). This metric identifies those functions that have been incorrectly implemented or have not been implemented instead they have been specified in requirements models. In MDD projects, the code (IM) is automatically generated by the compilers, so that, to use this metric it is necessary to evaluate the CIM and the IM. Note that to go from CIM to IM it is necessary to go from CIM to PIM, then from PIM to PSM, and later for PSM to IM.

FICo = 1-(Number of incorrectly implemented or missing functions/Number of functions described in Req Spec) (3)

2) Accuracy: Corresponds to the capability of the software product to provide the right or agreed results or effects [5]. In order to measure the accuracy, it is necessary to define the concepts of trueness and precision. Trueness refers to the closeness of the mean of the measurement results to the true value; and precision refers to the closeness of agreement within individual measurement results. Therefore, according to the ISO standard, the term accuracy refers to both trueness and precision [22].

In order to measure the accuracy, two metrics has been defined [6]: Accuracy to expectation (AE) and Computational accuracy (CA). AE evaluates the actual results against the reasonable expected results in the operation time. CA evaluates how often the user found inaccurate results during the operation time. In both cases, to evaluate the accuracy is necessary to have the user executing the code. Thus, these metrics are used in later phases of the software development, so that, they do not contribute to the early quality evaluation of MDD projects.

3) Interoperability: Corresponds to the capability of a software product to interact with one or more specified systems [5]. The interoperability of a software product can be specified at the conceptual models that are used as input in an MDD approach. To do this, interfaces with other systems must be defined in the conceptual model to specify the data inputs and outputs.

The following metric is defined to evaluate the interoperability in [6]:

- Data Exchangeability (DE): This metric evaluates how correctly have been specified the exchange functions for specific data transfer using the formula presented in (4). To evaluate this metric it is necessary to specify the data formats that are exchanged with other systems and then apply inspection techniques to verify that the functions defined to exchange data are correctly defined. This metric can be evaluated at PIM of MDD approaches, where it is possible to specify the interaction with other systems. For instance, the MDD OO-method approach [23] allows the specification of *Legacy Views*, which corresponds to the abstraction of a software component that is represented by a class. The specification of the attributes and services of a legacy view requires the characterization of the functions or procedures that effectively carry out the corresponding function at other systems. By doing this, it is possible to identify whether software functions are compatible with the software that is specified by using the MDD approach. In addition, if the other system it is also specified by using an MDD approach, then, it is possible to specify the interactions between both systems at the metamodel level [24]. Thus, to verify this metric it is necessary to inspect the models following the syntax, semantics and restrictions specified in the metamodels.

DE = (Number of data formats which are approved to be exchanged successfully with other software or system during testing on data exchanges/Total number of data formats to be exchanged) (4)

4) Security: Corresponds to the capability of the software product to protect information in order to avoid unauthorized people or systems to read or modify them; and to provide authorized people or systems to have access to them [5]. The MDD approaches allow the specification of the users of the generated software. For instance, the OO-Method MDD approach allows the specification of agents at PIM, which have access to perform specific tasks and to read specific data.

The following metrics have been defined to evaluate the security [6]: Access Auditability (AA), Access Controllability (AC), and Data Corruption Prevention (DCP). AA, AC and DCP are calculated by using information of the user access at the operation time. Therefore, they do not contribute to the early evaluation of the quality of MDD projects. Nevertheless, it is important to note that in MDD projects the code is automatically generated from an input conceptual model. Thus, if an erroneous access is found in the access to the functionality, it can be corrected at PIM, and then regenerate the code.

B. Reliability

Reliability corresponds to the capability of the software product to maintain a specified level of performance when it is used under specified conditions [5]. To evaluate the performance it is necessary to execute the software, so that it cannot be simulated at design time of software, and it is

 ISBN: 978-1-61208-438-1

necessary to use the IM model, which corresponds to the code automatically generated by the MDD approach.

C. Usability

Usability corresponds to the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use [5]. This characteristic and its sub-characteristics are usually used once the software is executed, but we advocate that it is possible to measure this characteristic at early stages of the software development cycle by using an MDD approach. To do this, it is necessary that the conceptual model of the MDD approach has a holistic representation of the system, i.e., including the specification of the structure of the system, the behavior of the system, and the graphical user interface.

In [25], a new sub-characteristic of usability is presented: Complexity, which can be applied to MDD approaches. Two metrics have been defined to evaluate the complexity in the use of interfaces and operations in software.

- Complexity: This metric provides an indicator that measures the average number of operations per offered interface [25] using the formula presented in (5). For MDD projects, this metric can be evaluated by using the specification of the graphical user interfaces defined in the presentation model and the services that are accessed from these interfaces. Thus, this metric can be applied at the PIM of MDD approaches. This parameter can be compared with the user's opinion on how hard it is to use all the operations of a specific interface. This would indicate the perceived level of complexity if the system has high complexity or low complexity by using the IM model in order to define the acceptable value of this metric.
- Interface Defects Avoidance (IEA): This metric defines the level of understanding of a user after a defect occurs. The closer to 1, the better. IEA uses the average number of graphic operations failed recognized by the user in comparison to the total defects pre-defined by the developers. Thus, this metric is evaluated when the software is executed, so that it does not contribute to the early quality evaluation of MDD projects.

Complexity = (Operations in all offered interfaces/Offered interfaces) (5)

D. Efficiency

Efficiency corresponds to the capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions [5].

Unfortunately, there are many external factors, such as bandwidth, hardware, and number of users connected, which cannot be known at early stages of the software development cycle since they cannot be specified in the conceptual model. These factors are only known when the software is executed, so that this characteristic cannot contribute to the early evaluation of quality of MDD projects.

E. Maintainability

Maintainability corresponds to the capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software, and also, in requirements and functional specifications [5].

The maintainability can be evaluated in the conceptual models used by MDD approaches. An MDD approach allows the automatic generation of code by using as input a conceptual model, thus facilitating the detection of defects in the final product, and the corresponding corrections at the conceptual model. In addition, the automatic code generation allows software analysts to easily return to the initial steps of the software development cycle in order to include improvements or adaptations in the conceptual model. For this reason, the sub-characteristics of maintainability are also analyzed.

1) Analyzability: Corresponds to the capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified [5]. The metrics defined in [6] are focus to measure analyzability by observing the user's behavior, so that they do not make a contribution as a quality metric for MDD approaches.

2) Changeability: Corresponds to the capability of the software product to enable a specified modification to be implemented [5]. One of the main advantages of MDD approaches is the ease of change. This is due to the great advantage of the automatic generation of code that allows the quick return to any stage of the development cycle, and therefore, correct the problem by redefining the models of the software.

The metrics defined in [6] are not useful to define a quality model for MDD approaches regarding the changeability, because these metrics are focused on the user behavior using the software at a specific time, instead of measuring the behavior of the software itself. Despite this, we found a metric in [25], which has been defined to evaluate the changeability:

- Customizability Ratio (CR): This metric provides an indicator of the ability of modification of the software using the formula presented in (6). If the software offers few interfaces and many parameters, normally it would be very modifiable, though difficult to handle, while one with many interfaces and few parameters is slightly modified. This metric can be evaluated by using the PIM of an MDD approach.

CR = (Number of parameters/Number of interfaces offered) (6)

3) Stability: Corresponds to the capability of the software product to minimize unexpected effects from modifications of the software [5]. In [6], there are defined metrics focused on user's behavior so that they do not perform a contribution for the early quality evaluation of MDD.

4) Testability: Corresponds to the capability of the software product to enable modified software to be validated [5]. Software developed by MDD approaches can be easily tested by the automatic generation of code and test cases [26]. This allows testing the software model based on the software

 ISBN: 978-1-61208-438-1

requirements specification. If a problem occurs, it can be solved by returning to the initial stages of software development cycle.

For this reason, the metrics defined in [6], do not contribute to the early quality evaluation of MDD projects because they are focused on user's behavior.

F. Portability

Portability corresponds to the capability of a software product to be transferred from one environment to another [5]. In MDD approaches, the software products are developed under specific requirements, using models such as PSM [3]. Therefore, the same conceptual model can be used on different platforms assuring portability. Unfortunately, metrics defined in [6] doesn't help to evaluate the quality for MDD approach. This is because the metrics defined by the ISO 9126 are focused on reuse of the software developed. In contrast, MDD approaches allow going one step back and re-compile the conceptual model to different technological platforms by using different PSM and compilers.

G. Other Metrics of ISO 9126

The metrics presented with their formula are focused on measuring quality for MDD approach at an early stage of software development. Nevertheless, there are other metrics defined in the ISO 9126 standard that cannot be used to measure the quality of models used at MDD approaches, since they are used in final stages of software development, i.e., they need the execution of the software to be tested or they are focused on the user's behavior.

These metrics are (1) Functional Specification Stability (FSS), which counts the number of functions changed after entering in operation; (2). Precision (P), which counts the number of results with a level of precision different from required during the operation time; (3) Data exchangeability by the user (DEu), which counts the number of cases in which user failed to exchange data with other software or systems.

IV. CASE STUDY

This Section exemplifies how the metrics are used at a software development project using an MDD approach. To do this, we present a system called SICOVE, which corresponds to a vehicle trading system that supports the process of managing vehicles, premises, revenues and costs undertaken by a buy-sell generic vehicle company (accounting and taxes processes associated are excluded). Figure 2 shows the conceptual model for SICOVE system.

This conceptual model has been specified using OO-Method approach and the Integranova [27] tool, which is able to compile the conceptual model and automatically returns the generated code compiled to different platforms. To do this, the OO-Method conceptual model is comprised of four complementary views: the static view, the functional view, the dynamic view and the presentation view. The static view is specified in a class diagram, which allows the specification of the structure of the final system. The functional view is specified in a functional model, which allows the specification of the change of values of the attributes when a service is executed. The dynamic view is specified in a state transition diagram, which allows the specification of the valid lives of an object. The presentation view is specified in a presentation diagram, which allows the specification of the graphical user interface. We have selected this tool to apply the set of metrics since it is an MDD tool that has more than 10 years of successful usage in industry.

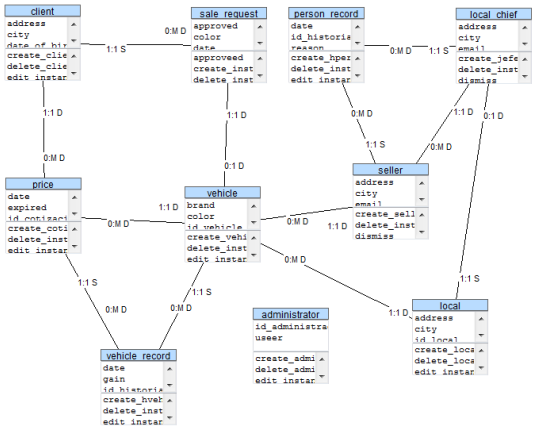

Figure 2. SICOVE Conceptual Model

Figure 2 shows the structural view of SICOVE system. All the functions of SICOVE have been specified by using the functional model (e.g., see Figure 3, which presents the specification of *create_client*). In Figure 3 it is possible to see the inbound arguments and the data type of each argument of the service.

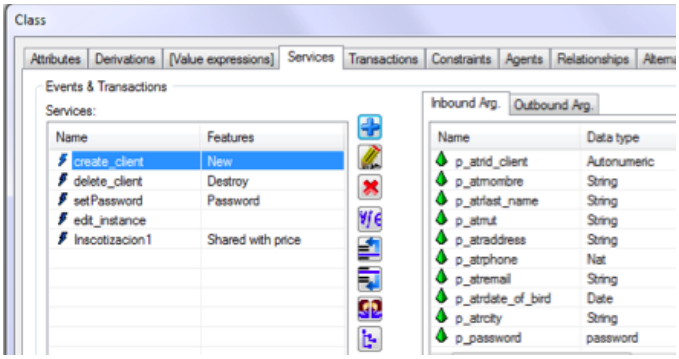

Figure 3. Example of SICOVE functional view

The generated code allows the testing of some of the functions of SICOVE system, for example create_client. In order to create a client we need to enter the following data into the system: id_client, name, last_name, rut, address, phone, email, date of birth, city (e.g., see Figure 4, which presents the attributes for the class client). Once entered the data, the system verifies that the user is not registered in the database in order to add it.

The SICOVE system has been used to evaluate the applicability of the metrics proposed in Section III. To do this, an analysis of all the functions defined in the specification of the system was performed in order to evaluate each metric proposed for MDD.

Table I shows the results obtained by applying the proposed metrics to the SICOVE system. This data was calculated by using the mathematical formulas described before, the requirements specification of the vehicle trading system that

 ISBN: 978-1-61208-438-1

was performed by using the IEEE 830 standard, and the conceptual model defined by the OO-Method approach, which correspond to the PIM abstraction level of MDA.

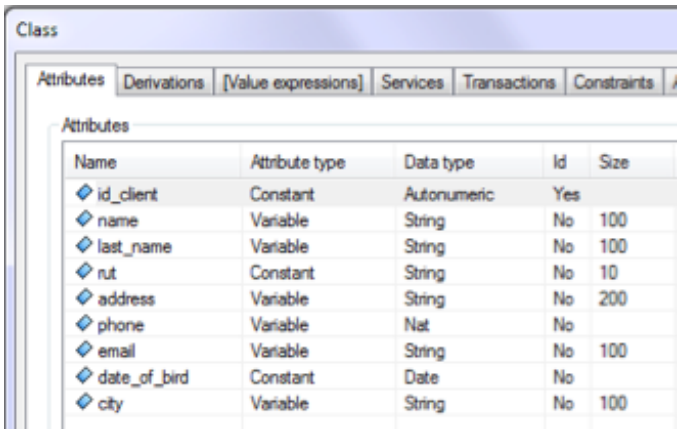

Figure 4. Example of SICOVE attributes

Regarding the functionality, FA, FIC and FICO metrics obtain a value less than one. This means that there are some functions that have been specified in the requirements but they do not have been generated at PIM. A summary of the functions defined in the requirements specification are presented in Table II. As this table shows, there is some functionality that is not fully present at the PIM conceptual model, such as Generate Quotation, Set Vehicle for Sale and Sell Vehicle. Thus, from a total of 17 functions defined for the SICOVE system, 3 of them are not fully implemented (e.g., see Figure 5). The result obtained after applying the mathematical formulas is 0.8 for each metric, which indicates that are some functions of the SICOVE system are not implemented. If the value of these metrics had been 1 this would indicate that all functions were correctly implemented.

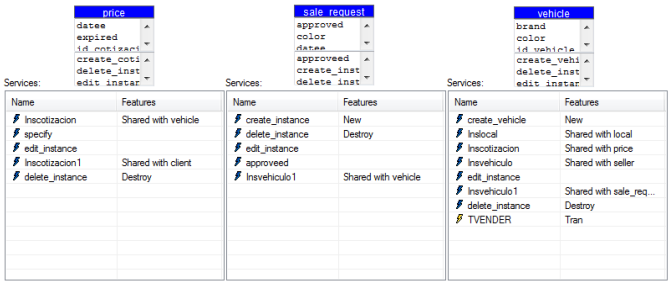

Figure 5. Functions specified for SICOVE system

TABLE I. RESULTS

Characteristic	Sub- Characteristic	Metric	Result
Functionality	*Suitability*	*FA*	*0.8*
		FIC	*0.8*
		FICo	*0.8*
	Interoperability	*DE*	-
Usability	*Complexity*	*Complex*	*5.3*
Maintainability	*Changeability*	*CR*	*7.1*

TABLE II. FUNCTIONS FOR SICOVE SYSTEM

ID	Functions of SICOVE system	Defined Functions	ID	Functions of SICOVE system	Defined Functions
1	Login to the system	Yes	10	Sell vehicle	**No**
2	Create user	Yes	11	View vehicle history	Yes
3	Edit user	Yes	12	View user history	Yes
4	Remove user	Yes	13	See income	Yes
5	Add local	Yes	14	View users	Yes
6	Modify local	Yes	15	View all local	Yes
7	Remove Local	Yes	16	View all vehicles	Yes
8	Set vehicle for sale	**No**	17	Generate quotation	**No**
9	Modify vehicle in system	Yes			

For DE metric, the SICOVE system works without requires inputs from other system. This means that is not dependant on other systems to perform their functions, so the connection between the SICOVE and other systems is not applicable. Thus, it is not possible to apply this metric in this case study.

For Complex and CR metrics, we identified 10 interfaces offered by SICOVE, 53 operations, and 71 parameters on all the graphical user interfaces offered, giving a result of 5.3 for Complex and 7.1 for CR metrics. In addition, the presentation view has been specified (e.g., see Figure 6, which presents the patterns to create the graphical user interface of new client service). The services with the arguments owned specified in Figure 3 are specified in Figure 5 as service interaction units. These results give us an indication of the current status of Complex and CR of the SICOVE system. These results indicate a normal level of complexity and customizability ratio to this system, due to it has the basic functions and parameters for the system to work.

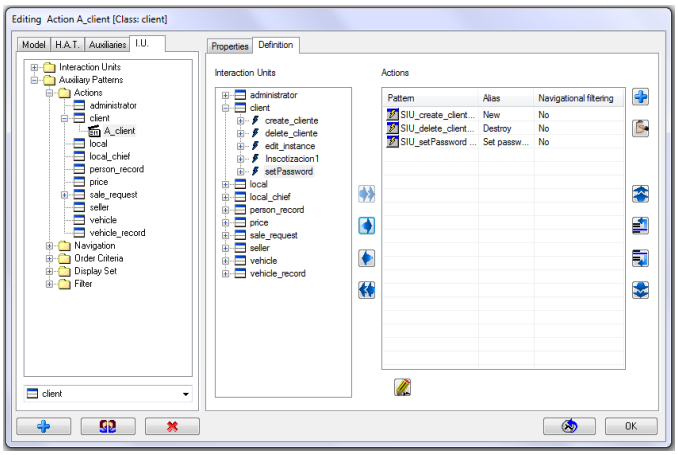

Figure 6. Example of SICOVE presentation view

All these metrics were applied manually to study the SICOVE system, which was automatically generated by the Integranova tool. Even though the case is small, and consequently the data delivered by not too big, it is enough to

 ISBN: 978-1-61208-438-1

understand the applicability of the ISO 9126 metrics to a particular MDD project. Thus, in this section we have exemplified the application of ISO 916 metrics to an MDD project, so that we verify the applicability of the selected metrics of ISO 9126 to an MDD project.

V. CONCLUSIONS AND FURTHER WORK

Software quality involves a strategy towards the production of software that ensures the user satisfaction, the absence of defects, the compliance with the budget and time constraints, and the application of standards and best practices for the software development. Thus, different techniques can be applied to the different artifacts used along the software development process. The ISO 9126 standard is a well-known quality model for software systems, so that in this paper we present an analysis of ISO 9126 regarding their applicability to MDD projects.

In particular, this paper presents an exploratory analysis of the ISO 9126 metrics that was performed in order to identify the metrics that could be used at early stages of software development cycle by analyzing the abstraction level of the conceptual models at which these metrics can be used. These early stages correspond to the specification of conceptual models for the analysis and design of software systems. In MDD projects, these conceptual models are located at different abstraction levels, which are the CIM, PIM, PSM or IM. In addition, these metrics have been used in an MDD project in order to evaluate their applicability. To calculate these metrics, the conceptual models of an industrial MDD approach were used. So that, we can conclude that these ISO 9126 metrics allow the early evaluation of quality of MDD projects, i.e., these metrics are useful for MDD projects.

Nevertheless, in MDD approaches there are many edges where is still possible to make a contribution to improve the quality evaluation of MDD projects, for instance extending the analysis to modeling languages, modeling tools, and modeling transformations, i.e., evaluating the quality of projects generated in MDE environments. Thus, immediate future work considers the inclusion of other metrics in order to have a well-defined set of metrics that conforms the basis of a quality model for MDD. And, later, further work considers the quality evaluation of MDE projects. We are aware that additional evaluation of our proposal to real development scenarios is necessary. Therefore, we consider as future work the development of empirical studies to evaluate the effectiveness and benefits of using these metrics under MDD approaches in real MDD projects.

ACKNOWLEDGMENT

This work was funded by Universidad Diego Portales and the FONDECYT project TESTMODE (Ref. 11121395, 2012-2015).

REFERENCES

[1] O. Pastor, J. Gómez, E. Insfrán, and V. Pelechano, "The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming", Information Systems, vol. 26, 2001, pp. 507–534.

[2] B. Selic, "The Pragmatics of Model-Driven Development", IEEE Software, vol. 20, 2003, pp. 19–25.

[3] OMG. MDA Products and Companies. Available: [retrieved: October, 2015] http://www.omg.org/mda/committed-products.htm

[4] OMG, "MDA Guide Version 1.0.1", 2003.

[5] ISO/IEC, "ISO/IEC 9126-1, Software Eng. – Product Quality – Part 1: Quality model", 2001.

[6] ISO/IEC, "ISO/IEC 9126-2, Soft. Eng. – Product Quality – Part 2: External metrics", 2003.

[7] ISO/IEC, "ISO/IEC 9126-3, Soft. Eng. – Product Quality – Part 3: Internal metrics", 2003.

[8] ISO/IEC, "ISO/IEC 9126-4, Soft. Eng. – Prod. Qual. – Part 4: Quality-in-Use metrics", 2004.

[9] F.D. Giraldo, S. Espana, and O. Pastor, "Analysing the concept of quality in model driven engineering literature: A systematic review". IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), 2014, pp 1–12.

[10] A. Fernandez, E. Insfran, and S. Abrahão, "Towards a Usability Evaluation Process for Model-Driven Web Development", I-USED'09, Uppsala, Sweden, 2009, pp.1-6.

[11] B. Marín, G. Giachetti, O. Pastor, and A. Abran, "A Quality Model for Conceptual Models of MDD Environments", Advances in Software Engineering, vol. 2010 - Article ID 307391, 2010, pp. 1-17.

[12] S. Abdulhadi, "i* Guide version 3.0", 2007.

[13] i*. Wiki Web Page. Available: [retrieved: October, 2015] http://istar.rwth-aachen.de/

[14] OMG, "Unified Modeling Language (UML) 2.4.1 Superstructure Specification " 2011.

[15] OMG, "Business Process Model and Notation (BPMN) 2.0", 2011-01-03 2011.

[16] M. Kardoš and M. Drozdová, "Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA)", Journal of Information and Organizational Sciences, vol. 34, 2010, pp. 89-99.

[17] B. Brahim, E. B. Omar, and G. Taoufiq, "A methodology for CIM modelling and its transformation to PIM", Journal of Information Engineering and Applications, vol. 3, 2013, pp. 1-21.

[18] M. Ruiz, Ó. P. López, and S. E. Cubillo, "A Traceability-based Method to Support Conceptual Model Evolution", CEUR-WS.org, 2014, pp-1-8.

[19] B. Marín, G. Giachetti, O. Pastor, "Applying a Functional Size Measurement Procedure for Defect Detection in MDD Environments" 16th European Conference EUROSPI 2009, Vol. CCIS 42, Springer-Verlag, 2009, pp. 57-68

[20] B. Marín, G. Giachetti, O. Pastor, and T. E. J. Vos, "A Tool for Automatic Defect Detection in Models used in Model-Driven Engineering", 7th International Conference on the Quality of Information and Communications Technology (QUATIC), Oporto, Portugal, 2010, pp. 242-247.

[21] G. Giachetti, B. Marin, and X. Franch, "Using Measures for Verifying and Improving Requirement Models in MDD Processes", 14th International Conference on Quality Software (QSIC), 2014, pp. 164-173.

[22] ISO, "ISO 5725-2 – Accuracy (trueness and precision) of Measurements Methods and Results – Part 2: Basic Method for the Determination of the Repeatability and Reproducibility of a Standard Measurement Method", 1994.

[23] O. Pastor and J. C. Molina, Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling, 1st edition ed. New York: Springer, 2007.

[24] O. Pastor, G. Giachetti, B. Marín, and F. Valverde, "Automating the Interoperability of Conceptual Models in Specific Development Domains", in Domain Engineering: Product Lines, Languages, and Conceptual Models, Springer, 2013, pp. 349-374.

[25] A. Mattsson, B. Lundell, B. Lings, and B. Fitzgerald, "Linking model-driven development and software architecture: a case study", IEEE Transactions on Software Engineering, vol. 35, 2009, pp. 83-93.

[26] P. Baker, Z. R. Dai, J. Grabowski, Ø. Haugen, I. Schieferdecker, and C. Williams, Model-Driven Testing: Using the UML Testing Profile: Springer-Verlag 2008.

[27] Integranova. (2015). Web Page. [retrieved: October, 2015] http://www.integranova.com

 ISBN: 978-1-61208-438-1