ARENA: Asserting the Quality of Modeling Languages

Francisco de Freitas Vilar Morais

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Alberto Manuel Rodrigues da Silva

Examination Committee

Chairperson:	Prof. José Luís Brinquete Borbinha
Supervisor:	Prof. Alberto Manuel Rodrigues da Silva
Member of the committee:	Prof. André Ferreira Ferrão Couto e Vasconcelos

July 2015

Em memória da minha avó Maria Amélia de Freitas Vilar e restantes familiares,
pela força, exemplo e amor incondicional que sempre me deram.

Acknowledgments

I would like to thank my advisor, Prof. Alberto Silva, that supported and counselled me in every possible way. Without his knowledge on User-Interface and Business Process Modeling Languages, academic experience, commitment and perseverance, I couldn't have structured, focused and developed this work.

I must also thank my co-advisor, Mr. Andreas Schoknecht, for all the academic materials, drive and motivation that he has given me throughout this work while I was in Germany on the ERASMUS programme, as well as Prof. Jan Dietz, which contribution in ICEIS 2015 enlightened me to understand DEMO and its competing languages.

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).

I would also like to thank to my parents Maria José and António Manuel and my friends, for supporting me, giving me the strength to carry on and to remind me that hard work pays off. Also, I want to give a word of gratefulness to my brother Stephan, for his language skills contribution and professional background being an example to me and to my dear colleague Catarina Moreira for her great availability and high technical skills.

Thanks so much to everyone that, in one way or another, have shown interest or contribute to this thesis. May you all shine in your ways.

July 2015

Abstract

Nowadays, we assist at a growing number of mobile and desktop applications. Some of them are developed using programming languages, the traditional way, while some have been developed using other approaches, such as Model-Driven (Software) Development — MD(S)D. It considers models as first class 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 Thesis has the main purpose of creating and debating a framework to evaluate the quality and effectiveness of developed Domain-Specific Modeling Languages, taking into account their domains and the influence they have when models are created. It should also be useful for General-Purpose Modeling Languages.

Keywords: Appropriateness, Business Process Modeling Languages, Comparison, Domain-Specific Languages, Evaluation, Frameworks, Model-Driven (Software) Development, Modelling Languages, Quality, User-Interface Modeling Languages.

Resumo

Hoje em dia, assistimos a um número crescente de aplicações móveis e fixas. Algumas são desenvolvidas usando linguagens de programação, o método tradicional, ao passo que outras têm sido desenvolvidas usando outras abordagens, tais como o Desenvolvimento (de *Software*) Orientado ao Modelo. Esta considera os modelos como elementos de primeira classe no contexto do desenvolvimento de *software*. Uma vez que existem várias linguagens de modelação, surge a necessidade de compará-las e escolher a melhor para cada situação concreta. A escolha da linguagem de modelação mais adequada pode influenciar a qualidade do resultado final, seja este apenas um conjunto de modelos ou *software*.

Esta Tese tem como objectivo principal a criação e o debate de uma *framework* para avaliar a qualidade e a eficácia das Linguagens de Modelação de Domínio Específico desenvolvidas, tendo em conta os seus domínios e a influência que têm quando os modelos são criados. Também deverá ser útil para Linguagens de Modelação de Propósito Geral.

Palavras-Chave: Adequação, Avaliação, Comparação, Desenvolvimento (de *Software*) Orientado ao Modelo, *Frameworks*, Linguagens de Domínio Específico, Linguagens de Modelação, Linguagens de Modelação de Interfaces de Utilizador, Linguagens de Modelação de Processos de Negócio, Qualidade.

Contents

List of Tables

List of Figures

List of Acronyms

BPML	Business Process Modeling Language
BPMN	Business Process Model and Notation
DEMO	Design & Engineering Methodology for Organizations
MB-UIDE	Model-Based User Interface Development Environment
MDA	Model-Driven Architecture
MDD	Model-Driven Development
MDE	Model-Driven Engineering
ML	Modeling Language
SysML	Systems Modeling Language
UIDL	User-Interface Description Language
UIML	User-Interface Modeling Language
UML	Unified Modeling Language
UMLi	Unified Modeling Language for Interactive Applications
UsiXML	User Interface Extensible Markup Language
XIS	Extreme Modeling Interactive Systems

Chapter 1

Introduction

Since ever, the human species always felt the need to represent, collect and organize information. It is known that cave paintings are the prime form of visual representation. They show us actions that Homo Sapiens tribes used to do and that were essential for their survival (e.g. hunting, fishing and fighting), therefore we can say that these paintings were a language that they used to communicate. Formally, this means that we model because we have the need to represent the real world. Also, modeling enhances communication between people thus helping them answering questions [14]. Eventually, that kind of primate representation evolved into more complex types, such as paintings, mathematical models or blueprints. Until the 1990s systems design had a crucial and respected role in the data processing industry. From that decade on, standardization of hardware and software resulted in the ability to build modular systems[1]. In 1996, the USA's Department of Defense defined *system design* as "a process of defining the hardware and software architecture, components, modules, interfaces, and data for a system to satisfy specified requirements" [51].

1.1 Motivation

On the present days, we can describe modeling as a team process to build a view of reality. It can be seen as a tool for developing, managing and sharing knowledge. The goal of that process is to reach a phase where all the participants have agreed on a base of understanding and they see their personal semantics in the group's semantics [14]. Now, we can use dozens of Modeling Languages for several contexts. Depending on each one, some of the most easy ways of representing information are either graphical, textual or a combination of both. But the main doubt

[1] "Interactive Systems Design" course, University of Glasgow (http://www.dcs.gla.ac.uk/ johnson/teaching/isd/course.html)

persists: How can we assure quality when selecting a Modeling Language to work on a specific domain project?

1.2 Problem

There is a big concern about whether the chosen Modeling Language is the most adequate to a certain domain or not. Since DSLs are used to provide more quality to software systems [52], the question that is imposed is:

Is it possible to have a framework that provides and compares essential and accessory aspects of Modeling Languages, helping to assure quality models?

This work is intended to answer positively the question, but we need to take several aspects into account. However useful the idea of using Modeling Languages to generate software and documentation might sound, in order to be widely adopted, we need to have systematic and reproducible approaches and tools to support the Software Language Engineer before selecting the correct one. The context, the elements, the syntax or the compatible tools that characterize each modeling language can play an important role in the selection process of the language and shall contribute to produce quality models and successful, functional software [29]. This means that we need a rigorous framework to evaluate and compare how adequate the Modeling Languages (both Domain-Specific and General-Purpose) are for the MDSD process and the instance models themselves.

Only this way the Software Language Engineer will be able to work properly with the Domain Expert and produce a flawless DSL workbench.

1.3 Investigation Goals

Now that we have defined the problem, the main purposes of this work are:

- Define and discuss a framework that evaluates rigorously the quality of Modeling Languages, focusing on Domain-Specific Languages, but also applicable to General-Purpose Languages.

- Create a collaborative Web System for comparing the quality of several DSMLs.
- Apply this framework to some well-known Modeling Languages, namely BPMN, DEMO, UML, UMLi, UsiXML, XIS and XIS-Mobile.

1.4 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 includes the most important concepts which are critical for understanding this work. It describes extensively Modeling Languages as the main subject of study as well as Domains and the contexts' relevance to languages. It also explains what are User-Interface and Business Process Modeling Languages and how they are used. It presents a definition of Model-Driven Software Development, a study of two community initiatives on this area and a definition of Quality and its importance for this work.

Chapter 3 lists and explains the most relevant academic works of modeling-related topics in the literature. Concretely, it presents frameworks that evaluate related topics or compare and choose a solution after a given problem, namely the quality of conceptual models, the appropriateness of three BPMLs to obtain a standardization of the Business Process modeling and adequate criteria to evaluate the redesign of business processes.

Chapter 4 describes the proposed solution — ARENA Framework — to the previously stated problem. Finally, the collaborative Web System is presented, along with the matrix whose layout, usability and information display served as an inspiration for our project.

Chapter 5 presents an extensive comparison regarding the selected modeling languages and applies the proposed solution, evaluating their quality according to ARENA's properties and metrics. It also details technical information about them.

Finally, chapter 6 closes this work, wrapping up all the given and analysed information. It also presents a discussion of future directions that this thesis can point to.

Chapter 2

Background

This chapter presents important concepts which are critical for the comprehension of this work. It gives an extended explanation of what are modeling languages, a brief description of what is a domain and the relevance of two important actors, domain experts and modeling experts, in the model-driven software development context. It also presents the characteristics of user-interface and business process modelling languages and the wherefores of its use rather than general-purpose languages. Finally, the last section explains the importance of quality on modeling, models and software.

2.1 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 (problem domain) [13]. It may be also described as the vehicle for the expression of the modeling notions that are provided to the designers and it shall have intuitive semantics as well as helpful syntax to convey the intended meaning [2]. It represents knowledge or systems in conceptual models that are defined by a consistent set of rules. Those are used for interpretation of the meaning of concepts. A large number of them appear in academic/scientific literature, concerning modeling restrictions, their structure and the generated models' meaning inside a domain. But there is still much to investigate about restrictions inside some modeling process and what are their relations with the already asked and answered questions about that topic [14]. Also, in 2005, Hoppenbrouwers et al., reported that many of the queries made while actual modeling is running aren't answered unless the produced model is

complete, finished and, most of all, read. According to them, that happens during the process of modeling. Since it is a team effort, conversation and brainstorm sessions are needed about the goals of the project and the means to achieve them. Once this work becomes clear, the team can begin to work focused on basic modeling strategies (paths for advancing on modeling dialogues) that aim to one big objective: answering all the questions that the participants, stakeholders and users might have (the last ones about the finished model) [14].

A modeling language is created from a metamodel (a set of concepts, terms and other things) within a certain domain. Therefore, it can be seen as a model of a modeling language [26, 42]. The process of metamodeling consists on creating a modeling language using its syntax, semantic mapping, semantic schema and notation, along with modeling procedures (steps of the modeling language application and its results — the produced models). In order to give the modeling languages functionality to use and evaluate the models, different types of mechanisms are included in this process (algorithms, generic mechanisms, specific mechanisms and hybrid mechanisms) [19]. 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, BPMN and SysML. Finally, the modeling language is used to support business models and implement technology on them (e.g. Web Service models).

2.1.1 Graphical types

Graphical modeling languages use a diagram technique with named symbols that represent concepts and lines that connect the symbols and represent relationships and various other graphical notation to represent constraints. Example of graphical modeling languages in the field of computer science, project management and systems engineering:

1. Architecture Description Language [17] (ADL) is used to describe and represent the system architecture of a system.

2. Business Process Model and Notation [34] (previously known as Business Process Modeling Notation and abbreviated as BPMN) is a graphical representation for specifying business processes in a business process model. Its Business Process Diagram (BPD), which is based on a flowcharting technique, resembles very much UML's activity diagram. Along with XML, they form Business Process Modeling Language (BPML), in order to provide a

more web-development friendly and popular modeling language.

3. Design & Engineering Methodology for Organizations [7] (DEMO) is an enterprise modeling methodology for transaction modeling and for analysing and representing business processes. It is also a methodology for designing, organizing and linking organizations. The central concept is the "communicative action": communication is considered essential for the functioning of organizations. Agreements between employees, customers and suppliers are indeed created to communicate. The same is true for the acceptance of the results supplied.

4. Integration Definition (IDEF) is a family of modeling languages in the context of systems and software engineering. They cover a wide range of uses, from functional modeling to data, simulation, object-oriented analysis/design and knowledge acquisition. These "definition languages" were developed under funding from U.S. Air Force and although still most commonly used by them, as well as other military and United States Department of Defense (DoD) agencies, they are in the public domain. The IDEF ranges of use include, for instance, IDEF0 for functional modeling, IDEF1X for information modeling, IDEF3 for business process modeling, IDEF4 for Object-Oriented Design and IDEF5 for modeling ontologies.

5. Petri Net [6] is a modeling language to describe distributed systems and mathematical models, invented in 1939 by Carl Adam Petri and still in use. It uses variations on exactly one diagramming technique and topology, namely the bipartite graph. The simplicity of its basic user interface has easily being able to enable extensive tool support over the years, particularly in model checking, graphically oriented simulation and software verification.

6. Service-Oriented Modeling Framework (SOMF) is for designing enterprise and application level architecture models in the space of enterprise architecture, virtualization, service-oriented architecture (SOA), cloud computing, and more.

7. System Modeling Language [33] (SysML) is a general-purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. It is defined as an extension of a subset of the Unified Modeling Language (UML) using UML's profile mechanism.

8. Unified Modeling Language [32] (UML) is another general-purpose modeling language that is an industry standard for specifying software-intensive systems. UML 2.4.1, the current version, supports thirteen different diagram techniques and has widespread tool support,

as we further detail in this section.

2.1.2 Textual types

Textual modeling languages use standardized keywords accompanied by parameters or natural language terms and phrases to make computer-interpretable expressions. Information models can also be expressed in formal languages, namely:

1. Architecture Analysis & Design Language [9] (formerly known as Avionics Architecture Description Language and abbreviated as AADL) supports early and repeated analysis of a system's architecture with respect to performance-critical properties through an expandable notation, a tool framework, and precisely defined semantics. It is both graphical and textual.

2. EFactory is a generic textual Modeling Language for Eclipse Modeling Framework (EMF) models. EFactory is an alternative to the standard tree-based EMF editors. EFactory is a generic EMF editor that provides all the advantages of a textual language. EFactory can be used to instantiate any EMF based model including Ecore itself. Models defined using EFactory integrate seamlessly into existing environments by being compatible on EMF resource level. For example, it is possible to reference an Ecore model that is defined using EFactory from an Ecore model that is defined using a graphical Ecore editor.

3. EXPRESS and EXPRESS-G [15] (its graphic correspondence) are international standard general-purpose data modeling languages for product data.

4. Gellish (originally derived from "Generic Engineering Language") has natural language variants such as Gellish Formal English and Gellish Formal Dutch (Formeel Nederlands), etc. Gellish Formal English is an information representation language or semantic modeling language that is defined in the Gellish English Dictionary-Taxonomy, which has the form of a Taxonomy-Ontology (similarly for Dutch). Gellish Formal English is not only suitable to express knowledge, requirements and dictionaries, taxonomies and ontologies, but also information about individual things. All that information is expressed in one language and therefore it can all be integrated, independent of the question whether it is stored in central or distributed or in federated databases. Information models in Gellish Formal English consists of collections of Gellish Formal English expressions, that use natural language terms and formalized phrases. For example, a geographic information model might consist of a number of Gellish Formal English expressions, such as:
   - the Eiffel tower <is located in> Paris

- Paris <is classified as a> city

whereas information requirements and knowledge can be expressed for example as follows:

- tower <shall be located in a> geographical area
- city <is a kind of> geographical area

As we have seen, there are modeling languages that have both representations, e.g. AADL and EXPRESS [9, 15]. Several modeling languages are executable with proper tool support (i.e., they are able to run and execute the model), and for those that are (e.g. UML and BPMN), their use doesn't necessarily mean that programmers are no longer required. On the contrary, executable modeling languages are intended to amplify the productivity of skilled programmers, so that they can address more challenging problems, such as parallel computing and distributed systems.

One of the biggest problems in developing an appropriate conceptual model for a specific domain is that of testing it for validity and completeness [2, 14, 49]. The first one can be supported by the reasoning and explanation facilities provided by Description Logics. The second shall be more difficult to achieve, as it depends very much on the domain, ontologies, knowledge management and the selected modeling language. The proposed framework intends to bridge that gap.

Each ML has an abstract and a concrete syntax. The former one is the result of a joint effort from the ML's architect and the domain expert, to abstract, conceptualize and synthesize the domain knowledge, which is composed by a metamodel with all the domain's concepts. The latter is related to the interaction that users have with the ML's notation, in terms of understanding, reading and learning from it, specially if they find it useful and easy to work with [42].

A modeling language might be classified as general-purpose (GPML) or domain-specific modeling language (DSML) [20, 25, 29, 53]. 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 [53]. 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 [29].
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 [32, 34], DEMO is specific to enterprise architecture [7], XIS to Interactive Applications [43], and Petri Nets to Distributed Systems [6]. Since there are more languages and approaches than domains, this results in overlapping effort for researchers and disorientation for modelers.

The Modeling Language concept 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.

2.2 Domains

A Domain (also referred as Problem Domain) is an engineering term referring to all information that defines the problem and constrains the solution (the constraints being part of the problem). It includes the goals that the problem owner wishes to achieve, the context within which the problem exists and all the rules that define essential functions or other aspects of any solution product. It represents the environment in which a solution must operate, as well as the problem itself[1]. In other words, it is any subset of a conception (being a set of elements) of the concrete or abstract universes, that is created as being some part or aspect of those universes [14].

Also, it is the area of expertise or application that needs to be examined to solve a problem. A problem domain simply looks at only the relevant topics and excludes everything else.
Some examples of domains are [52]:

- Multimedia (includes Web Computing, Image manipulation, 3D Animation and Drawing);

[1] "Problem Domain", by Cunningham & Cunningham, Inc. (http://c2.com/cgi/wiki?ProblemDomain)

- Telecommunications (namely String and Tree languages for model checking, Communication Protocols, Telecommunication Switches and Signature Computing);
- Software Engineering (for instance: Financial products, Behaviour control and coordination, Software Architectures and Databases);
- Systems Software (mainly Description and Analysis of abstract syntax trees, Video device driver specifications, Cache coherence protocols, Data Structures and Operating System specialization); and
- Miscellaneous (for example Simulation, Mobile Agents, Robot Control, Solving partial Differential Equations and Digital Hardware Design).

This concept is important because the domain or the nature/context of the problem may influence the Modeling Language's appropriateness for it.

2.3 User-Interface Modeling Languages

UIMLs are DSMLs that are specifically used for modeling the user interface of desktop, web or mobile applications, supporting the design and implementation phases [1]. They denote conceptual abstractions that are present on user interfaces [10]. This paper analyses and compares the main properties of four UIMLs, namely: UMLi [45], UsiXML [23], XIS [44] [27] [43] and XIS-Mobile [40] [39]. 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 [50], IFML [35], MARIA XML [36] or WebML [5]. 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.

The Unified Modeling Language for Interactive Applications, abbreviated as UMLi, is a conservative extension to UML focused on modeling user interfaces [45]. It intends to bridge UML's natural gap on web application interfaces, due to its general-purpose nature and the modeler's difficulty of designing user interfaces and domain objects simultaneously [46]. UMLi uses an MB-UIDE approach, which provides the ability to model and implement user interfaces in a systematic way [45]. Tasks are modelled using extended activity diagrams that include six constructors, which are put inside "containers", so the modeler has a better preview of the interface [47].

USIXML is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces (AUIs) and Multimodal User Interfaces (MUIs). It allows to design interaction supporting the Îij7 concept: multi-device, multi-user, multi-culturality/linguality, multi-organization, multi-context, multi-modality, and multi-platform [23].

The eXtreme modeling Interactive Systems (XIS) is a project that started in 2003, intended to implement Model-Driven Architecture best practices on Modeling Languages development [44]. In 2007, ProjectIT was presented as a research project that provides a software development workbench with support for project management, requirements engineering, and analysis, design and code generation activities and encompasses now XIS2, as a UML profile [43]. XIS UML profile is a modeling language based on UML extensions that creates the possibility to design interactive systems in a visual and high-level modeling way.

XIS-Mobile is defined as a language based on XIS, since it is likewise Model-Driven Development-oriented, supported by a UML Profile and suited for modelling platform-independent applications, but is focused on mobile environment, instead of desktop. It was designed to mitigate problems related to app development, such as specificity of each platform (leads to incompatibility), specific development tools (which causes software development complexity), application markets and mobile OS platform fragmentation [38].

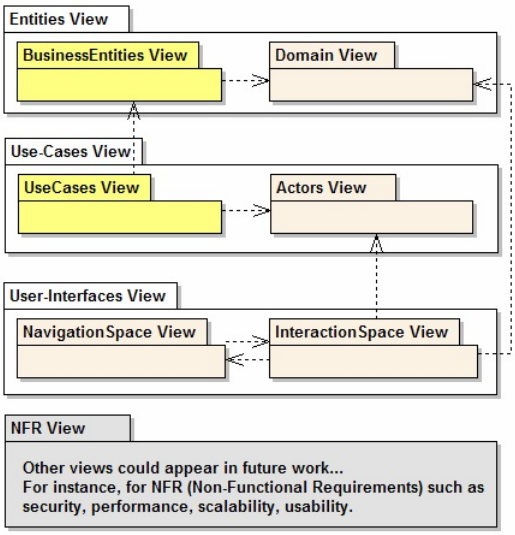

Figure 2.1: The multi-view organization of XIS [43]

2.4 Business Process Modeling Languages

BPMLs are DSMLs that contain concepts and graphical elements which allow the user to design business process. Normally, they are Extensible Markup Language (XML)-based metalanguages, as a means of modeling business processes, much as XML is, itself, a metalanguage with the ability to model enterprise data. Some of their features are being able to list easily distinguishable concepts, include control, structure and cancellation patterns and hierarchical models [31].

The Unified Modeling Language (UML) is a graphical language for specifying, constructing, and documenting the artefacts of systems. It is a general-purpose modeling language that can be used with all major object and component methods, and that can be applied to all application domains (e.g., health, finance, telecom, aerospace) and implementation platforms (e.g., J2EE, .NET) [32]. UML is a language with a very broad scope that covers a large and diverse set of application domains. Not all of its modeling capabilities are necessarily useful in all domains or applications. This suggests that the language should be structured modularly, with the ability to select only those parts of the language that are of direct interest. On the other hand, an excess of this type of flexibility increases the probability that two different UML tools will be supporting different subsets of the language, leading to interchange problems between them. Consequently, the definition of compliance for UML requires a balance to be drawn between modularity and ease of interchange.

The Business Process Model and Notation (BPMN) is another graphical language, which main goal is to provide a notation that is easily understandable by all stakeholders, whom include business analysts that create the initial drafts of the processes, technical developers that are responsible for implementing the technology that will perform those processes, and finally, business people who will manage and monitor those processes. Another goal, also with great importance, is to ensure that XML languages built for the execution of business processes, such as WSBPEL (Web Services Business Process Execution Language), can be visualized with a business-oriented notation. Object Management Group has created this language to perform a set of best practices within the business modeling community to define the notation and semantics of Collaboration diagrams, Process diagrams and Choreography diagrams [34].

DEMO (Design and Engineering Methodology for Organisations) is the leading methodology in the new discipline of Enterprise Engineering (EE). The theory of DEMO is that this social interaction takes place in universal patterns, called transactions. Business processes become clear tree structures of transactions, instead of mind-bending railroad yards. ICT (Information Communication and Technologies) applications support people, they do not take over responsibility. The essence of every organisation is that it consists of a network of transactions and actors (employees with authority and responsibility), completely independent of any implementation. This essence is captured in four integrated models: the Construction Model (actors and transactions), the Process Model (business events and business processes), the Fact Model (business objects and business facts) and the Action Model (business rules and work instructions). Because these models are formalised, ICT applications can directly be generated from them, and the behavior of organisations can be studied through simulation [7] [8].

2.5 Model-Driven Software Development

Model-Driven Software Development (often abbreviated to MDSD or MDD) consists in a software development process of designing models that represent how the system does what it is supposed to do. It is mainly focused on analysing requirements and implementing model-to-model and model-to-code transformations in order to improve software productivity, process quality and the final output [42]. It is used as well for software testing, business process development, software architectures and enterprise architectures. The produced models use elements, such as notation, syntax and semantics to represent the concepts included on a Modeling Language, as defined on the subsection 2.1. MDSD can be seen as a possible concretization of Model-Driven Engineering, since it defends that domain models must be created and explored and the code

shall be automatically generated from and synchronized with it. In this work, we will focus both on Domain-Specific and General-Purpose Modeling Languages. The models and their views shall give a good technical specification, so the developers can transform the represented system into testable prototypes and lastly, software with good user experience.

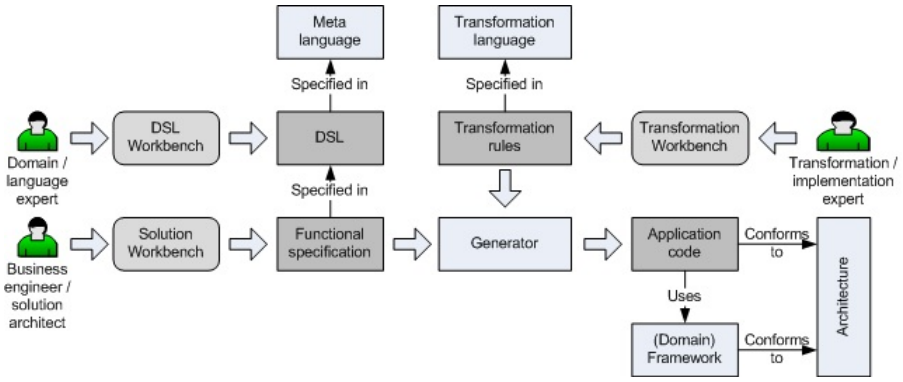

Figure 2.2: MDSD overview (extracted from http://www.theenterprisearchitect.eu/)

As we can see in Figure 2.2, modeling is a learning process in which cooperative actors construct together their perception of reality [14]. This approach is more efficient because it takes advantage of a professional, qualified and trustworthy team that uses communication and tool expertise to produce quality models that satisfy the requirements and may generate quality software. Also, since it is generated automatically, many syntax or misspelling errors are avoided, compared to traditional programming, where humans tend to forget some characters or switch some words, among other common mistakes. Still, MDSD has some downsides, two of them being the detail capacity and the quality assurance [37]. Today it isn't yet possible to specify everything with diagrams as it is with code lines.

2.6 Quality

Quality can be a concept with a subjective description, as it is intimately related to completeness, satisfaction, user comprehension and user experience and it is the goal of many approaches in different areas within the information systems field [21]. We like to see it as a mean to measure the added value and usefulness of any product, comparing the expected requirements to the inherent characteristics.

Quality is essential because it allows evaluation about functionality and requirements matching of software. There are many proposals to classify them into a more detailed definition, but they can be divided into three groups: Normal, Expected and Exciting [21]. The Normal requirements are functions that stakeholders talk about informally, which cover basic aspects of the application. The Expected are aspects that users suppose the programmers already know, so their absence may result in user disappointment. Finally, the Exciting ones are surprise features that users are not counting on. This may include new ways of working with a functionality or an innovative approach to take care of a process.

Despite the modeling's goal to provide tools for the user be able to create quality models, he can't do that without an appropriate modeling language for the related domain and the project's context. Therefore, choosing a good language can help the user to get a good model, along with his experience with a compatible tool and the language itself. Krogstie also considered evaluating modeling language's quality as an important part of the software production process and defined six categories to concretize that theory: Domain appropriateness, Comprehensibility appropriateness, Participant appropriateness, Modeller appropriateness, Tool appropriateness and Organisational appropriateness [21]. On the same work, he evaluated UML 2.0 and concluded that: UML has a very good support for modelling according to an object-oriented perspective; it can be argued to be overly complex, with 233 different concepts (which may result in redundancy); and it is described through a meta-model made in the structural model of UML with accompanying OCL-rules and natural language descriptions of the semantics.

All in all, the use of an appropriate modeling language, in the modeling process, is an important mean to achieve model quality [21].

According to ISO and IEC, regarding a model that may generate software, its quality can be divided into internal and external.
Internal quality consists on the totality of characteristics of the software product from an internal view. It is measured and evaluated against the internal quality requirements. These are used to specify the level of required quality from the internal view of the product. Internal quality requirements are used to specify properties of interim products. These can include static and dynamic models, other documents and source code. Internal quality requirements can be used as targets for validation at various stages of development. They can also be used for defining strategies of development and criteria for evaluation and verification during development [16]. Specific internal quality requirements should be specified quantitatively using internal metrics. Details of

software product quality can be improved during code implementation, reviewing and testing, but the fundamental nature of the software product quality represented by internal quality remains unchanged unless redesigned. Also, in the conceptual modeling domain, this property is called verifiability, which means that there are objective judgements to check if the statements are logical or not [14]. For formal models, the actor that is focused on this verification is the System Analyst. On the other hand, the Domain Expert assures quality for informal models.
External quality is defined by the totality of characteristics of the software product from an external view. It is the quality when the software is executed, 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. They include requirements derived from user quality needs, including quality in use requirements. Also, they are used as the target for validation at various stages of development. External quality requirements for all the quality characteristics defined in quality models should be stated in the quality requirements specification using external metrics, should be transformed into internal quality requirements and should be used as criteria when a product is evaluated [16]. The System Analyst is responsible to assure this property, also known as validity, comparing the derived model with the real world situation for several cases and if it is valid for all of them, the model is considered complete [14].

Quality is not yet a priority factor at the time to develop a Domain-Specific Language (DSL). The focus is currently on getting up to get 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 [48].

2.6.1 Agile Modeling

Agile Modeling (AM) is a practice-based methodology for effective modeling and documentation of software-based systems[1]. At a high level, AM is a collection of best practices, depicted in the pattern language on Figure 2.3 below. At a more detailed level, AM is a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light-weight manner. Scott Ambler + Associates have their own version

[1] Agile Modeling official webpage (http://www.agilemodeling.com/)

of MDSD, called Agile Model Driven Development (AMDD) and they believe that through Test-Driven Design, refactoring and several instances of reviews, the models will improve on quality.

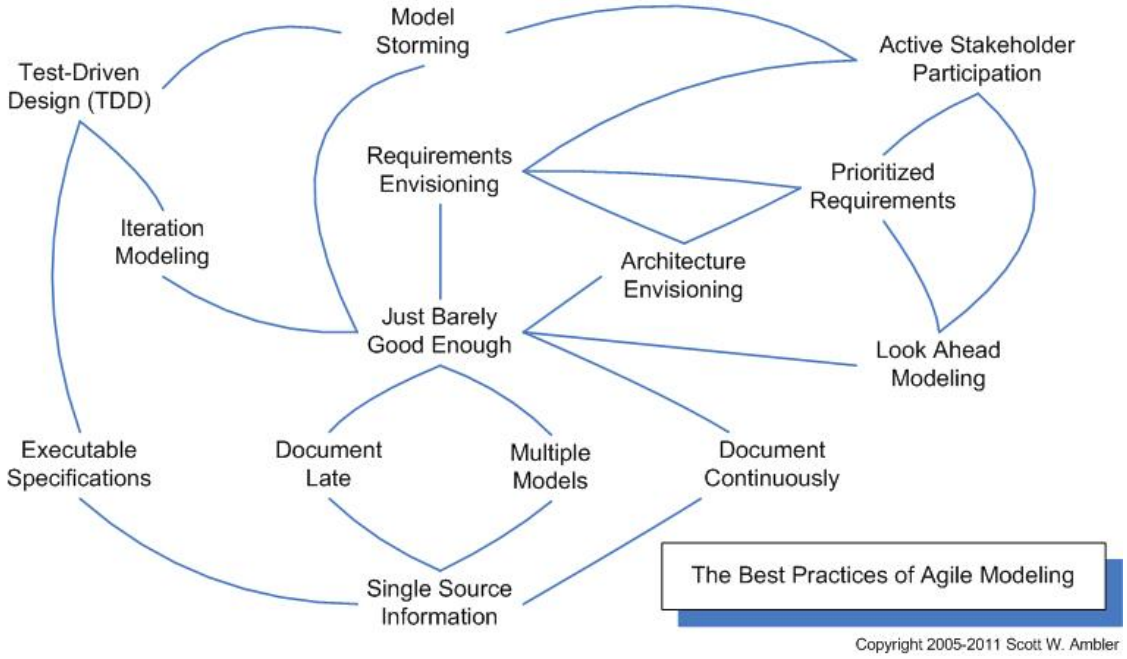

Figure 2.3: Best Practices of Agile Modeling (extracted from http://www.agilemodeling.com/)

2.6.2 ISO/IEC Standards

Since the software is generated from the model (according to the MDSD process, which we described in subsection 2.5) and there is no ISO standard about Domain-Specific Modeling Languages' quality, we believe that both the original languages as well as the generated software should share some quality factors that are presented on the ISO/IEC standards 9126 and 25010 (respectively about Software Product Quality and its Requirements and Evaluation).

ISO/IEC standard 9126 was originally created in 1991, with the purpose of providing a framework that would present the evaluation of software and development process qualities. It defines six characteristics that describe software quality, although no sub-characteristics or metrics were specified: Functionality (the set of functions that the software supports), Reliability (the software's reliability degree), Usability (the experience and easiness that the user gets after trying the software), Efficiency (the software's performance degree), Maintainability (the software's flexibility to be modified) and Portability (the software's compatibility with another machines and architectures) [16]. It was eventually replaced and renamed in 2001 as 9126-1:2001 (the first of four

related documents), since the needs to define a quality model and quality requirements were identified. This new version preserved the characteristics of its predecessor and added: the introduction of normative sub-characteristics, most of which are based on the informative sub-characteristics in ISO/IEC 9126; the specification of a quality model; and the introduction of quality in use. It also removed the evaluation process (which is now specified in the ISO/IEC 14598 standards) and made an effort to coordinate its content with ISO/IEC 14598-1 [16]. ISO/IECs 9126-2:2003 and 9126-3:2003 are respectively about External and Internal Metrics and ISO/IEC 9126-4:2001 refers to Quality in Use.

ISO/IEC standard 25010 started as a reformulated and renamed version of ISO 9126 in 2008, but eventually evolved in 2011 into specifying the requirements and evaluation of a quality model. It presented new and modified sub-characteristics, Security gained more influence, Compatibility, Safety and Transferability were created, but more important than that: Flexibility and Usability in Use were created, due to the usage of the software as a part of a computer system and how they adapt to computer system behaviour [18].

2.7 Summary

This chapter presented the most important concepts and related work which are critical for understanding this work. It gave an extensive description of modeling languages, including examples of graphical and textual types. It also presented the distinction between General-Purpose and Domain-Specific modeling languages, as well as a list of (problem) domains such as Multimedia or Software Engineering. Afterwards, four User-Interface and three Business Process modeling languages were presented, which we selected to evaluate with our framework. Finally, we showed a brief description about quality for modeling languages and tools.

Chapter 3

Related Work

This chapter intends to explain the ideal structure for ARENA, as well as a revision of the existing literature that we consider to be relevant for the conception of the proposed solution. Being a hot topic in the late 90's, the modeling languages have taken a lot of interest in the academic community, specially due to its diversification and potential. So this chapter talks about two community initiatives that are also focused on taking the best advantages of modeling, CMA and ReMoDD. It also shows a project that was inspired by modeling theorists such as the model-driven software development as a recent and different software development approach. Finally, we will expose some evaluation frameworks that were created and used in similar projects and tell the reasons that make them insufficient to apply adequately to this challenge, on the last section, which wraps up this chapter, showing also how all of these frameworks are related with the project's purpose.

3.1 Community Initiatives

In this section, we are presenting two community initiatives that aim to the core of what Model-Driven Development defends: the model as the main focus of attention and as a way to produce software and documentation.

3.1.1 CMA MODELS

The "Comparing Modeling Approaches" (also known as CMA MODELS) is an international annual event, started in 2011, that is comprised of presentations and discussion on modeling approaches and styles about the best solution to a case study which was presented at AOM Bellairs Workshop, which occurred on the same year [4].

The case study presents a Car Crash Management System and offers the choice of modeling either a single system or a software product line. The case study also includes a comparison criteria that is intended to help understanding, analysing and comparing the presented approaches. They are divided into Modeling Dimensions and Key Modeling Concepts. The first group is characterized to evaluate the approach by development phases, activities in which it is useful and languages and notations used (e.g. documents used for the assessment, which problems does the approach address, semantics etc.). The second group aims to classify the approach in terms of building blocks and attributes and identify qualities or improvements brought by the approach (e.g. Modularity, Traceability, Reduction of Modeling Effort etc.). The latter can be applied also to the models produced by the approach, although the focus is the approach itself [4].

The first edition (in 2011) had the main goal of unify existing Aspect-Oriented Modeling (AOM) and more traditional Object-Oriented Modeling (OOM) approaches and to generalize individual approaches into a comprehensive end-to-end method. Several different answers were submitted and then published on ReMoDD (a web supporting tool that we'll describe with more detail in the next subsection) [12].

The second edition (in 2012) presented a selection of the most accurate submitted solutions of the previous year. Their developers would have to target particular development phases (i.e., requirements specification and analysis, high-level and architectural design, low-level design, evolution, run-time, and validation/verification at any of these phases). They have also been required to show comparison criteria categorizations prior to the workshop meeting date [11].

The third and last edition happened on October 2013 and comprised of two main objectives: to continue applying the comparison criteria to other modeling approaches; and to propose and execute analyses of the existing results, thus enabling practitioners to propose and evaluate end-to-end methodologies. This time, three different types of papers will be accepted: Models of the bCMS case study including assessments; Papers reflecting on, improving, or extending the

current comparison criteria document or assessments; and Papers proposing analyses based on the assessments of the currently covered modeling approaches and the comparison criteria document. It also presented GEMOC, a workshop that intended to bring together researchers and practitioners in the modeling languages community to discuss the challenges associated with integrating multiple, heterogeneous modeling languages. MODELS continued to have workshops and keynote speakers, but this case study was closed.

The initiative contributed to this work because we got to know some notions about which criteria matters when comparing different approaches.

3.1.2 ReMoDD

The Repository for Model-Driven Development (ReMoDD) is a resource that aims to support the work of researchers and educators in the Model-Driven Development (MDD) community. Researchers and practitioners can use the repository as a vehicle for sharing exemplar models, illustrative descriptions of modeling methodologies and techniques, detailed modeling case studies, modeling success stories and other forms of modeling experience and knowledge. Resources shared within ReMoDD can be used to gain significant insights into the use of models across the software lifecycle, as a source of data for MDD experiments, as a source of models for testing MDD tools, and to better understand relationships among ongoing MDD research projects. In particular, educators and trainers can use ReMoDD resources to illustrate modeling concepts and approaches in the classroom.

The development of the ReMoDD infrastructure is led by researchers in Colorado State University's Computer Science Department and Michigan State University's Computer Science Department[1]. ReMoDD has been supporting several modeling or MDD-based projects, such as:

- **Modeling in Software Engineering (MiSE) @ International Conference on Software Engineering (ICSE)**, a workshop series that aims to promote the exchange of innovative ideas on the use of models in software engineering and to promote cross-fertilization between the Model-Driven Development communities and software engineering communities;

[1] Repository for Model Driven Development (ReMoDD) Overview (http://www.cs.colostate.edu/remodd/v1/)

- **Models@run.time**, a group that discusses, presents and publishes papers about the problematic of complex distributed software-based systems that operate in highly dynamic environments. Our society is becoming too much dependable on these systems that can provide us disaster response systems, flood prediction and other crisis prevention scenarios. Research in the Models@run.time community focuses on how software models can be used to manage the complexity of adapting behavior at runtime;
- **CMA MODELS 2011 and 2012**, the first and second editions of the event described in the previous subsection;
- **The Bellairs Crisis Management System (bCMS) Group**, which is exploring how aspect-oriented modeling (AOM) techniques can be applied across the software lifecycle and comparing the use of AOM versus non-AOM modeling techniques;
- **Dagstuhl MDD Tool Group**, a seminar on Meta-Modeling Model-Based Engineering Tools that occurred on April 2013.

We believe that this tool will continue supporting the 2015 MODELS' conferences as well as being a useful forum for the academic community debating theories and solutions.

3.2 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.

3.2.1 SEQUAL

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 [22]. 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, being "semantic quality" (Model <—> Domain), "syntatic quality" (Model <—> Language), "pragmatic quality" (Model <—> Audience interpretation) and "appropriateness" on the three remaining relations (Domain

<—> Language, Language <—> Audience interpretation and Domain <—> Audience interpretation). In Figure 3.4, we display its 1995 extension, which brought much more complexity: on one hand, the addition of a new entity (Participant knowledge) and the creation of four relationship types ("physical quality", "perceived semantic quality", "social quality" and "language quality"). On the other hand, Appropriateness disappeared due to its lack of specification. 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. The main changes are the shift of feasible validity and feasible completeness (former semantic quality metrics), feasible comprehension (former pragmatic quality metric) and feasible agreement (former social quality) to deontic quality along with the creation of feasible perceived validity and feasible perceived completeness [21].

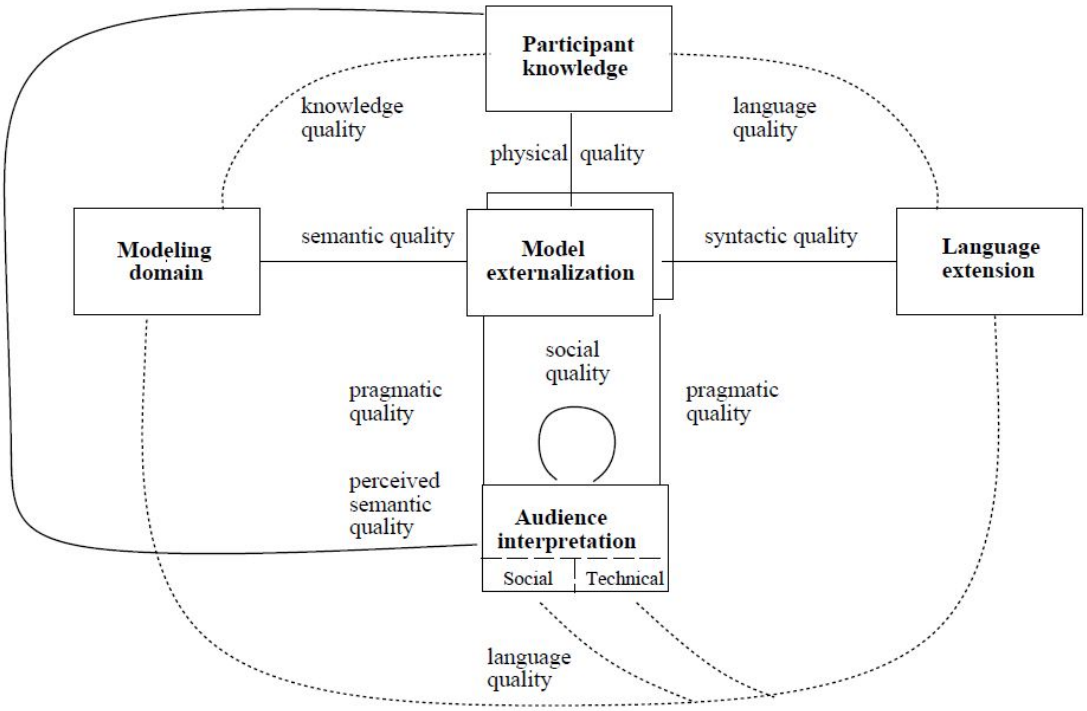

Figure 3.4: SEQUAL framework, extended version (1995) [22]

SEQUAL's formulas are based on the audience, the language, the model, the domain, the audience's knowledge and the audience's interpretation.

The SEQUAL framework contributed to this work because it is the first and main reference regarding frameworks that evaluate quality, in the MDSD context. It is an extensive developed work that also contemplates models' quality and also evaluates UML and BPMN.

3.2.2 BPML Quality Assessment with Generic Framework

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 [31].

This paper contributed to our work because we realised both BP-specific and general purpose properties that were used to compare UML and BPMN.

3.2.3 Testbed project

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, inherence, clarity, consistency, orthogonality and generality [49]. Considering the Testbed evaluation framework, it was applied to behaviour models and it focused on answering three questions: "which aspects of business process can be expressed?", "how easily can they be expressed?" and "when should we use specific models?". We can see in figure 3.5 the materialized framework, that was designed with 4 dimensions: Functionality, Ease of use, Business Process Redesign (BPR) trajectory and General.

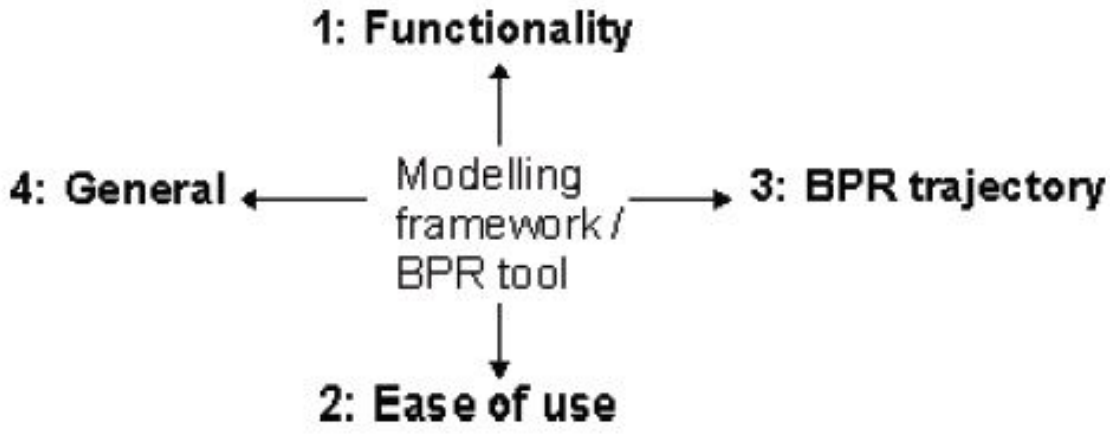

Figure 3.5: The Testbed evaluation framework [49]

These dimensions include criteria such as descriptive power, structuring, formal support (1), accessibility, usability, openness (2), phases of a BPR that are supported (3), price of a certain tool and customer support (4). Then, they apply the framework to a fictitious car insurance company's business process (on that case, a settling insurance claim process), first decomposing the process in activities from an external point of view, then showing the relations between each activity and each item, after which they explain the entities envolved in the process and their relationships and finally they create a diagram showing the behaviour associated to each entity. To assure completeness, they detail one specific activity and they decompose it, again relating it to the items and the entities involved. All the diagrams are consistent and easy to understand.

Although it is focused on models, the Testbed project contributed to this work because we found some of the quality modeling criteria appropriated to our framework. Besides, the example that is shown how their project adds quality for "a number of large, administrative organizations" is, in our opinion, one to take into account when we want to do a demonstration of our modeling-related work.

3.2.4 Comparison

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, as we can see on Table 3.1.

Framework	Context	Reason
SEQUAL	Definition of Systematic Quality Criteria for Conceptual Models	Focused on Conceptual Models, instead of Modeling Languages
BPML Quality Assessment	Evaluation of Quality on three BPMLs	Intends to analyse the Quality of Business Process, which is too specific
Testbed project	Quality Assurance on Redesign of Business Process of Conceptual Models	It is focused in Business Process Redesign, not Modeling Languages

Table 3.1: Summary of the related work frameworks

The selection of the appropriate Modeling Language influences the quality of the final output, whether it is a model or software [18, 22, 37]. Quality integration in software development processes that use models, such as MDSD, 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 modeling 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.

3.3 Summary

In this chapter, we described some work that is related to our investigation. Two community initiatives were presented, namely CMA MODELS and ReMoDD, as well as three frameworks that can complement ARENA, since they are also used for model-driven purposes.

Chapter 4

Proposed Solution

This chapter unwinds our proposal to solve the problem of this thesis: a framework that can show the fitness of a given modeling language. On it, we present the general and specific properties the compose ARENA, giving them definitions and setting weights, quantitative and qualitative measures. Also, we describe the web system, namely its inspiration, present and future functionalities. Ultimately, the last section ends this chapter, showing how our proposed solution intends to gather and accomplish the project's purposes.

4.1 ARENA Framework

Since ARENA is a tool conceived to evaluate languages that allow concept representation, it needs a conceptual model of its own. Figure 4.6 shows that a Modeling Language's Quality is the central class and it contains the final quantitative output. This class uses the Modeling Language's information as an input and returns a Value within a range, as an output.

The latter 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 5.1. 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 being 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, which are described in the following subsections.

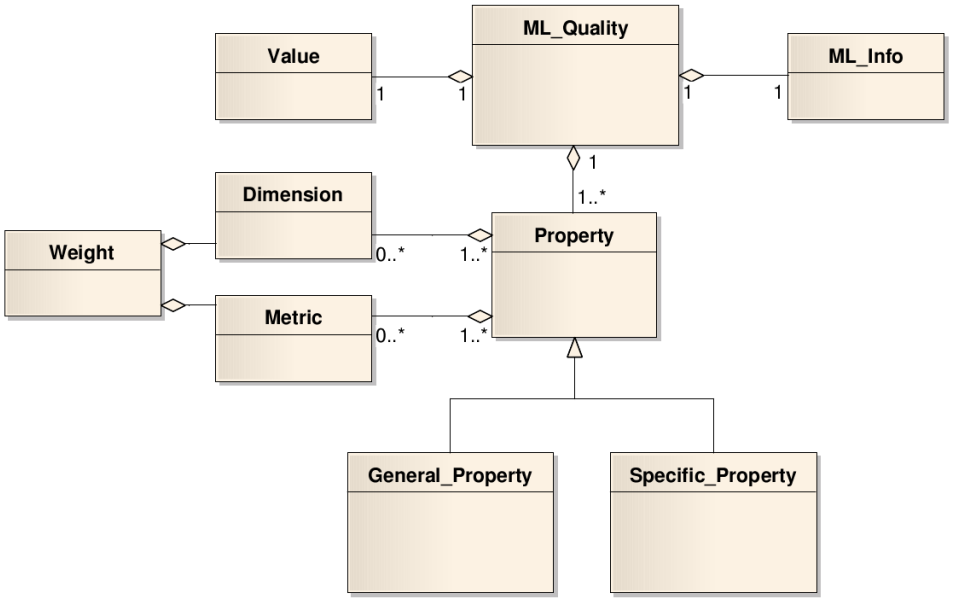

Figure 4.6: ARENA Framework — Main Concepts

4.1.1 General Properties

Comprehension. The language shall be able to precisely specify systems so that the developing team and the stakeholders (e.g. customers, operators, analysts, designers) can share their conceptions of reality and better comprehend the produced models. All team members have their unique perspective and interpretation of the real world and when it comes to achieve a common understanding, communication helps to mediate discussions, develop education and manage knowledge [14]. Every participant must acknowledge his part of the externalized model, as we have seen on subsection 3.2.1. For instance, a designer shall be able to understand the concepts and rules of the language, in order to optimize his working time and effort to raise productivity [49]. Dimensions: Communication and Expressiveness.

Domain. The environment where the language is going to be used to create models can facilitate or constrain its use [31] and all the statements that are possible to make with it shall be correct and relevant to the problem's resolution, within the domain. This will influence in the model's validity [22]. This framework states representing the domain as domain appropriateness, i.e. the ability to describe the domain's statements using the modeling language [22]. It should ideally be able to express things that are only in the domain but be powerful enough to include everything that is in the domain. In the best case scenario, there is no statement that can't be expressed. The conceptual modeling activity must be adapted in the context in which models are going to be

created [14]. Metric: Ratio between Domain's Number of Concepts and Language's Meta-Model Number of Concepts.

Functionality. This property can be formally defined as the particular set of functions or capabilities associated with computer software or hardware or an electronic device, alternatively, the system's behaviour quality [24]. We take into account if the ML has a list of simple and reusable templates that help the user solving common problems that appear during the design phase and 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. On one hand, and according to what we said in Section 2.5 about MDSD, it is crucial to evaluate the language's ability or inability to generate textual artefacts from models and if so, which mechanisms or techniques make it possible. On the other hand, it is important to know which systems, applications or mechanisms can analyse the models and diagrams created by the user and validate them. Metrics and Dimensions: Pattern Usage, Tool Support - Model to Model (M2M) Transformations, Tool Support - Model to Text (M2T) Transformations and Tool Support - Validation.

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 [18], Number of Integration Mechanisms [19] and Compatibility.

Maintenance. The language must be appropriate to the present needs, that is, it should not have obsolete concepts or operations. Its concepts shall also be consistent and non-ambiguous, i.e., each one of them must have only one meaning in the real world, when representing it [49]. Also, if the generated models systematically don't contain semantic quality (completeness and validity), it may imply that the language needs maintenance. Nevertheless, the language must have evolution and be flexible so that it is possible to add further elements or layers. Dimensions: Stability, Changeability, Consistency, Reusability and Extensibility.

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 [2, 14, 49]. 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.

Usability. The language must be easy for the team members to use, so that the connection between these and computing platforms can be established naturally as if it was an interface [3]. It must also specify system requirements, structures and behaviour. The processes shall be easy to model, the language environment should offer pre-defined constructs and libraries of high-level concepts, these should be easy to adapt to individual needs and the modeling methods need to be comprehensible and well documented [49]. Dimensions: Understandability, Learnability, Operability, Attractiveness, User Satisfaction [16] and Adaptability [49].

Other Features. It is a set of MLs' properties that provide additional information about them. They may or may not be unique abilities. They will be considered, but not weighted to calculate the MLs' quality values.

4.1.2 Specific Properties — UIMLs

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

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.

4.1.3 Specific Properties — BPMLs

Executability. A language is executable as long as the user can produce behavioural models with enough fine granularity so they can run as programs. In order to this translation happen effectively, the compiler must be fast and reliable and the generated code must also be fast and robust [41]. There have been model execution tools and environments for years, but the scalability is always a challenge. Each tool defined its own semantics for model execution, often including a proprietary action language, and models developed in one tool could not be interchanged with or interoperate with models developed in another tool, as we can confirm at `modeling-languages.com/new-executable-uml-standards-fuml-and-alf/`.

List of Actions. This property intends to list all the actions included in business processes that are possible to represent with the modeling language.

Modularity. It can be defined as the capacity of construct and re-construct some parts of the models, in a way that is easy for any user to collaborate and accept them. Modules can be

related 1-to-1 to interfaces, so that model's structure becomes more visible [28].

4.2 ARENA Website

Apart from useful, we wanted ARENA to be available to as many modelers and developers as possible. This tool intends to be a support for quality choice destined to academic researchers as well as enterprise professionals, at a world-wide level, so we felt inspired by content management tool CMS Matrix[1]. Therefore, we have developed the ARENA website at `http://arenaframework.comeze.com`, with PHP and minimal CSS. Figure 4.7 shows the framework's homepage, which has a brief description of the tool's goal and context, the list of languages that a user can compare, the possibility of editing a listed language or add a new one.

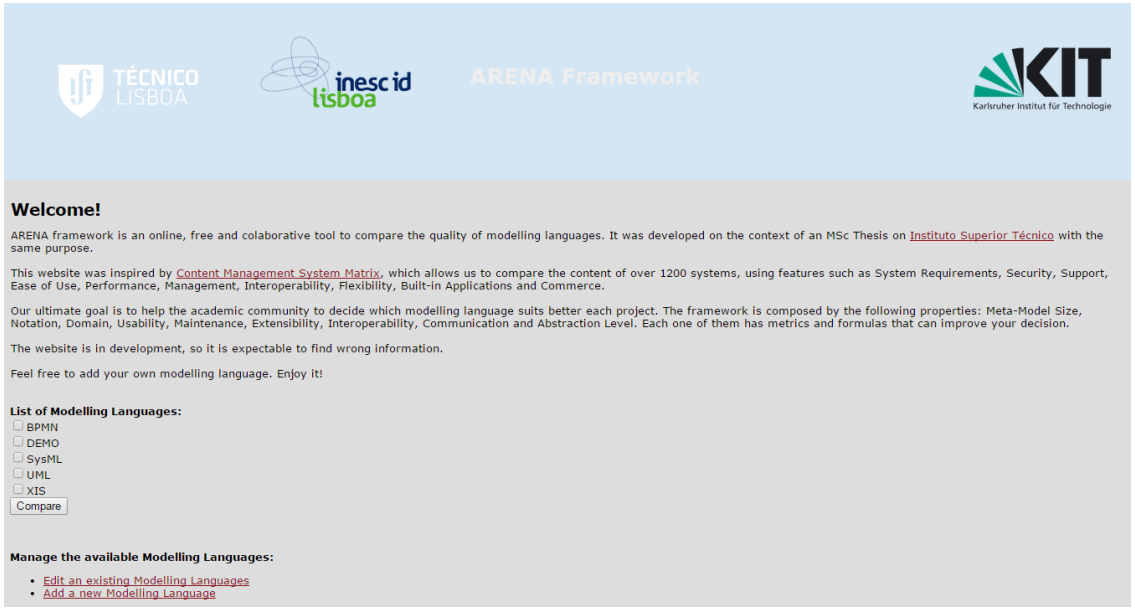

Figure 4.7: ARENA's homepage (extracted from http://arenaframework.comeze.com)

In order to make the comparison, the user simply has to click on the language's checkboxes and then on "Compare". For instance, if UML and XIS are selected, a screen like Figure 4.8 will appear. The selected languages are displayed in columns and ARENA's properties, metrics and dimensions are displayed in lines. It is possible to click on a Property to see its definition on the right side of the panel. If the user wants to add or remove compared languages, he/she just has to check or uncheck the box on the left side of the panel. At least two languages must be selected, otherwise the system will display an error message.

This tool is being updated regarding language's data and some features are being developed.

[1] http://www.cmsmatrix.org

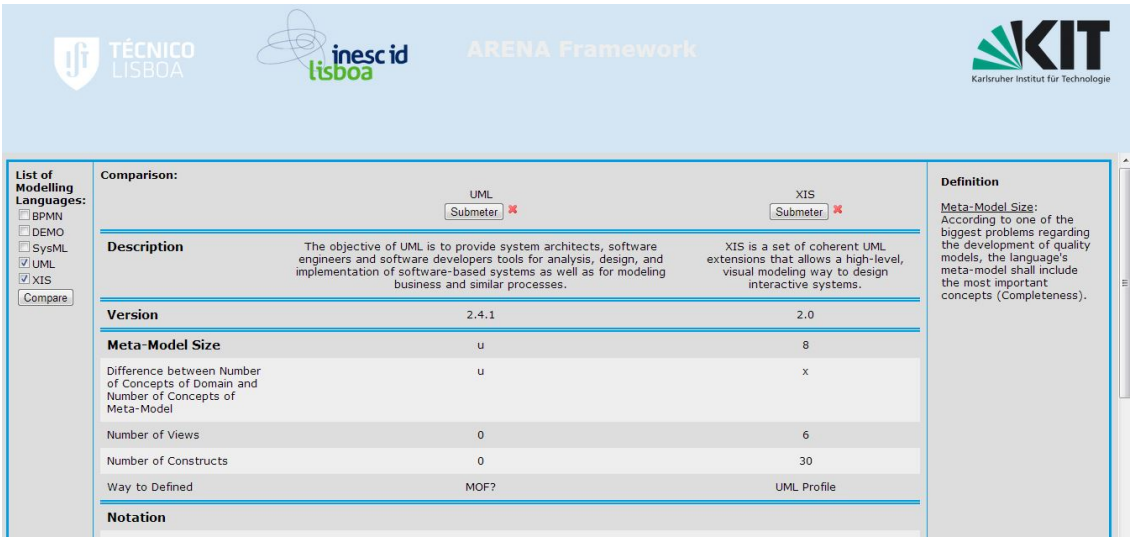

Figure 4.8: Comparing UML and XIS with ARENA (extracted from http://arenaframework.comeze.com)

We want to group languages according to its domain, namely UIMLs, BPMLs or GPMLs and add the possibility of selecting the domain's languages, instead of one by one, at the homepage. Also, regarding countable items, we believe that adding charts will make information analysis smarter, as well as adding links to language's compatible software, papers and projects in which they were used.

4.3 Summary

In this chapter, we described the solution proposal. All of its general properties, as well as specific properties for both domains we selected were listed and described. Also, we have shown how this online tool was developed and how it works, so it can be used by modeling researchers and professionals.

Chapter 5

Validation and Results

This chapter presents a short version about the ARENA framework approach of quality assurance by giving an overview of how the modeling languages were evaluated. It also presents the results obtained for the analysed modeling languages, as well as a brief discussion of those results.

This chapter is organized as follows. Section 5.1 describes how the method of our solution was applied to the selected modeling languages, namely the relevant information to each characteristic and the calculus of each modeling language's appropriateness. Section 5.2 presents a more detailed discussion about UIMLs and BPMLs, as their full characteristics unfold.

5.1 Evaluation

Table 5.2 summarizes the main properties of four analysed UIMLs — UMLi [45], UsiXML [23], XIS [27] [43] and XIS-Mobile [39] [40]. The lines 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 line Compatibility refers to a dimension from ARENA's property Interoperability, as well as metrics Number of Compatible Applications and Number of Integration Mechanisms. ML's Functionality is evaluated using dimensions Pattern Usage, Tool Support - M2M Transformations, Tool Support - M2T Transformations and Tool Support - Validation. The last three lines have exactly the same name and meaning of the specific properties, as have shown on the previous chapter, subsection 4.1.2

Property \ Language	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-modeling	XML Metamodel and Cameleon reference framework	UML Profile	UML Profile	—
Compatibility	"	ArgoUML	Eclipse, Enterprise Architect, Microsoft Visual Studio, Notepad++, Kate, W3schools.com and others	ProjectIT Studio	Enterprise Architect	—
Pattern Usage	General Properties	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 - 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	—
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	—
Number of Compatible Applications	"	1	>6	1	1	2
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
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.	—
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
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

Table 5.2: UIMLs comparison based on the ARENA Framework

In a similar way, Table 5.3 enframes the principal characteristics of the three BPMLs we chose — UML [32], BPMN [34] and DEMO [7]. The only difference regarding Table 5.2 is that for this domain, the specific properties are Executability, List of Actions and Modularity, as we presented on subsection 4.1.3.

The quality value of a language is determined by Formula 5.1, designated as the ARENA General Equation of Quality. The names of the plots to be firstly multiplied and then added, are no more than shortened versions of the properties considered in Tables 5.2 and 5.3.

$$
\begin{aligned}
QualityML_n &= \sum_{i=1}^{j} w_i * r_i \\
&= w\_absStxProps * r\_absStxProps + w\_concStxProps * r\_concStxProps + \\
&+ w\_genProps * r\_genProps \\
&= w\_numViews * r\_numViews + w\_numClassifs * r\_numClassifs + \\
&+ w\_numRelats * r\_numRelats + w\_repType * r\_repType + \\
&+ w\_suppMechs * r\_suppMechs + w\_numPatterns * r\_numPatterns + \\
&+ w\_M2MTrfs * r\_M2MTrfs + w\_M2TTrfs * r\_M2TTrfs + \\
&+ w\_Valid * r\_Valid + w\_numCompApps * r\_numCompApps + \\
&+ w\_numIntMechs * r\_numIntMechs
\end{aligned}
\tag{5.1}
$$

Since we are comparing two different specific domains — User-Interface and Business Process — , we need to create two specific equations based on ARENA General Equation of Quality, as displayed on Formulas 5.2 and 5.3.

$$
\begin{aligned}
QualityUIML_n &= QualityML_n + \sum_{i=1}^{j} wSpecUIProps_i * rSpecUIProps_i \\
&= QualityML_n + w\_numAppActs * r\_numAppActs + \\
&+ w\_numUserInts * r\_numUserInts + w\_numWidgets * r\_numWidgets
\end{aligned}
\tag{5.2}
$$

Language / Property	Property Group	UML (Activity Diagram)	BPMN (Business Process Diagram)	DEMO (Process Model)	Average (BPMLs)
Number of Views	Abstract Syntax	1	1	2	1
Number of Classifiers	"	43	29	6	26
Number of Relationships	"	3	11	1	5
Representation Type	Concrete Syntax	Graphical	Graphical	Graphical	—
Supporting Mechanisms	"	MOF	MOF	eBNF	—
Compatibility	"	ArgoUML, Dia, Eclipse UML2 Tools, Enterprise Architect, IBM Rational Rhapsody, Microsoft Visio, Modelio, NetBeans, Papyrus, StarUML, Umbrello UML Modeller and others	Agiles BPMS & ECM, BiZZdesign Architect, BPMN Visio Modeler, Eclipse BPMN2 Modeler, Enterprise Architect, HP Process Automation, IBM Process Designer, Microsoft Visio 2013 and others	Essential Actions Engineers, Formetis, Modelworld, Mprise Tooling and Open Modeling	—
Pattern Usage	General Properties	50	86	1	46
Tool Support - M2M Transformations	"	Yes. It is possible to transform UML activities, Activity Edge, Call Behaviour Action, Decision Node and other graphical elements into BPMN, using UML 2.0 Diagram Interchange	Yes. BPMN's Diagram (Interchange) Definition provides a basis for modeling and interchanging graphical notations, specifically node and edge style diagrams as found in BPMN, UML and SysML	No. It is only theoretically defined, but it is not implemented	—
Tool Support - M2T Transformations	"	Yes. IBM's Rational Software Architect allows UML-to-Java transformations. Also, it is possible to transform UML models into Communicating Sequential Processes (CSPs), using graphs	Yes. It can be mapped to WS-BPEL and XML and possibly to Finite State Machines	No. Enterprise Engineering Institute doesn't see any added value for DEMO with this feature	—
Tool Support - Validation	"	XML uses XSD to validate instances of XML documents	The "implementation" attribute, which is present in Service Task, Send Task and other models, must have one of these values: "##unspecified" or "##WebService"; Messages exchanged within a Conversation are validated using CorrelationKeys and CorrelationProperties; and others	All compatible tools perform Syntax Analysis on DEMO models and return "warning" or "OK" messages	—
Number of Compatible Applications	"	41	58	5	35
Number of Integration Mechanisms	"	7 (MDA-driven, Exports to XMI, Generates languages, Generates languages using reverse-engineering, Can be integrated with IDEs, Can be integrated with Web Browsers and Can be integrated with Office applications)	2 (Relationship Types and Diagram Interchange package)	4 (DEMOWORLD, J2EE, Meetingworks and Xemod)	4
Other Features	"	(N/A)	XSLT transformations allow inter-changing model formats between XSD and XMI. Also, BPMN has Diagram Interchange package, a set of BPMN meta-classes that allows BPMN models' interoperability between different tools.	It is possible to add comments on models	—
Reusability	"	A fine-grained, flexible metamodel library is provided that is reused to define the UML metamodel, as well as other architecturally related metamodels, such as MOF or CWM	It is possible to copy data between graphical elements, using the same ItemDefinition or a DataAssociation with a transformation Expression	(N/A)	—
Extensibility	"	UML Profiles can be used to customize the language for particular platforms and domains	It has an Extension Class that allows extending standard BPMN elements with additional attributes	There has been an effort in this way, modeling an organization's activities with DEMO and then extending them to allow activities' generation, operation and discontinuation	—
Executability	Specific Properties	There are specific types of components that can be deployed as Executable Artifacts and they can be related to a Node using the DeployedArtifact and DeploymentTarget elements	In order to modeling tools be able to emit executable models, there are some restrictions: Data type definition language MUST be XML Schema, Service Interface definition language MUST be WSDL and Data access language MUST be XPath	(N/A)	—

Table 5.3: BPMLs comparison based on the ARENA Framework

$$
\begin{aligned}
QualityBPML_n &= QualityML_n + \sum_{i=1}^{j} wSpecBPProps_i * rSpecBPProps_i \\
&= QualityML_n + w\_numActions * r\_numActions
\end{aligned}
\tag{5.3}
$$

The notation used on the formulas 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; Compatibility — 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 Compatibility 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 4.1, and the sum of this products will return the language's quality, according to the ARENA General Equation of Quality.

5.2 Discussion

5.2.1 UIMLs

In a related paper [30], we also have compared these four UIMLs. 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 Compatibility. This can be explained by their maturity and popularity. Also, there are patterns concerning number of classifiers and notation, 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 compatibility, 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 Compatibility line, 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 Compatibility 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" [45].

According to Table 5.4, for this evaluation, XIS-Mobile has the highest quality value — 4 — among these four UIMLs, in the considered domain, as we concluded previously [30].

Language / Property	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
Compatibility	0.033	3	4	2	4
Pattern Usage	0.1	1	2	3	5
Tool Support - M2M Transformations	0.1	2	1	2	4
Tool Support - M2T Transformations	0.1	2	1	4	4
Tool Support - Validation	0.1	3	1	3	3
Number of Compatible Applications	0.05	3	3	3	3
Number of Integration Mechanisms	0.05	3	3	3	3
Other Features	—	—	—	—	—
Application Actions	0.1	3	1	3	3
User Interactions	0.1	2	5	2	3
Widget Types	0.1	1	2	5	4
Total Quality	1	2	2	3	4

Table 5.4: UIMLs evaluated with ARENA Framework

5.2.2 BPMLs

In this subsection, we present Table 5.5, in which the considered three BPMLs' most relevant properties are evaluated by our framework. At a glance, it is possible to see that generally, UML and BPMN are more developed than DEMO. This difference is clear regarding the number of classifiers, of relationships, of patterns, of compatible applications and the transformation possibilities.

Since we are considering only diagrams focused on actions and process specification, it is plausible a low Number of Views: both UML and BPMN only have one, the diagram itself, yet DEMO has two (Process Structure Diagram and Transaction Process Diagram). Respecting Number of Classifiers and Number of Relationships, DEMO has a smaller value than the other two languages. Albeit there is a big distance in terms of UML's 43 and BPMN's 29 classifiers to DEMO's 6, it doesn't mean that the latter is incomplete. It may not provide as much flexibility as the other two, but surely it is simpler to understand and has less redundancy probability. The difference between them is reduced when comparing the Number of Relationships, as they all are close to the average value, 5.

Referring now to Representation Type, there is an unanimity. All selected languages are Graphical, which allow process specification and hierarchy to be easier than if only text was used, taking into account the number of concepts and graphical elements that this domain contains. This choice is in line with the MDSD approach and its advantages, as we presented in Section 2.5. Considering Supporting Mechanisms, UML and BPMN have a MOF meta-model and DEMO has an extended BNF grammar. The reason for this difference is that when developing DEMO, Jan Dietz and the EE team have chosen a non-object-oriented approach, because they believe it would be too restrictive, so they preferred to define grammars as they are more open to extensibility and compatible with other languages. Regarding Compatibility, the two OMG's languages share some common IBM, Microsoft and Sparx Systems applications, while the third has Microsoft Windows' compatible tools and web-based tools for Apple and Linux users.

For Tool Support - Model to Model Transformations and Model to Text Transformations, UML and BPMN have a clear advantage over DEMO, due to OMG's development focus on interchangeable diagrams, namely between themselves, as they both have a Diagram Interchange package, which allows node-to-node mapping and their models can be exported to two well-knowed and

wide-used text languages, Java and XML. DEMO has no available transformations, either because it is still planned to be developed or because they do not see advantages on them. As for Tool Support - Validation, the languages have different approaches: both UML and BPMN use XSD to validate XML documents, but BPMN has specific validation mechanisms for some elements' attributes and DEMO only has a syntax analysis algorithm, since its models are very simple, due to the smaller number of graphical elements.

About the Number of Compatible Applications, BPMN leads with an impressive number of 58, followed by UML's 41 and, above the average, comes DEMO with 5. Since DEMO is the most recent one and, therefore, the most unknown, this difference can be explained. However, we believe that when more universities and companies adopt it and see its value on its simplicity, more tools will be developed and more accepted this language will be. The BPMLs are well balanced in terms of Number of Integration Mechanisms, with an average 4, the same as DEMO's, higher than BPMN's 2 and lower than UML's 7.

BPMN is the top language in what comes about Pattern Usage. It has 86, which are divided in Control-Flow (28), Data (18), Resource (8) and Exception Handling (32). DEMO only has 1, the Transaction (connection between two agents, which is composed by 20 steps) and UML has 25 Control-Flow patterns, 17 Data patterns and 8 Resource patterns, making a total of 50. As far as Reusability is concerned, UML contains a library which allows editing its metamodel or others, such as Meta Object Facility (MOF) or Common Warehouse Metamodel (CWM), while BPMN has a more model-focus approach, making it possible to copy data between elements. We can't compare DEMO in this topic, since we could not find related information. In terms of Extensibility, the UML can be extended in two ways: A new dialect of UML can be defined by using Profiles to customize the language for particular platforms and domains, or a new language related to UML can be specified by reusing part of the InfrastructureLibrary package and augmenting with appropriate metaclasses and metarelationships. BPMN includes an Extension Class composed by four elements, that allows extending standard BPMN elements with additional attributes, such as Artifacts. Still, they need to have valid BPMN Core, be semantically compatible with any BPMN element and extended Diagrams should keep the basic look-and-feel to maintain easy understanding, which means that Events, Activities and Gateways must not be altered [34]. EE has been doing an effort in this way with DEMO, taking already developed models or modeling an organization's activities with this language and then extending them to allow activities' generation, operation and discontinuation.

And lastly, about Executability and once again, we don't know how this property can be evaluated in the DEMO case, but we realized that UML and BPMN are executable modeling languages. The first one has the possibility of making some components executable, by defining stereotype

<<executable>> in an Artifact and then relating them to nodes, using deployment linking elements The latter has some restrictions implemented (as listed on the bottom line of Table 5.5) so the executable models can be emitted and its private business process models must have the "isExecutable" boolean set to "TRUE", so they can become executable.

From our point of view, we defend that for the considered domain and the languages selected, BPMN has the highest quality value — 4 — compared to the other two BPMLs.

Language / Property	Weight	UML (Activity Diagram)	BPMN (Business Process Diagram)	DEMO (Process Model)
Number of Views	0.033	4	4	4
Number of Classifiers	0.033	3	4	2
Number of Relationships	0.033	2	4	3
Representation Type	0.033	4	4	4
Supporting Mechanisms	0.033	4	4	4
Compatibility	0.033	4	4	3
Pattern Usage	0.1	4	5	2
Tool Support - M2M Transformations	0.1	4	4	2
Tool Support - M2T Transformations	0.1	4	4	2
Tool Support - Validation	0.1	3	3	3
Number of Compatible Applications	0.05	4	4	3
Number of Integration Mechanisms	0.05	3	3	3
Other Features	—	—	—	—
Reusability	0.1	3	4	1
Extensibility	0.1	3	4	1
Executability	0.1	4	3	3
Total Quality	1	3	4	2

Table 5.5: BPMLs evaluated with ARENA Framework

5.3 Summary

This chapter presented a comparison and evaluation of the results obtained in the analysis of seven modeling languages using ARENA framework's general and specific properties, applied to their characteristics. We have concluded that XIS-Mobile has the highest quality value for UI domain and BPMN for BP domain.

Chapter 6

Conclusions and Future Work

This is the final chapter of this work. On it, the final remarks of this work are presented. Section 6.1 exposes the conclusions of this thesis by making an overview of this work, the MDSD approach and modelling. Lastly, Section 6.2 presents some directions of future work that this thesis can point to, in order to improve ARENA and strengthen other MDE projects and ideas.

6.1 Conclusions

This work 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 and three BPMLs using ARENA, a framework oriented to that solution. In this case, ARENA's most adequate dimensions and metrics were used, along with User-Interface and Business Process specific properties, defined for this purpose and context. With this analysis, it is possible to list the features in which each UIML and BPML stands out from the others, either because they are better or they are unique. Clearly there is a trend for using graphical notations. This can be due to guidelines that compose the well known Model-Driven Engineering. An advantage, is also to use these graphical models to implement the alternative approach to produce software: Model-Driven Software Development.

With this report, one can conclude that MDSD seems to be the optimal approach to future code generation but there is still a lot of work to do. MDSD has some issues to deal with, such as: The rigidity of modeling (i.e., there's no possibility of changing every detail); The flexibility depends

on the selected tool and DSL; The roles of project members are rather different (e.g., the person who builds the solution is the Business Engineer, and not the Programmer); The modeling environment doesn't always support version control; The team normally lose their focus, experimenting new features instead of working on the projects' objectives and more[1]. Although there are many Domains to even more Modeling Languages, quality is a factor that is important during the development of prototypes, but isn't still well considered during the selection process. This report will help Software Language Engineers to have a easier modeling language and model *brainstorm*, in order to help MDSD becoming the next big revolution on 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 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.

6.2 Future Work

In this section, some directions are presented of future work that this thesis can point to.

Analyze other specific Domains (for instance Software Process Modeling Languages) and other DSMLs, such as Petri Nets (for Mathematical Models) or CML-ML (for CMS-based applications), to provide more credibility and variety to ARENA framework, similarly to the CMS Matrix project, available at `http://www.cmsmatrix.org/`.

Promote and search workshops related to these and other UIMLs and BPMLs, in order to get a more objective and solid opinion on them, both with people who have worked with MLs as with others that have not worked. XIS-Mobile's workshop was profitable and if there were more, more easily properties and metrics would be defined and supported. The questions of the related feedback inquiries should be focused on the properties, metrics and dimensions of ARENA.

Apply ARENA to more GPMLs, for instance SysML, to understand if it is a suitable framework,

[1] "8 reasons why Model-Driven Development is dangerous", by Johan den Haan (http://www.theenterprisearchitect.eu/archive/2009/06/25/8-reasons-why-model-driven-development-is-dangerous)

namely if it has a good balance between generic and specific properties, adding, editing or removing properties and metrics, or if GPMLs must have their own framework.

Study the properties and metrics' adequacy, according to these new analysis by introducing or removing them and changing weights in the formula that gives the Quality percentage.

Bibliography

[1] ACHILLEOS, A., YANG, K., GEORGALAS, N., AND AZMOODECH, M. Pervasive service creation using a model driven petri net based approach. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International* (Aug 2008), pp. 309–314.

[2] BAADER, F., CALVANESE, D., MCGUINESS, D., NARDI, D., AND PATEL-SCHNEIDER, P. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[3] BARIŠIĆ, A., AMARAL, V., GOULÃO, M., AND BARROCA, B. *Evaluating the Usability of Domain-Specific Languages*. IGI Global, 2012, ch. 14, pp. 386–407.

[4] CAPOZUCCA, A., CHENG, B. H., GEORG, G., GUELFI, N., ISTOAN, P., AND MUSSBACHER, G. Requirements definition document for a software product line of car crash management systems. *Comparing Modeling Approaches Workshop MODELS 2012* (2012).

[5] CERI, S., FRATERNALI, P., BONGIO, A., BRAMBILLA, M., COMAI, S., AND MATERA, M. *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[6] DESEL, J., AND JUHÁS, G. *What Is a Petri Net?*, vol. 2128 of *LNCS*. Springer-Verlag, 2001.

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

[8] DIETZ, J. *Enterprise Ontology - Theory and Methodology*. Springer, 2006.

[9] FELLER, P., GLUCH, D., AND HUDAK, J. The architecture analysis & design language (aadl): An introduction. Tech. rep., Carnegie Mellon University, 2006.

[10] FRANK, M. R. *Model-based User Interface Design by Demonstration and by Interview*. College of Computing, Georgia Institute of Technology 1996. Directed by James Foley., 1996.

[11] GEORG, G., ALI, S., CHENG, B., COMBEMALE, B., FRANCE, R., KIENZLE, J., KLEIN, J., LAHIRE, P., LUCKEY, M., MOREIRA, A., AND MUSSBACHER, G. Modeling approach comparison criteria for the cma workshop at models 2012. *Colorado State University MODELS 2012* (2012).

[12] GEORG, G., MUSSBACHER, G., CHENG, B., MOREIRA, A., AND FRANCE, R. Modeling approach comparison criteria for models 2011 cma workshop. Tech. rep., Colorado State University, 2011.

[13] HE, X., MA, Z., SHAO, W., AND LI, G. A Metamodel for the Notation of Graphical Modeling Languages. In *Proceedings of the 31st Annual International Computer Software and Applications Conference - Volume 01* (2007), vol. 1 of *COMPSAC '07*, IEEE Computer Society, pp. 219–224.

[14] HOPPENBROUWERS, S., PROPER, E., AND VAN DER WEIDE, T. P. A Fundamental View on the Process of Conceptual Modeling. In *Conceptual Modeling - ER 2005* (2005), vol. 3716, Springer-Verlag, pp. 128–143.

[15] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO 10303-216:2003 Industrial automation systems and integration - Product data representation and exchange, 2003.

[16] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, AND INTERNATIONAL ELECTROTECHNICAL COMMISSION. ISO/IEC 9126-1:2001(E) Quality Model, 2001.

[17] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, AND INTERNATIONAL ELECTROTECHNICAL COMMISSION. ISO/IEC/IEEE 42010 Systems and Software Engineering - Architecture Description, 2001.

[18] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, AND INTERNATIONAL ELECTROTECHNICAL COMMISSION. ISO/IEC CD 25010.2 Software and Quality in use models, 2008.

[19] KARAGIANNIS, D., AND KÜHN, H. Metamodelling Platforms. In *Proceedings of the Third International Conference EC-Web 2002 - Dexa 2002* (2002), vol. 2455, Springer-Verlag, pp. 182–195.

[20] KOSAR, T., OLIVEIRA, N., MERNIK, M., PEREIRA, M. J. V., ČREPINŠEK, M., DA CRUZ, D., AND HENRIQUES, P. R. Comparing general-purpose and domain-specific languages: An empirical study. In *Computer Science and Information Systems* (2010), vol. 7, University of Novi Sad, Serbia.

[21] KROGSTIE, J. *Model-Based Development and Evolution of Information Systems: A Quality Approach*. Springer-Verlag London, 2012, ch. 5, pp. 249–280.

[22] KROGSTIE, J., LINDLAND, O. I., AND SINDRE, G. Defining quality aspects for conceptual models. *Faculty of Electrical Engineering and Computer Science - The Norwegian Institute of Technology* (1995).

[23] LIMBOURG, Q., VANDERDONCKT, J., MICHOTTE, B., BOUILLON, L., FLORINS, M., AND TREVISAN, D. USIXML: A User Interface Description Language for Context-Sensitive User Interfaces.

[24] LÓPEZ, C., CUADRADO, J.-J., AND SÁNCHEZ-ALONSO, S. Conceptualizing measures of required software functionality. In *Proceedings of ONTOSE 2007-Second International Workshop on Ontology: Conceptualizations and Epistemology for Software and Systems Engineering* (2007).

[25] LUOMA, J., KELLY, S., AND TOLVANEN, J.-P. Defining domain-specific modeling languages: Collected experiences. In *OOPSLA 4th Workshop on Domain-Specific Modeling* (2004), ACM.

[26] MA, H., SHAO, W., ZHANG, L., MA, Z., AND JIANG, Y. Applying OO Metrics to Assess UML Meta-models. In *UML 2004, LNCS* (2004), vol. 3273, Springer-Verlag, pp. 12–26.

[27] MARTINS, C., AND SILVA, A. R. Modeling User Interfaces with the XIS UML Profile. In *Proceedings of the ICEIS 2007* (2007).

[28] MAXWELL, T., AND COSTANZA, R. A Language for Modular Spatio-Temporal Simulation. *Ecological modelling 103*, 2 (1997), 105–113.

[29] MERNIK, M., HEERING, J., AND SLOANE, A. M. When and how to develop domain-specific languages. *ACM Computing Surveys 37* (2005), 316–344.

[30] MORAIS, F., AND SILVA, A. R. Assessing the quality of user-interface modeling languages. In *Proceedings of 17th International Conference on Enterprise Information Systems (ICEIS)* (April 2015), Springer-Verlag, Ed., vol. 2, INSTICC, SCITEPRESS, pp. 311–319.

[31] NYSETVOLD, A. G., AND KROGSTIE, J. Assessing business processing modeling languages using a generic quality framework, 2005. Norwegian University of Science and Technology.

[32] OBJECT MANAGEMENT GROUP. *OMG Unified Modelling Language (OMG UML), Infrastructure - Version 2.4.1*. Object Management Group, 2011.

[33] OBJECT MANAGEMENT GROUP. *OMG Systems Modelling Language (OMG SysML) - Version 1.3*. Object Management Group, 2012.

[34] OBJECT MANAGEMENT GROUP. *Business Process Model and Notation (BPMN) - Version 2.0.2*. Object Management Group, 2013.

[35] OBJECT MANAGEMENT GROUP. *Interaction Flow Modelling Language (IFML) - FTF - Beta 2 - Revision 21*. Object Management Group, 2014.

[36] PATERNÓ, F., SANTORO, C., AND SPANO, L. D. MARIA: A Universal, Declarative, Multiple, Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments. *ACM Transactions on Computer-Human Interaction 16* (2009), 1–30.

[37] RECH, J., AND BUNSE, C. *Model-Driven Software Development: Integrating Quality Assurance*. Information Science Reference, 2009.

[38] RIBEIRO, A. Development of Mobile Applications using a Model-Driven Software Development Approach. Master's thesis, Instituto Superior Técnico, 2014.

[39] RIBEIRO, A., AND SILVA, A. R. Evaluation of XIS-Mobile, a Domain Specific Language for Mobile Application Development. In *Journal of Software Engineering and Applications* (2014), no. 7 in 11, Scientific Research Publishing.

[40] RIBEIRO, A., AND SILVA, A. R. XIS-Mobile: A DSL for Mobile Applications. In *Proceedings of ACM SAC 2014 Conference* (2014), ACM.

[41] RUMPE, B. Executable modeling with uml. a vision or a nightmare? *arXiv preprint arXiv:1409.6597* (2014).

[42] SILVA, A. R. Model-Driven Engineering: A Survey Supported by a Unified Conceptual Model. In *Computer Languages, Systems & Structures* (2015), Elsevier, pp. 1–25.

[43] SILVA, A. R., DE SOUSA SARAIVA, J., SILVA, R., AND MARTINS, C. XIS - UML Profile for eXtreme Modelling Interactive Systems. In *Proceedings of the MOMPES 2007* (2007), IEEE Computer Society.

[44] SILVA, A. R., LEMOS, G., MATIAS, T., AND COSTA, M. The XIS Generative Programming Techniques. In *Proceedings of the 27th COMPSAC Conference* (2003), IEEE Computer Society.

[45] SILVA, P. P. *Object Modelling of Interactive Systems: The UMLi Approach*. PhD thesis, University of Manchester, 2002.

[46] SILVA, P. P., AND PATON, N. W. UMLi: The Unified Modeling Language for Interactive Applications. In *3rd International Conference on the Unified Modeling Language* (2000), LNCS, Ed., vol. 1939, Springer, pp. 117–132.

[47] SILVA, P. P., AND PATON, N. W. User Interface Modeling in UMLi. In *IEEE Software* (2003), vol. 20, IEEE Computer Society, pp. 62–69.

[48] STREMBECK, M., AND ZDUN, U. An approach for the systematic development of domain-specific languages. *Software Practice and Experience 39* (2009), 1253–1292.

[49] TEEUW, W. B., AND VAN DER BERG, H. On the Quality of Conceptual Models. In *16th International Conference on Conceptual Modeling - ER'97* (1997), vol. 1331, Springer-Verlag, pp. 1–18.

[50] TRÆTTEBERG, H. Integrating Dialog Modeling and Domain Modeling - the Case of Diamodl and the Eclipse Modeling Framework. In *Journal of Universal Computer Science* (2008), vol. 14, J.UCS, pp. 3265–3278.

[51] UNITED STATES DEPARTMENT OF DEFENSE. *Department of Defense Dictionary of Military and Associated Terms*. Joint Chiefs of Staff, 2010.

[52] VAN DEURSEN, A., KLINT, P., AND VISSER, J. Domain-specific languages. Tech. rep., Centrum voor Wiskunde en Informatica, 2000.

[53] VAN DEURSEN, A., KLINT, P., AND VISSER, J. Domain-specific languages: An annotated bibliography. *ACM SIGPLAN Notices 35*, 6 (June 2000), 26–36.

Appendix A

Documentation of the Evaluated Languages

This appendix shows the records of assessed languages on this work.

User-Interface Modelling Languages

UMLi

Last Version	1.0	Date	2002

Main Reference(s):

[Sil02] P. P. da Silva, Object Modelling of Interactive Systems: The UMLi Approach (PhD Thesis), Department of Computer Science, University of Manchester (2002)

Complementary Reference(s):

[SP03] P. P. da Silva and N. W. Paton, *User Interface Modeling in UMLi* , IEEE Software, Vol.20, No. 4, pp. 62-69 (2003)

[SP00] P. P. da Silva and N. W. Paton, *UMLi: The Unified Modeling Language for Interactive Applications* , 3rd International Conference on the Unified Modeling Language, LNCS Vol 1939, pp. 117-132, Springer (2000)

[SP03a] P. P. da Silva and N. W. Paton, *Improving UML Support for User Interface Design: A Metric Assessment of UMLi* , Proceedings of ICSE'2003, pp. 76-83, IFIP (2003)

Website(s):

http://trust.utep.edu/umli/

http://paulopinheiro.info/publications.html

Organization(s):

University of Manchester

Brief Description (extracted from main references):

UMLi is a conservative extension to UML focused on modeling user interfaces **[Sil02]**. It intends to bridge UML's natural gap on web application interfaces, due to its general-purpose nature and the modeler's difficulty of designing user interfaces and domain objects simultaneously **[SP00]**. UMLi uses an MB-UIDE approach, which provides the ability to model and implement user interfaces in a systematic way **[Sil02]**. Tasks are modelled using extended activity diagrams that include six constructors, which are put inside "containers", so the modeler has a better preview of the interface **[SP03]**.

Domain Type	Domain-Specific
Domain	Interactive Systems & Apps
Application Area	UI Design
Abstract Level	PIM

Abstract Syntax

#Views	4

User Interfaces, Integrated Activities, Data Types extension mechanisms and State Machines extension mechanisms.

#Classifiers	13

#Classes	#Properties	#Elements	#Enumerations	#UseCases
9	0	4	0	0
Others				
#Interfaces				
0				

Classes: InteractionClass, Container, FreeContainer, ActionInvoker, PrimitiveInteractionClass, Inputter, Displayer, Editor (all from UserInterface Package/View), InitialInteraction (from DataTypes extensions).
Elements: SelectionState, OptionalState, OrderIndependentState, RepeatableState (all from IntegratedActivities Package/View).

#Relationships	11

#Associations	#Generalizations	#Realizations	#Compositions	#Dependencies
9	0	0	2	0

Associations: Fork, Join (both from StateMachine Package/View), <<presents>>, <<interacts>>, <<confirms>>, <<cancels>>, <<activate>> (from Interaction Object Flows model), <<communicates>>, <<uses>>.
Compostitions: Transition, ReturnTransition (IntegratedActivities Package/View).

Notes:

To enable construction of Interaction Object Flows, UMLi also adds OCL rules to the constructs of UML's StateMachine Package/View. Comparing UMLi's Metamodel to UML's, the User Interfaces and Integrated Activities packages are exclusive of UMLi, the State Machines and Data Types packages were extended from UML (which turned into new constructs and new OCL rules) and all the other packages are the same as in UML **[Sil02]**.

Concrete Syntax

Representation Type	Graphical
Supporting Mechanism	Core meta-modeling

Notes:

Specific Properties

Application Actions		OK, Cancel, Search, Back, Next, Up, Down, Quit, other customizable actions.
User Interactions		Click, Select, (Keyboard) Type, Scroll.
Pattern Usage		Abstract Presentation Pattern, Concrete Interaction Object.
Tool Support	Validation	Its grammar is specified in terms of the UML metamodel, which makes it possible to import UMLi models to ARGOi, the tool that can also import UML models from ArgoUML **[Sil02]**.
	Model to Model Transformations	No. There could be more compatibility between UML and UMLi models, but that doesn't exist due to LOTOS specifications non-trivial mapping with UMLi constructs **[Sil02]**.
	Model to Text Transformations	No. Despite ArgoUML's facilities to generate code in C++, C#, Java, PHP and SQL languages from models, that feature wasn't implemented for UMLi models. It is part of Future Work referred by the author **[Sil02]**.
Widget Types		Label, Text field, Combo box, Selectable list, Button.
Other Features		It is composed by several packages and has transition constructs for them. It is possible to model Activity Diagrams in UMLi, using the Use Case Diagram and the InitialInteraction construct, which identifies entry points for interactive applications in activity diagrams. ARGOi is UMLi's model development tool and an extension of ArgoUML **[Sil02]**.

Quality Properties

(Alternative Rating: 1 [Very Low], 2 [Low], 3 [Medium], 4 [High], 5 [Very High])

(property evaluation based on evaluator's perception and little experience of ArgoUML)

Usability	Understandability		4 [High]. Its concepts are easy to understand, due to being easy and a small set.
	Learnability	1. XP with other MLs	4 [High]. For amateur users, it's very easy to work with this extension, specially because there's the concept that inside an element there shall be other elements and different elements have different functions and representations, which are well documented.
		2. Replacing MLs	2 [Low]. Being a "conservative extension" to UML, it is limited, very dependable on UML and therefore, not expected to replace other UIMLs.
	Operability		3 [Medium]. It's easy to create User Interface Diagrams, as it's only needed to select object and drag it to the workspace, like in ArgoUML.
	Attractiveness	1. Dealing with ML's Concepts knowing the Domain	4 [High]. Its stereotypes are easy to memorize, due to familiar and small number of symbols.
		2. Using ML's Notation and Supporting Mechanism	3 [Medium]. On UMLi, it is easy to identify the interaction elements, but it isn't possible to change their state or behaviour when the interaction context changes.
	Adaptability	1. Dealing with ML's Concepts not knowing the Domain	4 [High]. It is easy, since they are simple and a small set.
		2. ML's compatible programs	4 [High]. It is easy, because it integrates and is fully compatible with UML applications.
	User Satisfaction	1. Confort with ML	(N/A)
		2. This ML versus PLs	(N/A)

Maintenance	Stability		5 [High]. The project has been discontinued, so the last and only version is stable.

	Changeability	1 [Very Low]. It would require changing LOTOS semantics and OCL rules. Since this project has ended, it's impossible to do that.
	Consistency	Yes
	Reusability	(N/A)
	Extensibility	3 [Medium] 1. There are diagrams to represent existing interaction spaces, although isn't a diagram to represent the navigation between them. 2. There are interaction elements on those diagrams that could be used for the referred naviagtion.

Interoperability	Compatibility - Compatible Applications	Number	1-5 (1)
		List of Items	ArgoUML
	Compatibility - Integration Mechanisms	Number	1-5 (3)
		List of Items	ARGOi, OCL rules and LOTOS rules
	Model Format		LOTOS and XMI

Comprehension	Communication	4 [High]. The fact that User Interface Diagram uses easily understandable and memorizable symbols may help analyzing and explaining models.
	Expressiveness	3 [Medium]. The User Interface Diagram is easily described, visually clear and easy readable, in opposition to the extended Activity Diagram, adapted from the Use Cases Diagram, which is extensive and visually complex.

Overall Quality

(N/A)

UsiXML

Last Version	1.2	Date	2004

Main Reference(s):

[LVM04] Q. Limbourg, J. Vanderdonckt, B. Michotte et al., *UsiXML: a User Interface Description Language for Specifying Multimodal User Interfaces* (2004)

Complementary Reference(s):

[VC09] J. Vanderdonckt and J. M. Calleros, *UsiXML, a User Interface Model and Language a User Interface Model and Language Engineering approach* (2009)

[SV03] N. Souchon and J. Vanderdonckt, *A Review of XML-Compliant User Interface Description Languages* (2003)

[FV10] D. Faure and J. Vanderdonckt, *USer Interface eXtensible Markup Language,* Workshop EICS'2010, ACM (2010)

Website(s):

http://www.usixml.org/en/home.html?IDC=221

http://www.usixml.eu/

Organization(s):

Université catholique de Louvain (UCL), Information Technology for European Advancement (ITEA 2)

Brief Description (extracted from main references):

USIXML is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces (AUIs) and Multimodal User Interfaces (MUIs) **[usixml.org]**. It allows to design interaction supporting the µ7 concept: multi-device, multi-user, multi-culturality/linguality, multi-organization, multi-context, multi-modality, and multi-platform **[usixml.eu]**.

Domain Type	Domain-Specific
Domain	Desktop Apps
Application Area	UI Design
Abstract Level	PIM

Abstract Syntax

#Views	5

Task model, Domain model, Context model, Abstract User Interface, Concrete User Interface.

#Classifiers	12

#Classes	#Properties	#Elements	#Enumerations	#UseCases
12	0	0	0	0
Others				

Classes: auiModel, cuiModel, contextModel, resourceModel, domainModel, mappingModel, taskModel, transformationModel **[VC09]**, archetypalModel, userModel, platformModel, environmentModel **[LVM04]**.
Properties: —.
Elements: —.
Enumerations: —.
Use Cases: —.

#Relationships	20

#Associations	#Generalizations	#Realizations	#Compositions	#Dependencies

Associations: Temporal, Operation, Container, Collection, Element (task relationships), Ad Hoc (domain relationship), Suspend, Resume, Disables, Enables, Grouping (AUI and AIO relationships), Equal, Meets, Overlaps, During, Starts, Finishes (spatio-temporal relationships) **[LVM04]**.
Generalizations: Decomposition (task relationship), (itself) (domain relationship) **[LVM04]**.
Realizations: —.
Compositions: Aggregation (domain relationship) **[LVM04]**.
Dependencies: —.

Concrete Syntax

Representation Type	Textual
Supporting Mechanism	XML Metamodel and Cameleon reference framework

Notes:

The Cameleon reference framework predicts building Achetypal models, starting from Ontological models, using successive steps such as capturing the essential concepts, users and tasks to create the Concepts and Task Model, designing an Abstract Interface, defining the platform and the environment to create the Concrete Interface and taking into account the evolution and transition, building the Final UI lining up with the Runtime Infrastructure and the Observed models, which shall be obtained directly from the Ontological models.

Specific Properties

Application Actions		(N/A)
User Interactions		Move pointer, Click, Double click, Depress, Release, Drag over, Drag drop, Focus, Select, Choose, Toggle, View
Pattern Usage		Concepts & Task Model, Abstract UI, Concrete UI, Final UI, Inter-model mapping, Context translation
Tool Support	Validation	(N/A)
	Model to Model Transformations	(N/A)
	Model to Text Transformations	(N/A)
Widget Types		Push button, List box, Check box, Window, Panel, Table, Cell, Dialog box, Embeded multimedia, Menu, Spin button
Other Features		It implements the µ7 concept, as it is Device-, User-, Culturality-, Organization-, Context-, Modality- and Platform-Independent.

Quality Properties

(Alternative Rating: 1 [Very Low], 2 [Low], 3 [Medium], 4 [High], 5 [Very High])

(property evaluation based on evaluator's perception)

Usability	Understandability		(N/A)
	Learnability	1. XP with other MLs	(N/A)
		2. Replacing MLs	(N/A)
	Operability		(N/A)
	Attractiveness	1. Dealing with ML's Concepts knowing the Domain	(N/A)
		2. Using ML's Notation and Supporting Mechanism	(N/A)
	Adaptability	1. Dealing with ML's Concepts not knowing the Domain	(N/A)
		2. ML's compatible programs	(N/A)
	User Satisfaction	1. Confort with ML	(N/A)
		2. This ML versus PLs	(N/A)

Maintenance	Stability	(N/A)
	Changeability	(N/A)
	Consistency	Yes
	Reusability	4 [High]. USIXML provides the designer with a set of pre-defined relationships allowing to map elements from heterogeneous models. This may be useful, for instance, for architecture derivation, for traceability in the development cycle, for addressing context sensitive issues or even for improving the preciseness of model derivation heuristics **[LVM04]**.
	Extensibility	(N/A)

Interoperability	Compatibility - Compatible Applications	Number	>6 (N/A)
		List of Items	Eclipse, Enterprise Architect, Kate, Microsoft Visual Studio, Notepad++, W3schools.com and others

	Compatibility - Integration Mechanisms	Number	1-5 (4)
		List of Items	MDG Technologies, XML Parser, UsiGesture and UsiDistrib
	Model Format		(N/A)

Comprehension	Communication	(N/A)
	Expressiveness	(N/A)

Overall Quality

(N/A)

XIS

Last Version	2.0	Date	2007

Main Reference(s):

[Mar07] C. Martins, *Modelação de Interfaces Gráficas no âmbito do ProjectIT (Master Thesis)* , Universidade da Madeira (2007)
[Mar07a] C. Martins, *Modelação de Interfaces Gráficas no âmbito do ProjectIT - Descrição Técnica do Perfil XIS* (2007)

Complementary Reference(s):

[MS07] C. Martins and A. R. Silva, *Modeling User Interfaces with the XIS UML Profile (ICEIS'07)* (2007)
[Sil04] A. R. Silva, *The XIS Approach and Principles (Euromicro'04)* (2003)
[SSS07] A. R. Silva, J. Saraiva, R. Silva et al., *XIS – UML Profile for eXtreme Modeling Interactive Systems (MOMPES'07)* (2007)

Website(s):

http://isg.inesc-id.pt/alb/ProjectIT@81.aspx

Organization(s):

INESC-ID

Brief Description (extracted from main references):

XIS is a UML Profile that allows modelling of interactive systems, using a platform-independent approach and model-driven development guidelines. It uses model-to-text transformation mechanisms that receive models as input and return code as output for several languages and technological platforms **[Mar07]**.

Domain Type	Domain-Specific
Domains	Desktop Interactive Apps
	Web Interactive Apps
Application Area	UI Design
Abstract Level	PIM

Abstract Syntax

#Views	6

Domain, Business Entities, Actors, Use Cases, Navigation Space and Interaction Space.

#Classifiers	18

#Classes	#Properties	#Elements	#Enumerations	#UseCases
4	3	8	1	0
Others				
#EnumerationLiterals	#NamedElements			
1	1			

Classes: XisEntity, XisBusinessEntity, XisActor, XisUseCase.
Properties: XisEntityAttribute, XisEntityAssociationEnd, XisBusinessDetailAssociation.
Elements: XisInteractionElement, XisInteractionSimpleElement, XisInteractionCompositeElement, XisActionElement, XisDataElement, XisDataTable, XisDomainElement and XisOtherElement.
Enumerations: XisEnumeration.
EnumerationLiterals: XisEnumerationValue.
NamedElements: XisInteractionSpace.

#Relationships	12

#Associations	#Generalizations	#Realizations	#Compositions	#Dependencies
11	1	0	0	0

Associations: XisEntityAssociation, XisBusinessComposedByAssociation, XisBusinessMasterAssociation, XisInheritanceAssociation, XisPerformsAssociation, XisOperatesOnAssociation, XisNavigationAssociation, XisElementPermission, XisPerformsNavigationAssociation, XisDomainAssociation and XisDomainAttributeAssociation.
Generalizations: XisInheritance.

Concrete Syntax

Representation Type	Graphical
Supporting Mechanism	UML Profile

Notes:

Specific Properties

Application Actions		CRUD, OK, Cancel, Navigate, Select, Close, Associate, Dissociate, other customizable actions.
User Interactions		Click, Select, Drag, (Keyboard) Type.
Pattern Usage		**Structural Design:** Composite. **UI Design**: Single Choice, Multiple Choice, List Selection, Continuous Filter, Menu Navigation, Grid Layout, Tab Menu, Tabular Set, Double List.
Tool Support	Validation	Supports transformations, using Project IT's three components: Requirements, UML Modeler and MDD Code Generator. The first one defines and manages requirements, the second is responsible for defining and managing models and the third sets and executes application generation processes. It also uses Eclipse .NET and .NET Framework **[Mar07]**.
	Model to Model Transformations	No. It was defined conceptually, but it wasn't yet implemented **[Mar07]**.
	Model to Text Transformations	Yes, defining architectures, templates, and interface generation processes that are compatible with Windows Forms.NET and ASP.NET platforms **[Mar07]**.
Widget Types		Button, Text box, List, Menu, Window, Link, Search bar, Checkbox, Radius button, Drop-down list, Form, Dialog, Label, Image, ComboBox, Date, Menu Item, Menu Separator, Tab, Group Box, List Select, Table, Table Associate, Dialog Exclamation, Dialog Information, Dialog Warning, Dialog Question and Dialog Error.
Other Features		Supports Windows Forms.NET, ASP.NET and JSP, using Model-to-Text transformations **[Mar07]**.

Quality Properties

(Alternative Rating: 1 [Very Low], 2 [Low], 3 [Medium], 4 [High], 5 [Very High])

(property evaluation based on the analysis of the author's thesis [XISTHESIS02] and evaluator's perception)

Usability	Understandability		(N/A)
	Learnability	1. XP with other MLs	(N/A)
		2. Replacing MLs	(N/A)
	Operability		(N/A)
	Attractiveness	1. Dealing with ML's Concepts knowing the Domain	(N/A)
		2. Using ML's Notation and Supporting Mechanism	(N/A)
	Adaptability	1. Dealing with ML's Concepts not knowing the Domain	(N/A)
		2. ML's compatible programs	(N/A)
	User Satisfaction	1. Confort with ML	(N/A)
		2. This ML versus PLs	(N/A)

Maintenance	Stability	(N/A)
	Changeability	(N/A)
	Consistency	Yes.
	Reusability	There are templates that can be reused between languages, but it is always necessary to build a template and an architecture for each language.
	Extensibility	1. Some interaction elements have associated marks that are Enumerates, which are lists of pre-defined values. XIS was developed in a way that allows to extend these lists with new generic controls or new interaction elements patterns. That would imply creating generation mechanisms and changing their templates. 2. It could also have included the possibility of generating and

		presenting reports, that could be represented as a special type of interaction space. 3. OCL could be used to specify elements' restriction based on interaction contexts. 4. It is possible to model different access to elements, depending on the actor, but it isn't possible to generate that specification on text. 5. It would raise productivity if some Model 2 Model Transformations were added, specially to generate User-Interfaces View from BusinessEntities View and Use-Cases View. 6. Transforming XIS models in XML-based languages would allow using already existing rendering mechanisms and future ones. It would improve efficiency and reduce inconsistency. 7. Since XIS is PIM, it would be interesting to see developed templates for non-Microsoft platforms, such as Java or mobile technologies, namely Android and iOS.

Interoperability	Compatibility - Compatible Applications	Number	1-5 (1)
		List of Items	ProjectIT Studio
	Compatibility - Integration Mechanisms	Number	1-5 (2)
		List of Items	Eclipse .NET and .NET Framework
	Model Format		TPL and ASPX

Comprehension	Communication	(N/A)
	Expressiveness	(N/A)

Overall Quality

(N/A)

XIS-Mobile

Last Version	1.3	Date	2014

Main Reference(s):

[Rib14] A. Ribeiro, *Development of Mobile Applications using a Model-Driven Software Development Approach (Master Thesis)* , Instituto Superior Técnico (2014)

Complementary Reference(s):

[RS14] A. Ribeiro and A. R. Silva, *XIS-Mobile – A DSL for Mobile Applications,* Proc of the 29th Symposium on Applied Computing (SAC'14), ACM (2014)

[RS12] A. Ribeiro and A. R. Silva, *Survey on Cross-Platforms and Languages for Mobile Apps* , Procedure of QUATIC'2012 Conference, IEEE Computer Society (2012)

[RS14a] A. Ribeiro and A. R. Silva, *Comparative Analysis of Workbenches to support DSML* (2014)

[RS14b] A. Ribeiro and A. R. Silva, *Evaluation of XIS-Mobile, a Domain Specific Language for Mobile Application Development* , Journal of Software Engineering and Applications (2014)

Website(s):

https://github.com/xis-mobile/XIS-Mobile

Organization(s):

INESC-ID

Brief Description (extracted from main references):

XIS-Mobile is defined as a language based on XIS **[Rib14]**, since it is likewise Model-Driven Development-oriented, supported by a UML Profile and suited for modelling platform-independent applications, but is focused on mobile environment, instead of desktop. It was designed to mitigate problems related to app development, such as specificity of each platform (leads to incompatibility), specific development tools (which causes software development complexity), application markets and mobile OS platform fragmentation **[RS14]**.

Domain Type	Domain-Specific
Domain	Mobile Interactive Apps
Application Area	UI Design
Abstract Level	PIM

Abstract Syntax

#Views	6

Domain, Business Entities, Use Cases, Architectural, Navigation Space and Interaction Space.

#Classifiers	46

#Classes	#Properties	#Elements	#Enumerations	#UseCases
34	1	0	1	3
Others				
#EnumerationLiterals	#Actors	#Interfaces	#Operations	
1	1	3	2	

Classes: XisEntity, XisBusinessEntity, XisProvider, XisServer, XisClientMobileApp, XisInteractionSpace, XisWidget, XisGesture, XisSimpleWidget, XisCompositeWidget, XisLabel, XisTextBox, XisCheckBox, XisButton, XisLink, XisImage, XisDatePicker, XisTimePicker, XisWebView, XisMapView, XisMapMarker, XisDropDown, XisMenuItem, XisRadioButton, XisList, XisListItem, XisListGroup, XisVisibilityBoundary, XisForm, XisMenuGroup, XisMenu, XisDialog, XisMobileApp, XisInternalProvider.
Properties: XisEntityAttribute.
Enumerations: XisEnumeration.
Use Cases: XisUseCase, XisEntityUseCase, XisServiceUseCase.
Enumeration Literals: XisEnumerationLiteral.
Actors: XisActor.
Interfaces: XisService, XisRemoteService, XisInternalService.
Operations: XisServiceMethod, XisAction.

#Relationships	16

#Associations	#Generalizations	#Realizations	#Compositions	#Dependencies
14	1	1	0	0

Associations: XisEntityAssociation, XisBE-EntityMasterAssociation, XisBE-EntityDetailAssociation, XisBE-EntityReferenceAssociation, XisActor-UCAssociation, XisEntityUC-BEAssociation, XisServiceUC-BEAssociation, XisServiceUC-ProviderAssociation, XisMobileApp-ServiceAssociation, XisInteractionSpaceAssociation, XisIS-BEAssociation, XisWidget-GestureAssociation, XisIS-MenuAssociation, XisIS-DialogAssociation.
Generalizations: XisEntityInheritance.
Realizations: XisProvider-ServiceRealization.

Concrete Syntax

Representation Type	Graphical
Supporting Mechanism	UML Profile

Notes:

Specific Properties

Application Actions		CRUD, OK, Cancel, Delete All, Open Browser, Web Service, Navigate, Select, other customizable actions.
User Interactions		Tap, Double Tap, Long Tap, Swipe, Pinch, Stretch, (Touchscreen) Type.
Pattern Usage		**Structural Design:** Composite. **UI Design**: Single Choice, Multiple Choice, List Selection, Single Text Entry, Multiple Text Entry, Springboard, List Menu, Tab Menu, Option Menu, Context Menu, Form, List, Explicit Search, Search Results, Filter, Call to Action Button, Dialog, Map, Location.
Tool Support	Validation	XIS Mobile Framework, namely EA's Model Validator, supports model validation **[Rib14]**.
	Model to Model Transformations	Yes, using Enterprise Architect's MDG Technologies **[Rib14]**.
	Model to Text Transformations	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 **[Rib14]**.
Widget Types		Button, Text box, List, Menu, Window, Link, Search bar, Checkbox, Radius button, Drop-down list, Label, Image, Date Picker, Time Picker, Web View, Map View, Menu Item, List Item, List Group, Menu Group, Dialog, Form.
Other Features		Model validation uses a set of rules defined in C#, implemented with EA's Automation Interface. It is possible to generate XIS Mobile models on EA, right-clicking the mouse, selecting "Extensions" > "XIS-Mobile Plugin" > "Generate Models" **[Rib14]**.

Quality Properties

(Alternative Rating: 1 [Very Low], 2 [Low], 3 [Medium], 4 [High], 5 [Very High])

(property evaluation based on a workshop performed at INESC-ID, with 9 users [XISMOBTHESIS14, Annex C])

Usability	Understandability		3 [Medium], average answer in the workshop feedback.
	Learnability	1. XP with other MLs	5 [Very High], average answer in the workshop feedback.
		2. Replacing MLs	4 [High], average answer in the workshop feedback.
	Operability		2 [Low], average answer in the workshop feedback.
	Attractiveness	1. Dealing with ML's Concepts knowing the Domain	3 [Medium], average answer in the workshop feedback.
		2. Using ML's Notation and Supporting Mechanism	4 [High], average answer in the workshop feedback.
	Adaptability	1. Dealing with ML's Concepts not knowing the Domain	4 [High], average answer in the workshop feedback.
		2. ML's compatible programs	4 [High], average answer in the workshop feedback.
	User Satisfaction	1. Confort with ML	3 [Medium], average answer in the workshop feedback.
		2. This ML versus PLs	4 [High], average answer in the workshop feedback.

Maintenance	Stability	4 [High]. There has been 3 major iterations during the 18-month development timeframe, always focused on model validation and generation, create and evaluate test cases and on improvement of XIS Mobile framework.
	Changeability	2 [Low]. Taking into account only the language, a change is made in the UML profile using EA. Along with the Framework, a change may force to modify e.g. toolboxes or diagrams profiles or Model-to-Text and Model-to-Model transformations. From the developer's point of view, this feature

		should be more automatic, for instance introducing scripts.
	Consistency	Yes. The language and its notation are consistents. In the process of creating models, the consistency level depends on the developer/modeler. The only restrictions are defined by the UML Profile (metaclasses types, stereotypes, tagged values and connections beetween these concepts).
	Reusability	5 [Very High]. The language is prepared to be reused for another domains that make sense. For example, if it is wanted to develop a language for desktop applications, the major part of concepts can be reused, needing only to add others that are necessary for that specific domain.
	Extensibility	5 [Very High]. It is possible to extend XIS Mobile in an easy way, due to inheritance relationships. It is also possible to aggregate similar concepts.

Interoperability	Compatibility - Compatible Applications	Number	1-5 (1)
		List of Items	Enterprise Architect
	Compatibility - Integration Mechanisms	Number	1-5 (2)
		List of Items	MDG Technologies and XML Parser
	Model Format		XMI

Comprehension	Communication	5 [Very High]. XIS Mobile uses a UML Profile, which is a standard largely used. For users that have experience with UML, it is relatively easy to understand the diagrams, which makes documentation comprehension and model design easier.
	Expressiveness	4 [High]. InteractionSpace View's representation should be more appealing and adjusted to what the user really sees, due to the quantity of

		concepts that it contains and frequency each one is used.

Overall Quality

(N/A)

	Abstract Syntax			Concrete Syntax			Specific Properties			Quality Properties										
	#Views	#Classifiers	#Relationships	Representation Type	Supporting Mechanism	Compatibility	Application Actions	User Interactions	Widget Types	Pattern Usage	Tool Support - Validation	Tool Support - Model to Model Transformations	Tool Support - Model to Text Transformations	Other Features	Reusability	Extensibility	# of Compatible Applications	# of Integration Mechanisms	Communication	Expressiveness
UMLi	4	13	11	Graphical	Core meta-modeling	ArgoUML	OK, Cancel, Search, Back, Next, Up, Down, Quit, other customizable actions.	Click, Select, (Keyboard) Type, Scroll.	Label, Text field, Combo box, Selectable list, Button.	Abstract Presentation Pattern, Concrete Interaction Object.	Its grammar is specified in terms of the UML metamodel, which makes it possible to import UMLi models to ARGOi, the tool that can also import UML models from ArgoUML [Sil02].	No. There could be more compatibility between UML and UMLi models, but that doesn't exist due to LOTOS specifications non-trivial mapping with UMLi constructs [Sil02].	No. Despite ArgoUML's facilities to generate code in C++, C#, Java, PHP and SQL languages from models, that feature wasn't implemented for UMLi models. It is part of Future Work referred by the author [Sil02].	It is composed by several packages and has transition constructs for them. It is possible to model Activity Diagrams in UMLi, using the Use Case Diagram and the InitialInteraction construct, which identifies entry points for interactive applications in activity diagrams. ARGOi is UMLi's model development tool and an extension of ArgoUML [Sil02].	(N/A)	3 [Medium] 1. There are diagrams to represent existing interaction spaces, although isn't a diagram to represent the navigation between them. 2. There are interaction elements on those diagrams that could be used for the referred naviagtion.	1-5 (1)	1-5 (3)	4 [High]. The fact that User Interface Diagram uses easily understandable and memorizable symbols may help analyzing and explaining models.	3 [Medium]. The User Interface Diagram is easily described, visually clear and easy readable, in opposition to the extended Activity Diagram, adapted
UsiXML	5	12	20	Textual	XML Metamodel and Cameleon reference framework	Eclipse, Enterprise Architect, Kate, Microsoft Visual Studio, Notepad++, W3schools.com and others	(N/A)	Move pointer, Click, Double click, Depress, Release, Drag over, Drag drop, Focus, Select, Choose, Toggle, View	Push button, List box, Check box, Window, Panel, Table, Cell, Dialog box, Embeded multimedia, Menu, Spin button	Concepts & Task Model, Abstract UI, Concrete UI, Final UI, Inter-model mapping, Context translation	(N/A)	(N/A)	(N/A)	It implements the µ7 concept, as it is Device-, User-, Culturality-, Organization-, Context-, Modality- and Platform-independent.	4 [High]. USIXML provides the designer with a set of pre-defined relationships allowing to map elements from heterogeneous models. This may be useful, for instance, for architecture derivation, for traceability in the development cycle, for addressing context sensitive issues or even for improving the preciseness of model derivation heuristics [LVM04].	(N/A)	>6 (N/A)	1-5 (4)	(N/A)	(N/A)
XIS	6	18	12	Graphical	UML Profile	ProjectIT Studio	CRUD, OK, Cancel, Navigate, Select, Close, Associate, Dissociate, other customizable actions.	Click, Select, Drag, (Keyboard) Type.	Button, Text box, List, Menu, Window, Link, Search bar, Checkbox, Radius button, Drop-down list, Form, Dialog, Label, Image, ComboBox, Date, Menu Item, Menu Separator, Tab, Group Box, List Select, Table, Table Associate, Dialog Exclamation, Dialog Information, Dialog Warning, Dialog Question and Dialog Error.	Structural Design: Composite. UI Design: Single Choice, Multiple Choice, List Selection, Continuous Filter, Menu Navigation, Grid Layout, Tab Menu, Tabular Set, Double List.	Supports transformations, using Project IT's three components: Requirements, UML Modeler and MDD Code Generator. The first one defines and manages requirements, the second is responsible for defining and managing models and the third sets and executes application generation processes. It also uses Eclipse .NET and .NET Framework [Mar07].	No. It was defined conceptually, but it wasn't yet implemented [Mar07].	Yes, defining architectures, templates, and interface generation processes that are compatible with Windows Forms.NET and ASP.NET platforms [Mar07].	Supports Windows Forms.NET, ASP.NET and JSP, using Model-to-Text transformations [Mar07].	There are templates that can be reused between languages, but it is always necessary to build a template and an architecture for each language.	1. Some interaction elements have associated marks that are Enumerates, which are lists of pre-defined values. XIS was developed in a way that allows to extend these lists with new generic controls or new interaction elements patterns. That would imply creating generation mechanisms and changing their templates. 2. It could also have included the possibility of generating and presenting reports, that could be represented as a special type of interaction space. 3. OCL could be used to specify elements' restriction based on interaction contexts. 4. It is possible to model different access to elements, depending on the actor, but it isn't possible to generate that specification on text. 5. It would raise productivity if some Model 2 Model Transformations were added, specially to generate User-Interfaces View from BusinessEntities View and Use-Cases View. 6. Transforming XIS models in XML-based languages would allow using already existing rendering mechanisms and future ones. It would improve efficiency and reduce inconsistency. 7. Since XIS is PIM, it would be interesting to see developed templates for non-Microsoft platforms, such as Java or mobile technologies, namely Android and iOS.	1-5 (1)	1-5 (2)	(N/A)	(N/A)
XIS Mobile	6	46	16	Graphical	UML Profile	Enterprise Architect	CRUD, OK, Cancel, Delete All, Open Browser, Web Service, Navigate, Select, other customizable actions.	Tap, Double Tap, Long Tap, Swipe, Pinch, Stretch, (Touchscreen) Type.	Button, Text box, List, Menu, Window, Link, Search bar, Checkbox, Radius button, Drop-down list, Label, Image, Date Picker, Time Picker, Web View, Map View, Menu Item, List Item, List Group, Menu Group, Dialog, Form.	Structural Design: Composite. UI Design: Single Choice, Multiple Choice, List Selection, Single Text Entry, Multiple Text Entry, Springboard, List Menu, Tab Menu, Option Menu, Context Menu, Form, List, Explicit Search, Search Results, Filter, Call to Action Button, Dialog, Map, Location.	XIS Mobile Framework, namely EA's Model Validator, supports model validation [Rib14].	Yes, using Enterprise Architect's MDG Technologies [Rib14].	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 [Rib14].	Model validation uses a set of rules defined in C#, implemented with EA's Automation Interface. It is possible to generate XIS Mobile models on EA, right-clicking the mouse, selecting "Extensions" > "XIS Mobile Plugin" > "Generate Models" [Rib14].	5 [Very High]. The language is prepared to be reused for another domains that make sense. For example, if it is wanted to develop a language for desktop applications, the major part of concepts can be reused, needing only to add others that are necessary for that specific domain.	5 [Very High]. It is possible to extend XIS Mobile in an easy way, due to inheritance relationships. It is also possible to aggregate similar concepts.	1-5 (1)	1-5 (2)	5 [Very High]. XIS Mobile uses a UML Profile, which is a standard largely used. For users that have experience with UML, it is relatively easy to understand the diagrams, which makes documentation	4 [High]. InteractionSpace View's representation should be more appealing and adjusted to what the user really sees, due to the quantity of concepts that it contains and

Falta acabar folhas do XIS e do USIXML

Fazer Download da aplicação Gephi, para gerar fórmulas, gráficos e grafos. Qualquer dúvida contactar barao.alexandre@gmail.com

Contactar barao.alexandre@gmail.com para usar Gelphi para criar um grafo de 2 ou + níveis para a minha framework e pô-la no meu website.

#Constructs = #Stereotypes + All Elements - Linking Elements

short_UIMLs_text

	Abstract Syntax			Concrete Syntax			Specific Properties			Quality Properties										
	#Views	#Classifiers	#Relationships	Representation Type	Supporting Mechanism	Compatibility	Application Actions	User Interactions	Widget Types	Pattern Usage	Tool Support - Validation	Tool Support - Model to Model Transformations	Tool Support - Model to Text Transformations	Other Features	Reusability	Extensibility	# of Compatible Applications	# of Integration Mechanisms	Communication	Expressiveness
UMLi	4	13	11	Graphical	Core meta-modeling	ArgoUML	> 8	4	5	2							1	3		
UsiXML	5	12	20	Textual	XML Metamodel and Cameleon reference framework	Eclipse, Enterprise Architect, Kate, Microsoft Visual Studio, Notepad++, W3schools.com and others	(N/A)	12	11	6							6	4		
XIS	6	18	12	Graphical	UML Profile	ProjectIT Studio	> 9	4	28	10							1	2		
XIS Mobile	6	46	16	Graphical	UML Profile	Enterprise Architect	> 8	7	22	20							1	2		
Avg UIMLs	5	22	15	3 Graphical,	2 UML Profile,	Melhor o UsiXML,	> 8	7	17	10				(N/A)			2	3		
#{Views, Pro	1	24	5	1 Textual	1 XML Metamodel,	depois XIS Mobile,														
#{Views, Pro	1	2	1		1 Core meta-modeling	depois UMLi e XIS														

Falta acabar fichas do XIS e do USIXML

Fazer Download da aplicação Gephi, para gerar fórmulas, gráficos e grafos. Qualquer dúvida contactar barao.alexandre@gmail.com

Contactar barao.alexandre@gmail.com para usar Gelphi para criar um grafo de 2 ou + níveis para a minha framework e pô-la no meu website.

#Constructs = #Stereotypes = All Elements - Linking Elements

short_UIMLs_num

Business Process Modelling Languages (selected diagrams)

UML (Activity Diagram)

Last Version	2.4.1	Date	2011

Main Reference(s):

[OMG11] Object Management Group (OMG), *Unified Modeling Language (UML) Superstructure Version 2.4.1* (August 2011)
[OMG11a] Object Management Group (OMG), *Unified Modeling Language (UML) Infrastructure Version 2.4.1* (August 2011)

Complementary Reference(s):

[AHK03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski et al., *Workflow Patterns* , Distributed and Parallel Databases, 14 (3), pp. 5-51 (July 2003)
[RHA06] N. Russell, A.H.M. Hofstede, W.M.P. van der Aalst et al., *Workflow Control-Flow Patterns: A Revised View* , BPM Center Report, BPMcenter.org (2006)
[Koc06] N. Koch, *Classification of Model Transformation Techniques used in UML-based Web Engineering* , Ludwig-Maximilians-Universität (2006)
[KHE03] J. M. Küster, R. Heckel and G. Engels, *Defining and Validating Transformations of UML Models* , University of Paderborn (2003)
[KL02] B. Korherr and B. List, *A UML 2 Profile for Event Driven Process Chains* , Vienna University of Technology (Set 2002)

Website(s):

http://www.omg.org/spec/UML/
http://www.workflowpatterns.com/evaluations/standard/uml.php

Organization(s):

Object Management Group (OMG)

Brief Description (extracted from main references):

The objective of UML is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes. One of the primary goals of UML is to advance the state of the industry by enabling object visual modeling tool interoperability. UML has a formal definition of a common MOF-based metamodel that specifies its abstract syntax, a detailed explanation of the semantics of each UML modeling concept, a specification of the human-readable notation elements for representing the individual UML modeling concepts and a detailed definition of ways in which UML tools can be made compliant with this specification, using an XML-based specification of corresponding model interchange formats (XMI) that must be realized by compliant tools **[OMG11]**. It is possible to use this diagram for a 1-to-1 mapping with a UML 2 Profile for Event-Driven Process Chain, helping software engineers to easily understand models that represent business processes and business requirements not needing to learn a new notation and tools **[KL02]**.

Domain Type	General-Purpose
Domain	Any (for instance Defense, Banking and Retail)
Application Area	Systems Design
Abstract Level	PIM

Abstract Syntax

#Views	1

Activity Diagram (single view)

#Classifiers	43

#Classes	#Properties	#Elements	#Enumerations	#UseCases
2	0	37	3	0
Others				
#EnumerationLiterals	#Actors	#Interfaces	#Operations	#Notes
0	0	0	0	1

Classes: Activity, Behavior **[OMG11, pp. 340-349, 372-373]**
Elements: AcceptEventAction, Action, ActionInputPin, ActivityFinalNode, ActivityParameterNode, ActivityPartition, AddVariableValueAction, BehavioralFeature, CallBehaviorAction, CallOperationAction, CentralBufferNode, Clause, ConditionalNode, DataStoreNode, DecisionNode, ExpansionNode, ExpansionRegion, FlowFinalNode, ForkNode, InitialNode, InputPin, InterruptibleActivityRegion, JoinNode, LoopNode, MergeNode, OutputPin, Parameter, ParameterSet, Pin, SendObjectAction, SendSignalAction, SequenceNode, StructuredActivityNode, UnmarshallAction, ValuePin, ValueSpecificationAction, Variable **[OMG11, pp. 333-450]**
Enumerations: ExpansionKind, ObjectNodeOrderingKind, ParameterEffectKind **[OMG11, pp. 393, 424, 427]**
Notes: Local pre- and postconditions **[OMG11, pp. 335-339, 449]**

#Relationships	3

#Associations	#Generalizations	#Realizations	#Compositions	#Dependencies
3	0	0	0	0

Associations: ControlFlow, ObjectFlow (Graphic Paths), ExceptionHandler (Graphic Element) **[OMG11, pp. 382-383, 389-392, 416-421]**.
Generalizations: —.
Realizations: —.
Compositions: —.
Dependencies: —.

Concrete Syntax

Representation Type	Graphical
Supporting Mechanism	MOF

Notes:

There are 7 elements on UML meta-model that weren't included on Classifiers list, because only concrete elements were counted: ActivityEdge, ActivityGroup, ActivityNode, ControlNode, ExecutableNode, FinalNode and ObjectNode.

Specific Properties

Executability		The UML Standard Profile defines several standard stereotypes that apply to Artifacts, e.g., «source» or «executable». There are specific types of components that can be deployed as Executable Artifacts and they can be related to a Node using the DeployedArtifact and DeploymentTarget elements **[OMG11, pp. 220-229]**.
List of Actions		(N/A)
Pattern Usage		**Control-Flow:** Sequence, Parallel Split, Synchronization, Exclusive Choice, Simple Merge, Multi-Choice, Multi-Merge, Arbitrary Cycles, Implicit Termination, Multiple Instances without Synchronization, Multiple Instances with a Priori Design-Time Knowledge, Multiple Instances with a Priori Run-Time Knowledge, Deferred Choice, Cancel Activity, Cancel Case, Structured Loop, Transient Trigger, Persistent Trigger, Cancel Region, Cancel Multiple Instance Activity, Cancelling Discriminator, Cancelling N-out-of-M Join, Thread Merge, Thread Split, Explicit Termination. **Data:** Block Data, Multiple Instance Data, Workflow Data, Task to Task, Block Task to SubWorkflow Decomposition, SubWorkflow Decomposition to Block Task, To Multiple Instance Task, From Multiple Instance Task, Data Transfer by Reference - With Lock, Data Transformation - Input, Data Transformation - Output, Task Precondition - Data Existence, Task Precondition - Data Value, Task Postcondition - Data Existence, Task Postconditon - Data Value, Event-Based Task Trigger, Data-Based Routing. **Resource:** Direct Allocation, Role-Based Allocation, Automatic Execution, Distribution by Allocation - Single Resource, Distribution on Enablement, Commencement on Creation, Chained Execution, Simultaneous Execution. **Exception Handling:** None.
Tool Support	Validation	XML uses XSD to validate instances of XML documents

		[OMG11, p. 739].
	Model 2 Model Transformations	Yes. It is possible to transform UML activities, Activity Edge, Call Behaviour Action, Decision Node and other graphical elements into BPMN, using UML 2.0 Diagram Interchange **[OMG11, pp. 319-431]**.
	Model 2 Text Transformations	Yes. IBM's Rational Software Architect allows UML-to-Java transformations **[ibm.com/developerWorks/]**. Also, it is possible to transform UML models into Communicating Sequential Processes (CSPs), using graphs **[KHE03]**.
Modularity		(N/A)
Other Features		UML-based Web Engineering allows several model transformations, such as Requirements2Content, Process2Navigation or Functionality2BigPicture, using implementation techniques namely Java, C#, AGG, VIATRA, ATL or QVT **[Koc06]**.

Quality Properties

(Alternative Rating: 1 [Very Low], 2 [Low], 3 [Medium], 4 [High], 5 [Very High])

(property evaluation based on evaluator's perception)

Usability	Understandability		(N/A)
	Learnability	1. XP with other MLs	(N/A)
		2. Replacing MLs	(N/A)
	Operability		(N/A)
	Attractiveness	1. Dealing with ML's Concepts knowing the Domain	(N/A)
		2. Using ML's Notation and Supporting Mechanism	(N/A)
	Adaptability	1. Deal with concepts	(N/A)
		2. ML's programs	(N/A)
	User Satisfaction	1. Confort with ML	(N/A)
		2. This ML versus PLs	(N/A)

Maintenance	Stability	(N/A)
	Changeability	(N/A)
	Consistency	(N/A)
	Reusability	3 [Medium]. A fine-grained, flexible metamodel library is provided that is reused to define the UML metamodel, as well as other architecturally related metamodels, such as the Meta Object Facility (MOF) and the Common Warehouse Metamodel (CWM) **[OMG11a, p. 23]**.
	Extensibility	4 [High]. The UML can be extended in two ways: • A new dialect of UML can be defined by using Profiles to customize the language for particular platforms (e.g., J2EE/EJB, .NET/COM+) and domains (e.g., finance, telecommunications, aerospace). • A new language related to UML can be specified by reusing part of the InfrastructureLibrary package and

		augmenting with appropriate metaclasses and metarelationships. The former case defines a new dialect of UML, while the latter case defines a new member of the UML family of languages **[OMG11a, p. 23]**.

Interoperability	Compatibility - Compatible Applications	Number	>6 (41)
		List of Items	AgileJ StructureViews, ArgoUML, Astah*, ATL, Borland Together, BOUML, CaseComplete, ConceptDraw PRO, Creately for UML, Dia, Eclipse UML2 Tools, Enterprise Architect, Gliffy, LucidChart, MagicDraw, Microsoft Visio, Modelio, MyEclipse, NClass, NetBeans, objectiF, Open ModelSphere, Papyrus, PlantUML, Poseidon for UML, PowerDesigner, Prosa UML Modeller, Rational Rhapsody, Rational Rose XDE, Rational Software Architect, Rational Software Modeler, Rational System Architect, Real Time Developer Studio, RISE, Software Ideas Modeler, StarUML, Umbrello UML Modeller, UMLet, UModel, Visual Paradigm for UML and yEd.
	Compatibility - Integration Mechanisms	Number	>6 (7)
		List of Items	MDA-driven, Exports to XMI, Generates languages, Generates languages using reverse-engineering, Can be integrated with IDEs, Can be integrated with Web Browsers, Can be integrated with Office applications
	Model Format		XML Schema

Comprehension	Communication	(N/A)
	Expressiveness	(N/A)

Overall Quality
(N/A)

BPMN (Business Process Diagram)

Last Version	2.0.2	Date	2013

Main Reference(s):

[OMG13] Object Management Group (OMG), *Business Process Model and Notation (BPMN) Version 2.0.2* (December 2013)

Complementary Reference(s):

[AHK03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski et al., *Workflow Patterns* , Distributed and Parallel Databases, 14 (3), pp. 5-51 (July 2003)

[RHA06] N. Russell, A.H.M. Hofstede, W.M.P. van der Aalst et al., *Workflow Control-Flow Patterns: A Revised View* , BPM Center Report, BPMcenter.org (2006)

[Tum15] J. Tuma, *Automatized model transformation connecting BPMN and BORM,* Czech University of Life Sciences Prague (2015)

Website(s):

http://www.omg.org/spec/BPMN/

http://www.workflowpatterns.com/evaluations/standard/bpmn.php

Organization(s):

Object Management Group (OMG)

Brief Description (extracted from main references):

The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Another goal, but no less important, is to ensure that XML languages designed for the execution of business processes, such as WSBPEL (Web Services Business Process Execution Language), can be visualized with a business-oriented notation. This International Standard represents the amalgamation of best practices within the business modeling community to define the notation and semantics of Collaboration diagrams, Process diagrams, and Choreography diagrams **[OMG13]**.

Domain Type	Domain-Specific
Domains	Any (for instance Banking, Insurance and IT & IS)
Application Area	Business Process
Abstract Level	PIM

Abstract Syntax

#Views	1	(includes 3 types of sub-models: Processes, Choreography and Collaboration)

Business Process Diagram (single view) **[OMG13, p. 69]**

#Classifiers	29

#Classes	#Properties	#Elements	#Enumerations	#UseCases
0	0	29	0	0
Others				
#EnumerationLiterals	#Actors	#Interfaces	#Operations	
0	0	0	0	

Elements: Start Event, Intermediate Event, End Event, Fork, Join, Activity, Gateway, Control Type Gateway, Data Object, Data Input, Data Output, Data Store, Pool, Lane, Group, Text Annotation, Task, Loop, Activity Loop, Sequence Flow Loop, Multi-Instance, Transaction, Compensation, Message, Decision, Exclusive Decision, Inclusive Decision, Event-Based Decision, Merging Decision **[OMG13, pp. 32, 55-69, 240-246]**

#Relationships	11

#Associations	#Generalizations	#Realizations	#Compositions	#Dependencies
11	0	0	0	0

Associations: Sequence Flow, Message Flow, Association (Basic BPMN Modeling Elements), Conditional Sequence Flow, Default Sequence Flow, Uncontrolled Sequence Flow, Normal Sequence Flow, Exception Sequence Flow, Compensation Association, Data Input Association, Data Output Association (Extended BPMN Modeling Elements) **[OMG13, pp. 32, 55-69]**
Generalizations: —.
Realizations: —.
Compositions: —.
Dependencies: —.

Concrete Syntax	
Representation Type	Graphical
Supporting Mechanism	MOF

Notes:

Specific Properties	
Executability	In order to modeling tools be able to emit executable models and to be able to add Data Types, Expressions and service operations, OMG has implemented some technical restrictions: · Data type definition language MUST be XML Schema. · Service Interface definition language MUST be WSDL. · Data access language MUST be XPath. Private Business Process models can be executable or non-executable, depending on boolean attirbute "isExecutable" and Activities are executable elements within a BPMN Process **[OMG13, pp. 36, 50, 176-179]**.
List of Actions	(N/A)
Pattern Usage	**Control-Flow:** Sequence, Parallel Split, Synchronization, Exclusive Choice, Simple Merge, Multi-Choice, Structured Synchronizing Merge, Multi-Merge, Structured Discriminator, Arbitrary Cycles, Implicit Termination, Multiple Instances without Synchronization, Multiple Instances with a Priori Design-Time Knowledge, Multiple Instances with a Priori Run-Time Knowledge, Deferred Choice, Interleaved Parallel Routing, Cancel Activity, Cancel Case, Structured Loop, Persistent Trigger, Cancel Multiple Instance Activity, Cancelling Discriminator, Generalised AND-Join, Acyclic Synchronizing Merge, General Synchronizing Merge, Thread Merge, Thread Split, Explicit Termination. **Data:** Task Data, Block Data, Case Data, Task to Task, Block Task to SubWorkflow Decomposition, SubWorkflow Decomposition to Block Task, Task to Environment - Push-Oriented, Task to Environment - Pull-Oriented, Environment to Task - Push-Oriented, Environment to Task - Pull-Oriented, Data Transfer by Value - Incoming, Data Transfer by Value - Outgoing, Data Transfer by Reference - With Lock, Task Precondition - Data Existence, Task Postcondition - Data Existence, Event-Based Task Trigger, Data-Based Task Trigger, Data-Based Routing

		Trigger, Data-Based Task Trigger, Data-Based Routing. **Resource:** Direct Allocation, Role-Based Allocation, Automatic Execution, Distribution by Allocation - Single Resource, Distribution on Enablement, Commencement on Creation, Chained Execution, Simultaneous Execution. **Exception Handling:** All 32.
Tool Support	Validation	The additional attributes that may extend standard BPMN elements also need to have valid BPMN Core. The "implementation" attribute, which is present in Service Task, Send Task, Receive Task, Business Rule Task and User Task models, must have one of these values: "##unspecified" or "##WebService". Messages exchanged within a Conversation are validated using CorrelationKeys and CorrelationProperties. To validate Collections, the attribute "isCollection" has to be set to *true* . In this case, if the actual type is not a collection type, the model is considered invalid. It is possible to declare the same executions and performances validation for one Process as applicable to another Processes, using Process attribute "supports" **[OMG13, pp. 72, 89, 102, 120, 176, 187-194]**.
	Model 2 Model Transformations	Yes. BPMN's Diagram (Interchange) Definition provides a basis for modeling and interchanging graphical notations, specifically node and edge style diagrams as found in BPMN, UML and SysML **[OMG13, p. 511]**. It has also been studied the possibility to transform BPMN models into Petri Nets models and then mapping them to BORM models **[Tum15]**.
	Model 2 Text Transformations	Yes. It can be mapped to WS-BPEL and XML **[OMG13, p.50]**. It has been studied the possibility to transform BPMN models into Finite State Machine or Mealy and Moore models, after having them also in Petri Nets format **[Tum15]**.
Modularity		(N/A)
Other Features		There is also BPMN DI (Diagram Interchange), a package with BPMN meta-classes that allows BPMN models' interoperability between different tools **[OMG13, p.397].** XSLT transformations allow inter-changing model formats between XSD and XMI **[OMG13, pp. 12, 507].**

Quality Properties

(Alternative Rating: 1 [Very Low], 2 [Low], 3 [Medium], 4 [High], 5 [Very High])

(property evaluation based on evaluator's perception)

Usability	Understandability		(N/A)
	Learnability	1. XP with other MLs	(N/A)
		2. Replacing MLs	(N/A)
	Operability		(N/A)
	Attractiveness	1. Dealing with ML's Concepts knowing the Domain	(N/A)
		2. Using ML's Notation and Supporting Mechanism	(N/A)
	Adaptability	1. Deal with concepts	(N/A)
		2. ML's programs	(N/A)
	User Satisfaction	1. Confort with ML	(N/A)
		2. This ML versus PLs	(N/A)

Maintenance	Stability	(N/A)
	Changeability	(N/A)
	Consistency	(N/A)
	Reusability	4 [High]. It is possible to copy data between graphical elements, using the same ItemDefinition or a DataAssociation with a transformation Expression **[OMG13, pp. 12, 250, 507]**.
	Extensibility	3 [Medium]. It includes an Extension Class composed by four elements, that allows extending standard BPMN elements with additional attributes, such as Artifacts. Still, they need to have valid BPMN Core, be semantically compatible with any BPMN element and extended Diagrams should keep the basic look-and-feel to maintain easy understanding, which means that Events, Activities and Gateways must not be altered **[OMG13, pp. 72, 85].**

Interoperability	Compatibility - Compatible Applications	Number	>6 (58)
		List of Items	ActiveVOS, Activiti Modeler, ADONIS, Agiles BPMS & ECM, Altova UModel, ARCWAY Cockpit, ARIS Express, AuraPortal, Axon.ivy Designer, Bizagi BPM Suite, Bizagi Process Modeler, BiZZdesign Architect, Bonita BPM, Borland Together, BPMN Visio Modeler, BPMN Web Modeler, Camunda Modeler, Cubetto, Cubetto Toolset, Eclipse BPMN2 Modeler, Enterprise Architect, Genexus WorkFlow, GenMyModel, HP Process Automation, IBM BlueWorks Live, IBM Process Designer, IBM Rational System Architect, iGrafx Flowcharter, iGrafx Process, INNOVATOR for Business Analysts, Intellileap Solutions, IYOPRO, jBPM, jBPMN, Logizian, LucidChart, MagicDraw, Microsoft Visio 2013, Modelio, OmniGraffle, Pega Systems, Process Modeler for Microsoft Visio, process4.biz BPM, ProcessCraft, QPR ProcessDesigner, QUAM, RunaWFE, SemTalk, Signavio Process Editor, Software Ideas Modeler, Stages, SYDLE SEED Community, TIBCO ActiveMatrix, Triaster, Visible Analyst, W4 BPMN+, Yaoqiang BPMN Editor and yEd.
	Compatibility - Integration Mechanisms	Number	1-5 (2)
		List of Items	Relationship Types and Diagram Interchange package **[OMG13, pp. 44, 90, 517-527]**
	Model Format		XML Schema

Comprehension	Communication	(N/A)
	Expressiveness	(N/A)

Overall Quality

(N/A)

DEMO (Process Model)

Last Version	3.0	Date	2015

Main Reference(s):

[Die06] J. L. G. Dietz, *Enterprise Ontology - Theory and Methodology* , Springer (2006)

Complementary Reference(s):

[Die13] J. L. G. Dietz, *Red Garden Gnomes Don't Exist - version 3.1* , Sapio.nl (2013)

[Die13a] J. L. G. Dietz, *The Essence of Organisation - version 2.0* , Sapio.nl (2013)

[AST10] D. Aveiro, A.R. Silva and J. Tribolet, *Extending the Design and Engineering Methodology for Organizations with the Generation Operationalization and Discontinuation Organization* , DESRIST 2010, LNCS 6105, Springer-Verlag, pp. 226–241, 2010

[DB99] J. L. G. Dietz and J. Barjis, *Supporting the DEMO Methodology with a Business Oriented Petri Net* , Proceedings of the International Workshop EMMSAD, 1999

Website(s):

http://www.ee-institute.org/

http://ciaonetwork.org/

Organization(s):

Technische Universiteit Delft, Enterprise Engineering Institute

Brief Description (extracted from main references):

DEMO (Design and Engineering Methodology for Organisations) is the leading methodology in the new discipline of Enterprise Engineering (EE). The theory of DEMO is that this social interaction takes place in universal patterns, called transactions. Business processes become clear tree structures of transactions, instead of mind-bending railroad yards.
ICT applications support people, they do not take over responsibility. The essence of every organisation is that it consists of a network of transactions and actors (employees with authority and responsibility), completely independent of any implementation.
This essence is captured in four integrated models: the Construction Model (actors and transactions), the Process Model (business events and business processes), the Fact Model (business objects and business facts) and the Action Model (business rules and work instructions).
Because these models are formalised, ICT applications can directly be generated from them, and the behavior of organisations can be studied through simulation.

Domain Type	General-Purpose
Domains	Any (for instance Banking, Retail and Insurance)
Application Area	Enterprise Architecture
Abstract Level	PIM

Abstract Syntax

#Views	2

Process Structure Diagram, Transaction Process Diagram **[Die13]**.

#Classifiers	6

(formally referred as "Facts")

#Classes	#Properties	#Elements	#Enumerations	#UseCases
0	1	5	0	0
Others				
#EnumerationLiterals	#Actors	#Interfaces	#Operations	
0	0	0	0	

Classes: —.
Properties: Attribute **[Die13]**.
Elements: Class, Type, Property, Actor Role, Stakeholder **[Die13]**.
Enumerations: —.
Use Cases: —.

#Relationships	1

(Transaction is the only "relationship" in a DEMO model)

#Associations	#Generalizations	#Realizations	#Compositions	#Dependencies
1	0	0	0	0

Associations: Transaction **[Die13].**
Generalizations: —.
Realizations: —.
Compositions: —.
Dependencies: —.

Concrete Syntax

Representation Type	Graphical
Supporting Mechanism	eBNF

Notes:

Specific Properties

Executability		(N/A)
List of Actions		(N/A)
Pattern Usage		Transaction (composed by 20 steps) **[Die06] [Die13a]**.
Tool Support	Validation	All compatible tools perform Syntax Analysis on DEMO models and return "warning" or "OK" messages.
	Model 2 Model Transformations	No. It is only theoretically defined, but it is not implemented. An effort was made to approach DEMO to a Business Oriented Petri Net, in order to analyze and simulate business process in an easier way **[DB99]**. Enterprise Engineering Institute researchers are doing this with "reverse engineering", that is, studying a UML or BPMN model, understanding it and design it with DEMO notation and elements.
	Model 2 Text Transformations	No. Enterprise Engineering Institute doesn't see any added value for DEMO with this feature.
Modularity		(N/A)
Other Features		It is possible to add comments on models **[Die06]**.

Quality Properties

(Alternative Rating: 1 [Very Low], 2 [Low], 3 [Medium], 4 [High], 5 [Very High])

(property evaluation based on evaluator's perception)

Usability	Understandability		(N/A)
	Learnability	1. XP with other MLs	(N/A)
		2. Replacing MLs	(N/A)
	Operability		(N/A)
	Attractiveness	1. Dealing with ML's Concepts knowing the Domain	(N/A)
		2. Using ML's Notation and Supporting Mechanism	(N/A)
	Adaptability	1. Deal with concepts	(N/A)
		2. ML's programs	(N/A)
	User Satisfaction	1. Confort with ML	(N/A)
		2. This ML versus PLs	(N/A)

Maintenance	Stability	(N/A)
	Changeability	(N/A)
	Consistency	(N/A)
	Reusability	(N/A)
	Extensibility	3 [Medium]. Since DEMO isn't as flexible as another languages in terms of dynamic organization and model changes, there has been an effort in this way, modeling an organization's activities with DEMO and then extending them to allow activities' generation, operation and discontinuation **[AST10]**.

Interoperability	Compatibility - Compatible Applications	Number	1-5 (5)
		List of Items	Essential Actions Engineers, Formetis, Modelworld, Mprise Tooling and Open Modeling **[DEMO.nl]**.
	Compatibility - Integration Mechanisms	Number	1-5 (4)
		List of Items	DEMOWORLD (for Formetis), J2EE (for Open Modeling), Meetingworks

			(for Essential Actions Engineers) and Xemod (for Mprise Tooling) **[DEMO.nl]**.
	Model Format		Java and XML **[Die06] [Die13]**.

Comprehension	Communication	(N/A)
	Expressiveness	(N/A)

Overall Quality

(N/A)

	Abstract Syntax			Concrete Syntax			Specific Properties			Quality Properties										
	#Views	#Classifiers	#Relationships	Representation Type	Supporting Mechanism	Compatibility	Executability	List of Actions	Modularity	Pattern Usage	Tool Support - Validation	Tool Support - Model to Model Transformations	Tool Support - Model to Text Transformations	Other Features	Reusability	Extensibility	# of Compatible Applications	# of Integration Mechanisms	Communication	Expressiveness
UML Activity Diagram	1	43	3	Graphical	MOF	ArgoUML, Dia, Eclipse UML2 Tools, Enterprise Architect, IBM Rational Rhapsody, Microsoft Visio, Modelio, NetBeans, Papyrus, StarUML, Umbrello UML Modeller and others	The UML Standard Profile defines several standard stereotypes that apply to Artifacts, e.g., «source» or «executable». There are specific types of components that can be deployed as Executable Artifacts and they can be related to a Node using the DeployedArtifact and DeploymentTarget elements [OMG11, pp. 220-229].	(N/A)	(N/A)	Control-Flow: 25. Data: 17. Resource: 8. Exception Handling: 0. Other: 0.	XML uses XSD to validate instances of XML documents [OMG11, p. 739].	Yes. It is possible to transform UML activities, Activity Edge, Call Behaviour Action, Decision Node and other graphical elements into BPMN, using UML 2.0 Diagram Interchange [OMG11, pp. 319-431].	Yes. IBM's Rational Software Architect allows UML-to-Java transformations [ibm.com/developerWorks/]. Also, it is possible to transform UML models into Communicating Sequential Processes (CSPs), using graphs [KHE03].	UML-based Web Engineering allows several model transformations, such as Requirements2Content, Process2Navigation or Functionality2BigPicture, using implementation techniques namely Java, C#, AQG, VIATRA, ATL or QVT [Koc06].	3 [Medium]. A fine-grained, flexible metamodel library is provided that is reused to define the UML metamodel, as well as other architecturally related metamodels, such as the Meta Object Facility (MOF) and the Common Warehouse Metamodel (CWM) [OMG11a, p. 23].	4 [High]. The UML can be extended in two ways: • A new dialect of UML can be defined by using Profiles to customize the language for particular platforms (e.g., J2EE/EJB, .NET/COM+) and domains (e.g., finance, telecommunications, aerospace). • A new language related to UML can be specified by reusing part of the InfrastructureLibrary package and augmenting with appropriate metaclasses and metarelationships. The former case defines a new dialect of UML, while the latter case defines a new member of the UML family of languages [OMG11a, p. 23].	>6 (41)	>6 (7)	(N/A)	(N/A)
BPMN BP Diagram	1	29	11	Graphical	MOF	Agilex BPMS & ECM, BizZdesign Architect, BPMN Visio Modeler, BPMN Web Modeler, Eclipse BPMN2 Modeler, Enterprise Architect, HP Process Automation, IBM BlueWorks Live, IBM Process Designer, IBM Rational System Architect, Microsoft Visio 2013, Process Modeler for Microsoft Visio and others	In order to modeling tools be able to emit executable models and to be able to add Data Types, Expressions and service operations, OMG has implemented some technical restrictions: Data type definition language MUST be XML Schema. Service Interface definition language MUST be WSDL. Data access language MUST be XPath. Private Business Process models can be executable or non-executable, depending on boolean attribute "isExecutable" and Activities are executable elements within a BPMN Process [OMG13, pp. 36, 50, 176-179].	(N/A)	(N/A)	Control-Flow: 28. Data: 18. Resource: 8. Exception Handling: 32. Other: 0.	The additional attributes that may extend standard BPMN elements also need to have valid BPMN Core. The "implementation" attribute, which is present in Service Task, Send Task, Receive Task, Business Rule Task and User Task models, must have one of these values: "##unspecified" or "##WebService". Messages exchanged within a Conversation are validated using CorrelationKeys and CorrelationProperties. To validate Collections, the attribute "isCollection" has to be set to true. In this case, if the actual type is not a collection type, the model is considered invalid. It is possible to declare the same executions and performances validation for one Process as applicable to another Processes, using Process attribute "supports" [OMG13, pp. 72, 89, 102, 120, 176, 187-194].	Yes. BPMN's Diagram (Interchange) Definition provides a basis for modeling and interchanging graphical notations, specifically node and edge style diagrams as found in BPMN, UML and SysML [OMG13, p. 511]. It has also been studied the possibility to transform BPMN models into Petri Nets models and then mapping them to BORM models [Tum15].	Yes. It can be mapped to WS-BPEL and XML [OMG13, p.50]. It has been studied the possibility to transform BPMN models into Finite State Machine or Mealy and Moore models, after having them also in Petri Nets format [Tum15].	There is also BPMN DI (Diagram Interchange), a package with BPMN meta-classes that allows BPMN models' interopeability between different tools [OMG13, p.397]. XSLT transformations allow inter-changing model formats between XSD and XMI [OMG13, pp. 12, 507].	4 [High]. It is possible to copy data between graphical elements, using the same ItemDefinition or a DataAssociation with a transformation Expression [OMG13, pp. 12, 250, 507].	3 [Medium]. It includes an Extension Class composed by four elements, that allows extending standard BPMN elements with additional attributes, such as Artifacts. Still, they need to have valid BPMN Core, be semantically compatible with any BPMN element and extended Diagrams should keep the basic look-and-feel to maintain easy understanding, which means that Events, Activities and Gateways must not be altered [OMG13, pp. 72, 85].	>6 (58)	1-5 (2)	(N/A)	(N/A)
DEMO Process Model	2	6	1	Graphical	eBNF	Essential Actions Engineers, Formetis, Modelworld, Mprise Tooling and Open Modeling [DEMO.nl].	(N/A)	(N/A)	(N/A)	Control-Flow: 0. Data: 0. Resource: 0. Exception Handling: 0. Other: 1.	All compatible tools perform Syntax Analysis on DEMO models and return "warning" or "OK" messages.	No. It is only theoretically defined, but it is not implemented. An effort was made to approach DEMO to a Business Oriented Petri Net, in order to analyze and simulate business process in an easier way [DB99]. Enterprise Engineering Institute researchers are doing this with "reverse engineering", that is, studying a UML or BPMN model, understanding it and design it with DEMO notation and elements.	No. Enterprise Engineering Institute doesn't see any added value for DEMO with this feature.	It is possible to add comments on models [Dek06].	(N/A)	3 [Medium]. Since DEMO isn't as flexible as another languages in terms of dynamic organization and model changes, there has been an effort in this way, modeling an organization's activities with DEMO and then extending them to allow activities' generation, operation and discontinuation [AST10].	1-5 (5)	1-5 (4)	(N/A)	(N/A)

Falta fazer fichas do UML Activity Diagram, BPMN Business Process Diagram e DEMO Interaction Model

Fazer Download da aplicação Gephi, para gerar fórmulas, gráficos e grafos. Qualquer dúvida contactar barao.alexandre@gmail.com

Contactar barao.alexandre@gmail.com para usar Gelphi para criar um grafo de 2 ou + níveis para a minha framework e pô-la no meu website.

#Constructs = #Stereotypes + All Elements - Linking Elements

short_BPMLs_text

	Abstract Syntax			Concrete Syntax			Specific Properties			Quality Properties										
	#Views	#Classifiers	#Relationships	Representation Type	Supporting Mechanism	Compatibility	Executability	List of Actions	Modularity	Pattern Usage	Tool Support - Validation	Tool Support - Model to Model Transformations	Tool Support - Model to Text Transformations	Other Features	Reusability	Extensibility	# of Compatible Applications	# of Integration Mechanisms	Communication	Expressiveness
UML Activity Diagram	1	43	3	Graphical	MOF	ArgoUML, Dia, Eclipse UML2 Tools, Enterprise Architect, IBM Rational Rhapsody, Microsoft Visio, Modelio, NetBeans, Papyrus, StarUML, Umbrello UML Modeller and others				50							41	7	(N/A)	(N/A)
BPMN BP Diagram	1	29	11	Graphical	MOF	Agiles BPMS & ECM, BIZZdesign Architect, BPMN Visio Modeler, BPMN Web Modeler, Eclipse BPMN2 Modeler, Enterprise Architect, HP Process Automation, IBM BlueWorks Live, IBM Process Designer, IBM Rational System Architect, Microsoft Visio 2013, Process Modeler for Microsoft Visio and others				86							58	2	(N/A)	(N/A)
DEMO Process Model	2	6	1	Graphical	eBNF	Essential Actions Engineers, Formetis, Modelworld, Mprise Tooling and Open Modeling [DEMO.nl].				1							5	4	(N/A)	(N/A)
Avg BPMLs	1	26	5	3 Graphical	2 MOF, 1 eBNF			#DIV/0!		46				(N/A)			35	4		
#(Views, Prop	0	17	6							(Ver também que padrões têm em comum e quais são exclusivos de cada BPML)										
#(Views, Prop	1	2	2																	

Falta fazer fichas do UML Activity Diagram e BPMN

Fazer Download da aplicação Gephi, para gerar fórmulas, gráficos e grafos. Qualquer dúvida contactar barao.alexandre@gmail.com

Contactar barao.alexandre@gmail.com para usar Gelphi para criar um grafo de 2 ou + níveis para a minha framework e pô-la no meu website.

short_BPMLs_num