Assessing the Quality of User-Interface Modeling Languages

Francisco Morais[1] and Alberto Rodrigues da Silva[1,2]
[1]*Instituto Superior Técnico — Universidade de Lisboa, Lisbon, Portugal*
[2]*INESC— ID, Lisbon, Portugal*
{francisco.morais, alberto.silva}@tecnico.ulisboa.pt

Keywords: Appropriateness, Domain-Specific Languages, Evaluation, Quality, User-Interface Modeling Languages.

Abstract: Model-Driven Development (MDD) is an approach that considers model as first citizen elements in the context of software development. Since there are so many modeling languages, there is a need to compare them and choose the best for each concrete situation. The selection of the most appropriate modeling language may influence the output's quality, whether it is only a set of models or software.
This paper introduces ARENA, a framework that allows to evaluate the quality and effectiveness of modeling languages. Then we will apply ARENA to a specific subset of User-Interface Modeling Languages (namely UMLi, UsiXML, XIS and XIS-Mobile), taking into account some of their characteristics and the influence they have when models are generated.

1 INTRODUCTION

A modeling language might be classified as general-purpose (GPML) or domain-specific modeling language (DSML) (van Deursen et al., 2000; Luoma et al., 2004; Mernik et al., 2005; Kosar et al., 2010). A GPML is characterized by having a greater number of generic constructs, which encourages a wider and widespread use in different fields of application. UML or SysML are popular examples of GPMLs by providing large sets of constructs and notations used for specifying and documenting, respectively, software systems according to the object-oriented paradigm, or for system engineering. On the other hand, DS(M)Ls tend to use few constructs or concepts that are closer to its application domain.

Since a DS(M)L is expressed using domain concepts, it is normally easier to read, understand, validate and communicate with, facilitating cooperation between developers and domain experts. Moreover, some argue that DS(M)Ls can improve productivity, reliability, maintainability and portability (van Deursen et al., 2000). However, the use of a DS(M)L can raise some problems, such as the cost of learning, implementing and maintaining a new language, as well as the support tools to develop with it (Mernik et al., 2005).

Either for GPMLs as for DSMLs, nowadays there is a great variety of modeling languages. For instance, BPMN is specific to business process design although UML's activity diagram is also adequate for this purpose (Object Management Group, 2013; Object Management Group, 2011), DEMO is specific to enterprise architecture (Dietz, 2001), XIS to Interactive Applications, and Petri Nets to Distributed Systems (Desel and Juhás, 2001). Since there are more languages and approaches than domains, this results in overlapping effort for researchers and disorientation for modelers.

In order to help professionals selecting the most suitable modeling language to work on so many specific and different contexts, evaluation frameworks are needed. An evaluation framework is a set of properties, metrics, concepts and other parameters that compare and assert the languages' characteristics. For this work, we have proposed ARENA Framework to enlighten individuals that work with models on this area. Our goal in this article is to help the user choosing the most appropriate modeling language, in order to assure a quality output, to feel more supported and to work in a more rational way.

In this paper we propose a framework to evaluate and compare the quality of User-Interface Modeling Languages (UIMLs), focused on either their general and specific characteristics. The definition of a modeling language involves multiple aspects or facets which have to be taken into consideration, namely abstract syntax, concrete syntax, and semantics.

UIMLs are DSMLs that are specifically used for modeling the user interface of desktop, web or mobile applications, supporting the design and implementation phases (Achilleos et al., 2008). They denote conceptual abstractions that are present on user interfaces (Frank, 1996). This paper analyses and compares the main properties of four UIMLs, namely: UMLi (Silva, 2002), UsiXML (Limbourg et al., 2004), XIS (Silva et al., 2003)(Martins and Silva, 2007)(Silva et al., 2007) and XIS-Mobile (Ribeiro and Silva, 2014b)(Ribeiro and Silva, 2014a). We have selected them due to their support, complete documentation, project visibility and availability of their papers.

We are aware of other UIMLs such as DiaMODL (Trætteberg, 2008), IFML (Object Management Group, 2014), MARIA XML (Paternó et al., 2009) or WebML (Ceri et al., 2002). However, we acknowledge that they don't fit the domain and our criteria as well as the previous four. WebML had strong influence in IFML, when Object Management Group created this language and that IFML's most recent version (from 2014) is still in Beta. Therefore, we consider that there were too much similarities between both and comparing them would not prove effective. Regarding DiaMODL and MARIA XML, the first one is too focused on Dialog Modeling, Dataflow and State logic while the second is not quite appropriate due to its models' development being highly dependable in Service-Oriented Architectures and Web Services, which implies exploiting annotations at design time and the language itself at runtime to support dynamic generation of user interfaces, thus limiting usability and analysis.

This paper is organized in 6 sections: Section 1 gives a brief explanation and context of this work. Section 2 introduces the background for this research, that includes a definition of Quality and its importance for this work, a brief overview of Modeling Languages and an analysis of other Evaluation Frameworks. Section 3 presents the proposed ARENA Framework to the previously referred problem. Section 4 evaluates the quality of four UIMLs with the ARENA's properties and metrics. Section 5 discusses the evaluation made based on the ARENA Framework. Finally, section 6 concludes this paper.

2 BACKGROUND

This section introduces the background of this research, namely Quality, Modeling Languages and Evaluation Frameworks.

2.1 Quality

Regarding a model that may generate software, its quality can be divided into internal and external (International Organization for Standardization and International Electrotechnical Commission, 2001).

Internal quality consists on the characteristics of the software product from an internal view. Internal quality requirements are used to specify properties of interim products. They can include static and dynamic models, other documents and source code (International Organization for Standardization and International Electrotechnical Commission, 2001).

External quality is defined by the characteristics of the software product from an external view. It is the quality of the executed software, which is typically measured and evaluated while testing in a simulated environment with simulated data using external metrics. These metrics should be aligned with the external quality requirements, which goal is to specify the required level of quality from the external view (International Organization for Standardization and International Electrotechnical Commission, 2001).

Quality is not yet a common priority when developing a Domain-Specific Language (DSL). The focus is currently on getting up a systematic development of these Modeling Languages, a goal that has not yet been reached because it has been doing more study of the technical aspects of DSLs' design and implementation, such as: case studies and technical reports on individual DSLs; design approaches and techniques for implementing DSLs; and integrating DSLs with other developmental approaches (Strembeck and Zdun, 2009).

2.2 Modeling Languages

A modeling language (ML) is a set of words and symbols, supported by validation rules and semantics, which make it possible to create models or diagrams. It can be graphical/visual or textual, depending on its scope and domain (He et al., 2007).

A modeling language is created from a metamodel within a certain domain, therefore, it can be seen as a model of a modeling language (Ma et al., 2004). In order to give the modeling languages functionality to use and evaluate the models, different types of mechanisms are included in the metamodeling process (algorithms, generic mechanisms, specific mechanisms and hybrid mechanisms) (Karagiannis and Kühn, 2002). In other words, the metamodel is able to highlight the properties of the model, derived from its capacity of abstraction. There are several metamodeling approaches. The most commonly used is Meta Object Facility (MOF) and it has been used, for instance, in the development of UML and SysML.

In terms of representation type, MLs can be divided in two groups: graphical and textual. On one hand, graphical MLs use a diagram technique with named symbols that represent concepts and lines that connect the symbols and represent relationships and various other notation to represent constraints. On the other hand, textual MLs use standardized keywords accompanied by parameters or natural language terms and phrases to make computer-processable expressions. Considering modeling approaches, in the field of computer science, more specific types of MLs have recently emerged, due to new challenges on very different areas, namely: Algebraic, Behavioural, Discipline-Specific, Domain-Specific, Framework-Specific, Object-Oriented or Virtual Reality.

An ML is very important because it helps to elucidate their stakeholders that have modeling expertise to understand what and how they can make a representation of the system-of-interested. Also, its features such as syntax, abstraction and compatibility with programs can facilitate its use and its correspondent metamodels may give a perception about the adequacy of the language to the concrete situation.

2.3 Evaluation Frameworks

Hereby, we are exposing three frameworks that have been developed to evaluate the quality of conceptual models and three business process modeling languages.

SEQUAL is a framework that was presented in 1994 with the goal of evaluating systematically the quality of information systems and other conceptual models through the notions of syntactic, semantic and pragmatic applied to them (Krogstie et al., 1995). One year later it was extended to more three features, inspired by FRISCO's six semiotic layers of communication. It is considered the first framework to have the objective on evaluating models' quality.

Essentially, the first version of the framework only had four concepts — Model, Domain, Language and Audience interpretation — and six pair relations between them. Its extension, in 1995, has brought the addition of a new entity (Participant knowledge) and the creation of four relationship types. In 2012, Krogstie makes a third instance of his framework and adds "deontic" as a new quality criteria that focuses on what is right and needed for the organization's goals (Krogstie, 2012). SEQUAL's formulas are based on the audience, the language, the model, the domain, the audience's knowledge and the audience's interpretation.

On 2005, a paper about focusing the standardization of business process modeling was presented. It intended to compare some Modeling Languages that could be used for this context. The three selected BPM languages were EEML, UML and BPMN, and their framework comprised the following items: Goals of modeling task, language extension, domain, externalized model, knowledge of the stakeholders, the social actors' interpretation and the technical actors' interpretation (Nysetvold and Krogstie, 2005).

On 1997, Teeuw and van der Berg published a paper about their perspective on general quality criteria for conceptual models and a framework that was used to evaluate the redesign of business processes. Like SEQUAL, they also defend that a good, quality model must have syntactic, semantic and pragmatic qualities and, considering that the concepts can be captured with a suitable language (assuring the first quality), they present as criteria for the second and the third: completeness, inheritance, clarity, consistency, orthogonality and generality (Teeuw and van der Berg, 1997). Considering the Testbed evaluation framework, it was applied to behaviour models and it was designed with 4 dimensions: Functionality, Ease of use, Business Process Redesign (BPR) trajectory and General.

Although they use interesting criteria, the related work frameworks don't satisfy the goal of this work because they evaluate other matters instead of UIMLs, respectively, the Quality of Conceptual Models, the Quality of Business Process and Business Process redesign.

The selection of the appropriate Modeling Language influences the quality of the final output, whether it is a model or a software application (International Organization for Standardization and International Electrotechnical Commission, 2008; Krogstie et al., 1995; Rech and Bunse, 2009). Quality integration in software development processes that use models, such as Model-Driven Software Development, is the ultimate reason that led us to propose this framework. ARENA displays a Framework that, after analysing the MLs' characteristics, shows the evaluation of each one and helps the user to choose the most appropriate ML, in order to assure a quality output. It is the most appropriate evaluation framework for modelling languages, because it is not only divided into general and specific properties (which takes into account the considered domain), but also because it is an extensive comparison and assessment, resembling the CMS Matrix.

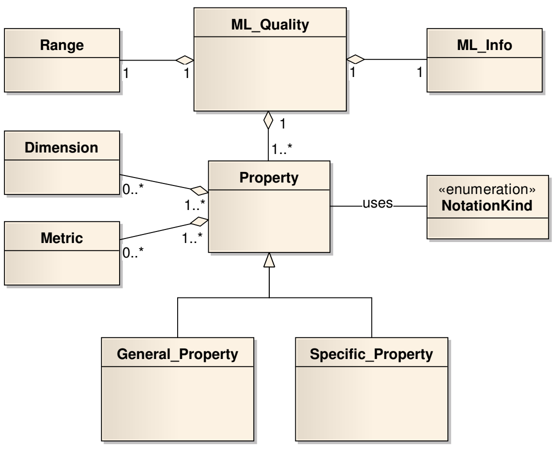

Figure 1: ARENA Framework — Main Concepts

3 THE ARENA FRAMEWORK

Figure 1 shows that a Modeling Language's Quality is the central class. It contains the final quantitative output. This class uses the Modeling Language's information as an input and has a Range.

This one is calculated using a formula that receives as an input the values of the quality and quantity evaluations (respectively represented as Dimension and Metric classes) and multiplies each by a previously defined weight. The returned output is a value that represents the language's Quality within a rating scale, as explained hereinafter on section 4. All domains share the General Properties, as opposite to Specific Properties, that differ from each other (e.g. UIMLs, BPMLs etc). The Notation Kind enumeration is developed as a droptext, so the user can select one option from the displayed list. The proposed framework intends to bridge that gap and it is composed by general and specific properties.

3.1 General Properties

Interoperability. The language must be fully compatible with several tools, i.e. should allow to do the same tasks and diagrams in different software tools. Also, it shall be possible to combine with another modeling and programming languages (PLs) and tools, being supported by mechanisms that guarantee those possibilities. Dimensions and Metrics: Number of Compatible Applications (International Organization for Standardization and International Electrotechnical Commission, 2008), Number of Integration Mechanisms (Karagiannis and Kühn, 2002) and Tool Supportability.

Notation. A Notation or concrete syntax is a set of signs that enable to represent models. A modeling language may have two types of notation: graphical/visual or textual. Dimensions: Representation Type and Supporting Mechanisms.

Size. Completeness is one of the biggest challenges regarding the development of modeling languages and respective models (Baader et al., 2003; Hoppenbrouwers et al., 2005; Teeuw and van der Berg, 1997). This property states that the language's meta-model shall include the most important concepts. Therefore, the meta-modeling approach is essential for assuring that the modeling languages allow producing concrete quality models. Metrics: Number of Views, Number of Classifiers and Number of Relationships.

3.2 Specific Properties

Application Actions. This property lists a series of actions that the generated application can support, in different contexts.

Other Features. It is a set of properties of the UIMLs that provide additional information about them. In this paper, they will be considered, but not weighted to calculate the quality.

Pattern Usage. This property displays a list of simple and reusable templates that help the user solving common problems that appear during the design phase.

Tool Support - Model to Model (M2M) Transformations. This property tells if the compatible programs are capable of transforming the language's models format into another format or if the language can produce more than one format.

Tool Support - Model to Text (M2T) Transformations. This property is a key aspect of model-driven development. It refers the language's ability or inability to generate textual artefacts from models and if so, which mechanisms or techniques make it possible.

Tool Support - Validation. This property has the purpose of listing which systems, applications or mechanisms can analyse the models created by the user and validate them.

User Interactions. This property intends to show which ways users can interact with the application, whether it is with mouse, keyboard, touch or other means.

Widget Types. This property lists a set of graphical user interface elements (either structural or behavioural) that the language makes available for the designer.

4 APPLYING ARENA TO EVALUATE USER-INTERFACE LANGUAGES

Table 1 summarizes the main properties of four analysed UIMLs — UMLi (Silva, 2002), UsiXML (Limbourg et al., 2004), XIS (Silva et al., 2003)(Martins and Silva, 2007)(Silva et al., 2007) and XIS-Mobile (Ribeiro and Silva, 2014b)(Ribeiro and Silva, 2014a). The columns Number of Views, Number of Classes and Number of Relationships represent the metrics with the same name, from ARENA's property Size. The ARENA property Notation is represented by two dimensions: Representation Type and Supporting Mechanisms. The column Tool Supportability refers to a dimension from ARENA's property Interoperability, as well as metrics Number of Compatible Applications and Number of Integration Mechanisms. The remaining columns have exactly the same name and meaning of the specific properties, as shown on the previous section. The quality value of each language is determined by the formula below, designated as the ARENA General Equation of Quality.

$$
\begin{gathered}
QualityLanguage_n = \sum_{i=1}^{j} w_i * r_i = \\
= w\_absStxProps * r\_absStxProps + w\_concStxProps * r\_concStxProps + w\_specProps * r\_specProps = \\
= w\_numViews * r\_numViews + w\_numClassifs * r\_numClassifs + w\_numRelats * r\_numRelats + \\
+ w\_repType * r\_repType + w\_suppMechs * r\_suppMechs + w\_toolSupp * r\_toolSupp + \\
+ w\_numAppActs * r\_numAppActs + w\_numUserInts * r\_numUserInts + w\_numPatterns * r\_numPatterns + \\
+ w\_Valid * r\_Valid + w\_M2MTrfs * r\_M2MTrfs + w\_M2TTrfs * r\_M2TTrfs + \\
+ w\_numCompApps * r\_numCompApps + w\_numIntMechs * r\_numIntMechs + w\_numWidgets * r\_numWidgets
\end{gathered}
$$

The notation used on the formula above has the following meaning: i — Framework's metric/dimension; j — Sum of the number of metrics with the number of dimensions; n — Language's name; r — Metric's/dimension's rate; w — Metric's/dimension's weight. The rating each property can have is the following: 1 — Very Low; 2 — Low; 3 — Medium; 4 — High; 5 — Very High.

If quantifiable and with equal level of importance, each property's value is compared towards that property's average value in this work. Its rating is given according to how far it is from the average, upwards or downwards, according to the following ranges: $Value < AvgValue - 1/2 * AvgValue$; $Value \in [AvgValue - 1/2 * AvgValue, AvgValue - 1/4 * AvgValue[$; $Value \in [AvgValue - 1/4 * AvgValue, AvgValue + 1/4 * AvgValue]$; $Value \in]AvgValue + 1/4 * AvgValue, AvgValue + 1/2 * AvgValue]$ and $Value > AvgValue + 1/2 * AvgValue$.

The exceptions (non-quantifiable properties) are: Representation Type — The most popular value has higher rating than the least; Supporting Mechanisms — Since it's very specific for each language, this rating reflects easiness to understand and extend the metamodel; Tool Supportability — It may be unique for each language, so its rating is focused on the programs' general usability and functionality; and Tool Support (Validation, Model to Model and Model to Text) — Since the programs referred on the Tool Supportability property generate models, we assumed that Validation is correctly done, as well as the transformations.

The languages in which we don't know how these characteristics are covered were assigned 1 (Very Low) and the languages that don't have these features were assigned 2 (Low). Each rated property will be multiplied by its weight, as referred on section 3, and the sum of this products will return the language's quality, according to the ARENA General Equation of Quality.

5 DISCUSSION

Having four UIMLs compared, it is possible to see that, for the same domain, the languages do not differ much in terms of number of views and number of relationships, but they do when it comes about supporting mechanisms and tool supportability. This can be explained by their maturity and popularity. Also, there are patterns concerning number of classifiers and no-

Table 1: UIMLs comparison based on the ARENA Framework

Language / Property	Property Group	UMLi	UsiXML	XIS	XIS-Mobile	Average (UIMLs)
Number of Views	Abstract Syntax	4	5	6	6	5
Number of Classifiers	"	13	12	18	46	22
Number of Relationships	"	11	20	12	16	15
Representation Type	Concrete Syntax	Graphical	Textual	Graphical	Graphical	—
Supporting Mechanisms	"	Core meta-modelling	XML Metamodel and Cameleon reference framework	UML Profile	UML Profile	—
Tool Supportability	"	ARGOi	Eclipse, Visual Studio, Notepad++, Kate, W3schools.com and others	ProjectIT Studio	Enterprise Architect	—
Application Actions	Specific Properties	OK, Cancel, Search, Back, Next, Up, Down, Quit, other customizable actions	(N/A)	CRUD, OK, Cancel, Navigate, Select, Close, Associate, Dissociate, other customizable actions	CRUD, OK, Cancel, Delete All, Open Browser, Web Service, Navigate, Select, other customizable actions	>8
User Interactions	"	Click, Select, (Keyboard) Type, Scroll	Move pointer, Click, Double click, Depress, Release, Drag over, Drag drop, Focus, Select, Choose, Toggle, View	Click, Select, Drag, (Keyboard) Type	Tap, Double Tap, Long Tap, Swipe, Pinch, Stretch, (Touchscreen) Type	7
Pattern Usage	"	Abstract Presentation Pattern, Concrete Interaction Object	Concepts & Task Model, Abstract UI, Concrete UI, Final UI, Inter-model mapping, Context translation	Composite, Single Choice, Multiple Choice, List Selection, Continuous Filter, Menu Navigation, Grid Layout, Tab Menu, Tabular Set, Double List	Composite, Single Choice, Multiple Choice, List Selection, Single Text Entry, Multiple Text Entry, Springboard, List Menu, Tab Menu, Option Menu and 10 more	10
Tool Support - Validation	"	ARGOi validates both UML and UMLi models, due to both grammars are specified in terms of the UML metamodel	(N/A)	Eclipse .NET, .NET Framework and Project IT's three components: Requirements, UML Modeler and MDD Code Generator	XIS-Mobile Framework, namely EA's Model Validator, supports model validation	—
Tool Support - M2M Transformations	"	No	(N/A)	No	Yes, using Enterprise Architect's MDG technologies	—
Tool Support - M2T Transformations	"	No	(N/A)	Yes, defining architectures, templates, and interface generation processes that are compatible with Windows Forms.NET and ASP.NET platforms	Yes, using XMI-validated and generated models, Acceleo and the OS's compatible programming languages. It can generate Java, C#, Objective-C, XML and XAML	—
Number of Compatible Applications	"	1 (ArgoUML)	1 (Eclipse)	1 (ProjectIT Studio)	1 (Enterprise Architect)	1
Number of Integration Mechanisms	"	3 (ARGOi, OCL rules and LOTOS rules)	4 (MDG Technologies, XML Parser, UsiGesture and UsiDistrib)	2 (Eclipse .NET and .NET Framework)	2 (MDG Technologies, XML Parser)	3
Widget Types	"	Label, Text field, Combo box, Selectable list, Button	Push button, List box, Check box, Window, Panel, Table, Cell, Dialog box, Embeded multimedia, Menu, Spin button	Button, Text box, List, Menu, Window, Link, Search bar, Checkbox, Radius button, Drop-down list, Form, Dialog, Label and 15 more	Button, Text box, List, Menu, Window, Link, Search bar, Checkbox, Radius button, Drop-down list, Label, Image, Date Picker and 9 more	17
Other Features	"	It is possible to model Activity Diagrams in UMLi, using the Use Case Diagram and the InitialInteraction construct.	It implements the μ7 concept, as it is Device-, User-, Culturality-, Organization-, Context-, Modality- and Platform-Independent.	Supports Windows Forms.NET, ASP.NET and JSP, using Model-to-Text transformations.	Model validation uses a set of rules defined in C#, implemented with EA's Automation Interface. It is possible to generate User-Interfaces View models on EA.	—

tation, both with a single exception.

If the Number of Views seems to be balanced for these four languages (all have between 4 and 6), it is not possible to conclude the same about the Number of Classifiers, because UsiXML, UMLi and XIS have, respectively, 12, 13 and 18 and XIS-Mobile has 46, resulting in a difference of 34 items to the lowest number. Less unbalanced is the Number of Relationships — it varies from UMLi's 11 to UsiXML's 20.

For Representation Type, clearly there is a preference for Graphical in disfavour of Textual, probably due to easiness to see the models in a clearer way and according to the WYSIWYG paradigm. About Supporting Mechanisms, as expected due to their nature, XIS and XIS-Mobile have the same value, opposing to UMLi that is an extension to UML and therefore uses a simpler approach and UsiXML that, being based on XML, results in a different architecture that is complemented by Cameleon. Considering Tool Supportability, all languages follow different tendencies, with a clear advantage to UsiXML. This might be due to XML's universality (as it is used in RSS, SOAP messages and XHTML), it is both human- and machine-readable and it is capable of representing all Unicode characters.

XIS is the top language concerning to Application Actions, since it allows the software to do at least 9 different actions. Contrasting with that, UsiXML is the better one in terms of User Interactions, since it provides 12 gestures that can be done, while XIS and UMLi only have 4 defined. The language that includes more Patterns is XIS-Mobile, Composite being the only one of Structural Design type and the remaining 19 of UI Design, that is 10 more than XIS. Still about this property, UsiXML has 4 which are based on Cameleon reference framework and UMLi has 2 patterns, each for a UML package that was extended.

About Tool Support - Validation, each language has its own method, always influenced by the compatible applications and the ones shown in Tool Supportability column, which in many cases, are the same. We don't know how it's done for UsiXML, but for XIS it is the most extensive process. Now referring to Tool Support - Model to Model Transformations, only XIS-Mobile has this feature implemented, taking advantage of EA's capabilities, while regarding Tool Support - Model to Text Transformations, XIS and XIS-Mobile provide this MDD-core feature, as opposed to UMLi, with a great focus on Web technologies. Again, we know nothing about these two properties on the UsiXML case.

Now regarding the Number of Compatible Applications, all languages have the same value, therefore we considered than none stood out more than the other. It seems to be a bit contradictory to the Tool Supportability property. The difference is that the latter doesn't take into account the language's suitable program but the produced models' compatibility with other platforms. For the Number of Integration Mechanisms, despite some are easier to work with (XML Parser) than others (LOTOS rules), all languages have received the same rating, due to the number is close to the average of this property, in this work.

The attractive and useful Widget Types inform that XIS and XIS-Mobile have a higher number of items (28 and 22, accordingly), although not exactly the same. UsiXML stands with 11 and UMLi has only 5, which is expected from an extension that claims to "be a conservative extension of UML" and "should introduce as few new models and constructs into the UML as possible" (Silva, 2002).

According to Table 2, for this evaluation, XIS-Mobile has the highest quality value — 4 — among these four UIMLs, in the considered domain.

6 CONCLUSION

This paper presents an overview of academic research related to the problematic of choosing the best from several modeling languages and the quality assessment frameworks as a response to that problem. We have evaluated four UIMLs using ARENA, a framework oriented to that solution. In this case, ARENA's most adequate dimensions and metrics were used, along with User-Interface specific properties, defined for this purpose and context. With this analysis, it is possible to list the features in which each UIML stands out from the others, either because they are better or they are unique. Clearly there is a tendency to bet on graphical notations. This can be due to guidelines that compose the well known Model-Driven Engineering. Also an advantage, is to use these graphical models to implement the alternative approach to produce software: Model-Driven Software Development.

The metamodel types chosen vary very much from modeling language to modeling language, since some use MOF, others BNF or UML Profiles, and even MVC is still used. Another possible choice is to extend other ML's metamodel. Tool support is very important, in terms of usability and functionality. They are also responsible to render and validate the produced models, so this aspect can be the quality bottleneck. In terms of look-and-feel, it is very important for the designer to have available design patterns

Table 2: UIMLs evaluated with ARENA Framework

Property \ Language	Weight	UMLi	UsiXML	XIS	XIS-Mobile
Number of Views	0.033	3	3	3	3
Number of Classifiers	0.033	2	2	3	5
Number of Relationships	0.033	2	4	3	3
Representation Type	0.033	4	3	4	4
Supporting Mechanisms	0.033	3	3	3	3
Tool Supportability	0.033	3	4	2	4
Application Actions	0.1	3	1	3	3
User Interactions	0.1	2	5	2	3
Pattern Usage	0.1	1	2	3	5
Tool Support - Validation	0.1	3	1	3	3
Tool Support - M2M Transformations	0.1	2	1	2	4
Tool Support - M2T Transformations	0.1	2	1	4	4
Number of Compatible Applications	0.05	3	3	3	3
Number of Integration Mechanisms	0.05	3	3	3	3
Widget Types	0.1	1	2	5	4
Other Features	—	—	—	—	—
Total Quality	1	2	2	3	4

and builder tools, so he can produce an attractive and easy interface. He must also be able to choose between several actions and widgets, so the mobile, web or desktop software application can have an attractive layout and great impact.

In the future, ARENA will also be used to compare and evaluate General-Purpose Modeling Languages, such as UML and SysML or Business Process Modeling Languages, namely BPMN and DEMO, always taking into account the general and the domain specific characteristics.

ACKNOWLEDGEMENTS

This work was partially supported by the ARENA 2012 IBM Country Project, and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UID/CEC/50021/2013 and EXCL/EEI- ESS/0257/2012 (DataStorm).

REFERENCES

Achilleos, A., Yang, K., Georgalas, N., and Azmoodech, M. (2008). Pervasive service creation using a model driven petri net based approach. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International*, pages 309–314.

Baader, F., Calvanese, D., McGuiness, D., Nardi, D., and Patel-Schneider, P. (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M. (2002). *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Desel, J. and Juhás, G. (2001). *What Is a Petri Net?*, volume 2128 of *LNCS*. Springer-Verlag.

Dietz, J. (2001). DEMO: Towards a discipline of organisation engineering. *European Journal of Operational Research*, 128:351–363.

Frank, M. R. (1996). *Model-based User Interface Design by Demonstration and by Interview*. College of Computing, Georgia Institute of Technology 1996. Directed by James Foley.

He, X., Ma, Z., Shao, W., and Li, G. (2007). A Metamodel for the Notation of Graphical Modeling Languages. In *Proceedings of the 31st Annual International Computer Software and Applications Conference - Volume 01*, volume 1 of *COMPSAC '07*, pages 219–224. IEEE Computer Society.

Hoppenbrouwers, S., Proper, E., and van der Weide, T. P. (2005). A Fundamental View on the Process of Conceptual Modeling. In *Conceptual Modeling - ER 2005*, volume 3716, pages 128–143. Springer-Verlag.

International Organization for Standardization and International Electrotechnical Commission (2001). ISO/IEC 9126-1:2001(E) Quality Model.

International Organization for Standardization and International Electrotechnical Commission (2008). ISO/IEC CD 25010.2 Software and Quality in use models.

Karagiannis, D. and Kühn, H. (2002). Metamodelling Platforms. In *Proceedings of the Third International Conference EC-Web 2002 - Dexa 2002*, volume 2455, pages 182–195. Springer-Verlag.

Kosar, T., Oliveira, N., Mernik, M., Pereira, M. J. V., Črepinšek, M., da Cruz, D., and Henriques, P. R. (2010). Comparing general-purpose and domain-specific languages: An empirical study. In *Computer Science and Information Systems*, volume 7. University of Novi Sad, Serbia.

Krogstie, J. (2012). *Model-Based Development and Evolution of Information Systems: A Quality Approach*, chapter 5, pages 249–280. Springer-Verlag London.

Krogstie, J., Lindland, O. I., and Sindre, G. (1995). Defining quality aspects for conceptual models. *Faculty of Electrical Engineering and Computer Science -*, The Norwegian Institute of Technology.

Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., and Trevisan, D. (2004). USIXML: A User Interface Description Language for Context-Sensitive User Interfaces.

Luoma, J., Kelly, S., and Tolvanen, J.-P. (2004). Defining domain-specific modeling languages: Collected experiences. In *OOPSLA 4th Workshop on Domain-Specific Modeling*. ACM.

Ma, H., Shao, W., Zhang, L., Ma, Z., and Jiang, Y. (2004). Applying OO Metrics to Assess UML Meta-models. In *UML 2004, LNCS*, volume 3273, pages 12–26. Springer-Verlag.

Martins, C. and Silva, A. R. (2007). Modeling user interfaces with the xis uml profile. In *Proceedings of the ICEIS 2007*.

Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys*, 37:316–344.

Nysetvold, A. G. and Krogstie, J. (2005). Assessing business processing modeling languages using a generic quality framework. Norwegian University of Science and Technology.

Object Management Group (2011). *OMG Unified Modelling Language (OMG UML), Infrastructure - Version 2.4.1*. Object Management Group.

Object Management Group (2013). *Business Process Model and Notation (BPMN) - Version 2.0.2*. Object Management Group.

Object Management Group (2014). *Interaction Flow Modelling Language (IFML) - FTF - Beta 2 - Revision 21*. Object Management Group.

Paternó, F., Santoro, C., and Spano, L. D. (2009). MARIA: A Universal, Declarative, Multiple, Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments. *ACM Transactions on Computer-Human Interaction*, 16:1–30.

Rech, J. and Bunse, C. (2009). *Model-Driven Software Development: Integrating Quality Assurance*. Information Science Reference.

Ribeiro, A. and Silva, A. R. (2014a). Evaluation of xis-mobile, a domain specific language for mobile application development. In *Journal of Software Engineering and Applications*, number 7 in 11. Scientific Research Publishing.

Ribeiro, A. and Silva, A. R. (2014b). Xis-mobile: A dsl for mobile applications. In *Proceedings of ACM SAC 2014 Conference*. ACM.

Silva, A. R., de Sousa Saraiva, J., Silva, R., and Martins, C. (2007). XIS - UML Profile for eXtreme Modelling Interactive Systems. In *Proceedings of the MOMPES 2007*. IEEE Computer Society.

Silva, A. R., Lemos, G., Matias, T., and Costa, M. (2003). The XIS Generative Programming Techniques. In *Proceedings of the 27th COMPSAC Conference*. IEEE Computer Society.

Silva, P. P. (2002). *Object Modelling of Interactive Systems: The UMLi Approach*. PhD thesis, University of Manchester.

Strembeck, M. and Zdun, U. (2009). An approach for the systematic development of domain-specific languages. *Software Practice and Experience*, 39:1253–1292.

Teeuw, W. B. and van der Berg, H. (1997). On the Quality of Conceptual Models. In *16th International Conference on Conceptual Modeling - ER'97*, volume 1331, pages 1–18. Springer-Verlag.

Trætteberg, H. (2008). Integrating Dialog Modeling and Domain Modeling - the Case of Diamodl and the Eclipse Modeling Framework. In *Journal of Universal Computer Science*, volume 14, pages 3265–3278. J.UCS.

van Deursen, A., Klint, P., and Visser, J. (2000). Domain-specific languages: An annotated bibliography. *ACM SIGPLAN Notices*, 35(6):26–36.