A method to evaluate *quality* of modelling languages based on the Zachman reference taxonomy

Fáber D. Giraldo · Sergio España · William J. Giraldo · Óscar Pastor · John Krogstie

Received: date / Accepted: date

Abstract The Model-Driven Engineering (MDE) paradigm promotes the use of conceptual models in information systems (IS) engineering and research. As engineering products, conceptual models must be of high quality, which applies to both conceptual models and the modelling language used to build them. *Quality* is a growing concern in the MDE field; however, studies such as (Giraldo et al., 2016b; Goulão et al., 2016) demonstrate the divergence in the several approaches that are proposed for addressing this topic. Due to the many challenges, divergences, and trends for quality assessment and assurance in the MDE context, one way to perform a quality evaluation process is to use an approach where the applicability

Fáber D. Giraldo · William J. Giraldo
SINFOCI Research Group
University of Quindío, Colombia
Cra 15 Calle 12N, Armenia Quindío, Postal code: 630004.
Tel.: +57 67359300 ext 908, 995
E-mail: {fdgiraldo, wjgiraldo}@uniquindio.edu.co

Sergio España
Department of Information and Computing Sciences
Utrecht University, The Netherlands
Office: Buys-Ballotgebouw (BBL) 580 — P.O. Box 80.089 — 3508 TB
Tel: +31 616 100 939
E-mail: s.espana@uu.nl

Óscar Pastor · Fáber D. Giraldo
PROS Research Centre
Universitat Politècnica de València
Camino de Vera S/N Valencia 46022, , Spain
Tel.: +34 963877350 (Ext. 77353)
E-mail: {opastor, fdgiraldo}@pros.upv.es

John Krogstie
Department of Computer and Information Science
Norwegian University of Science and Technology, Norway
E-mail: john.krogstie@idi.ntnu.no

and goals of modelling languages (and artifacts) can be compared with respect to the essential principles of the development of IS.

We propose using principles from an IS architecture reference (i.e., the *Zachman* framework) as a *taxonomy* that is applied on the modelling languages used in information system development in order to perform analytic procedures. We also demonstrate that this taxonomy can be considered as a *formal context* for the application of the *Formal Concept Analysis* (FCA) method.

This paper derives formal, methodological, and technological requirements for a modelling language quality evaluation method (MMQEF) with the potential to tackle some of the open MDE quality challenges. In addition, a tool that operationalizes the taxonomic evaluation procedure and the FCA analytic method is also presented. In this work, we discuss how this taxonomy supports analytics that are in modelling languages for quality purposes through its management of the semantics.

Keywords Quality, model-driven engineering, information systems, modelling language evaluation, reference taxonomy, the MMQEF method.

1 Introduction

The design and development of Information Systems (IS) with conceptual models is highly dependent on the cognitive abilities and experience of the people that participate in the modelling tasks. When an analysis is made on a conceptual model, it is difficult to qualify or give an opinion about its associated quality.

One of the main problems here is the *conceptual divergence* in the use and applicability of modelling languages. The cognitive process involved in the modelling act influences their definition and application. This leads to a decoupling between the initial goals of the languages and the real use that final users of the language give to them.

The key behind the analytics process for quality evaluation in Model-Driven Engineering (MDE) is the explicit management of the *semantics* dimension. The Model-Driven Architecture (MDA) guide 2.0 (OMG, 2014) promotes the *semantic analytics of models* as procedures over semantics data for assisting in decision making, monitoring, and quality assessment. These procedures include tasks such as validation, statistics, and metrics.

However, the current state of the model-driven initiatives does not relate the semantics of the modelling languages with the essential modelling features that are expected in an IS development process. The conceptual frameworks about the model-driven paradigm do not clearly answer whether or not any modelling artifact (e.g., models or diagrams) meets the MDE paradigm (Giraldo et al., 2016b). There is an open issue about the derivation and management of data about the semantics that is associated to quality evaluation processes on modelling languages. It is possible to support analytics on quality issues of modelling languages from the semantics data. Examples of issues are the following: suitability, expressiveness, arbitrariness, support for communication, management of traceability, and systematic support for transformations of models.

This paper proposes a method and a tool to support an analytic procedure on modelling languages. It belongs to a *taxonomic evaluation* method which uses a *taxonomy* that is extracted from an IS reference architecture based on the Zachman framework. This framework was chosen because it was one of the first and most precise proposals for a reference architecture for IS, which is recognized by important standards such as the ISO 42010 (ISO/IEC/IEEE, 2011).

The evaluation method is formally supported by the application of the *Formal Concept Analysis* (FCA) approach. We use FCA to explicitly identify relationships between concepts of languages. This approach (by default) manages the semantics of these concepts through a graphical representation in which the semantics is expressed as *distances* and *grouping* between concepts. From this application of FCA, we also derive a tool whose implementation was made on the Eclipse Modelling Project (specifically, the EMF and GEF frameworks). This project is recognized as being one of the most important technical environments for MDE (da Silva, 2015). The automation of the FCA method by a plugin for a native MDE development environment results in an appropriate alternative for the validation of quality in models and modelling languages.

Our work makes the following contributions:

– We discuss the applicability of an IS reference architecture as a *taxonomy* in a MDE quality evaluation procedure. We support it using an ontological validation of the reference taxonomy. The sufficiency of the reference architecture as a *taxonomic theory* is also presented.
– We present a quality tool for managing the semantics and making quality evaluation procedures on modelling languages.
– Two examples of quality evaluation on modelling approaches are described in order to show the applicability of our evaluation method in combination with its associated tool.

The remainder of this paper is structured as follows. Section 2 shows the main features of the taxonomic evaluation method, the FCA method, and its application in the taxonomic evaluation of modelling languages, and also the application of the quality evaluation method with an implemented tool for supporting the management of the semantics data that are obtained from the taxonomic analysis. Section 3 discusses the formal support of the Zachman framework for considering it as a *taxonomy*, and, therefore, performing classification analysis. Section 4 discusses the main features of the quality evaluation method, including its applicability with respect to previous methods for quality evaluation at model-driven levels. Section 5 concludes the paper and outlines future work.

2 The evaluation method

2.1 An example about quality reasoning and evaluation on modelling languages

MDE proposes modelling languages as the new abstraction units. The adoption of MDE approaches has guided the development of a large number of model-driven initiatives. Although MDE emphasizes the use of models as the primary artifacts of an engineering construction process (mostly for software construction), it causes a conceptual divergence in the support of specific views and/or concerns belonging to an IS. This phenomenon is strengthened by the lack of (semantic) support offered by the languages.

One example of conceptual divergence can be a comparison between the UML and BPMN languages for supporting business concerns. UML has been commonly considered the modelling language (by default) for software systems, focusing on functional features of these systems and deriving objects that interact with each other to perform those expected functionalities.

On the other hand, BPMN is one of the most recognized approaches for modelling business processes. Some BPMN platforms (or BPMN suites) support desirable features of modelling languages, such as the execution of models (business process models), the automatic generation of software code, and the native use for domain experts or business analysts.

In the analysis of these two languages, we consider the guidance that was provided in the Unified Process (Kruchten, 2000) for the specific use of UML in the most representative disciplines of Software Engineering. This process was the first methodological prescription that proposes the use of UML in the construction of software systems. It also considers the use of an UML profile to perform *business modelling*, which is the discipline that prescribes a business engineering effort in order to derive software system requirements for software solutions that must fit into an organization. This consideration was formulated before the official release of the BPMN approach.

The comparison of these two modelling languages is performed w.r.t. the applicability of each one in the modelling of IS views, which is independent of the concrete features and advantages that are provided by each language. Each view of an IS could have a concrete modelling support. The *multiple IS views* feature determines the application of multiple modelling languages and also the presence of quality issues that are associated with this IS feature such as suitability, integration, traces, and organization of languages.

2.2 The Zachman framework as the foundational basis of the evaluation method

Taxonomies play an important role in the development of IS projects(Olivé, 2001). In (Laware and Kowalkowski, 2005), the authors describe the business value of taxonomies and ontologies in terms of their relationship with architectural models that support IS practices.

This work is based on a taxonomic analytic procedure that uses *taxonomy* that relies on the Zachman framework (Zachman, 1987)(Sowa and Zachman, 1992). This framework is one of the most relevant reference architectures for Information Systems. Despite its commercial applications in the enterprise architecture field, we focus only on the essential principles that define this framework around the classification of artifacts that belong to an IS in an organizational context. The taxonomy allows us to derive an analytic procedure for evaluating modelling languages.

The framework is a two-dimensional logical structure in the form of a *matrix*. The *rows* represent the *abstraction levels* involved in an IS development process, which move from organizational towards specific computational implementations. The *columns* are philosophical questions (*what, why, how, where, when, who*), which are used to understand the role of the modelling artifacts that are presented in an IS project with respect to the viewpoints. The combinations of abstraction levels and questions generate *cells* in the matrix. Each cell is unique (i.e., each cell has only a scope), and each cell is independent of the others.

The taxonomy establishes seven rules (Sowa and Zachman, 1992) in order to perform the classification. These are as follows:

R1 The columns do not have any specific order.
R2 Each column has a basic (*simple*) model. This implies that there is an *essential concept* for each column that answers the *question* of its associated column. The basic model constitutes the generic metamodel for any column.
R3 The basic model of each column is unique.
R4 Each row represents a perspective that is unique and different.
R5 Each cell is unique.
R6 The integration of the models of the cells in a row constitutes a *complete model* of this row.
R7 The logic of the framework is recursive (i.e., the essential models can explain themselves), and each cell could be analyzed with the use of the entire taxonomy.

The two-dimensional structure of the taxonomy gives a set of information that is useful and valuable for modelling and reasoning about any phenomena inside the scope of IS (Smith, 2013). The rules for classification guarantee the consistency for any IS modelling activity.

The taxonomy recognizes the presence of different approaches and graphic representations (i.e., modelling languages) that are appropriate for the cells, but it also recognizes that they are not completely adequate due to the different purposes that are addressed by each cell. For this reason, any attempt to fix this insufficiency without a systematic procedure (e.g., to arbitrarily join the formalisms of some cells) could lead to serious problems in the posterior design of the IS design; also, a sub-optimization of formalisms could be evident (Sowa and Zachman, 1992).

2.3 The MMQEF method

Figure 1 summarizes the classification of artifacts of modelling languages in the Zachman taxonomy. The information that comes from modelling languages (such as semantic constructs) and modelling elements (e.g., data derived from diagrams) is classified with respect to the taxonomy structure depending on the purpose (or goal) of the modelling languages that is perceived by the analyst, and the information of the modelling languages that can be captured for the cells of the taxonomy.

The classification activity with the reference taxonomy and the derived analytic procedure that were described in Figure 1 constitute the *Multiple Modelling Quality Evaluation Framework* method (MMQEF). It is a methodological and technological framework for evaluating quality issues in modelling languages by the application of a taxonomic analysis. This method helps to find quality issues in modelling languages according to their support for specific concerns and phenomena inside an IS.

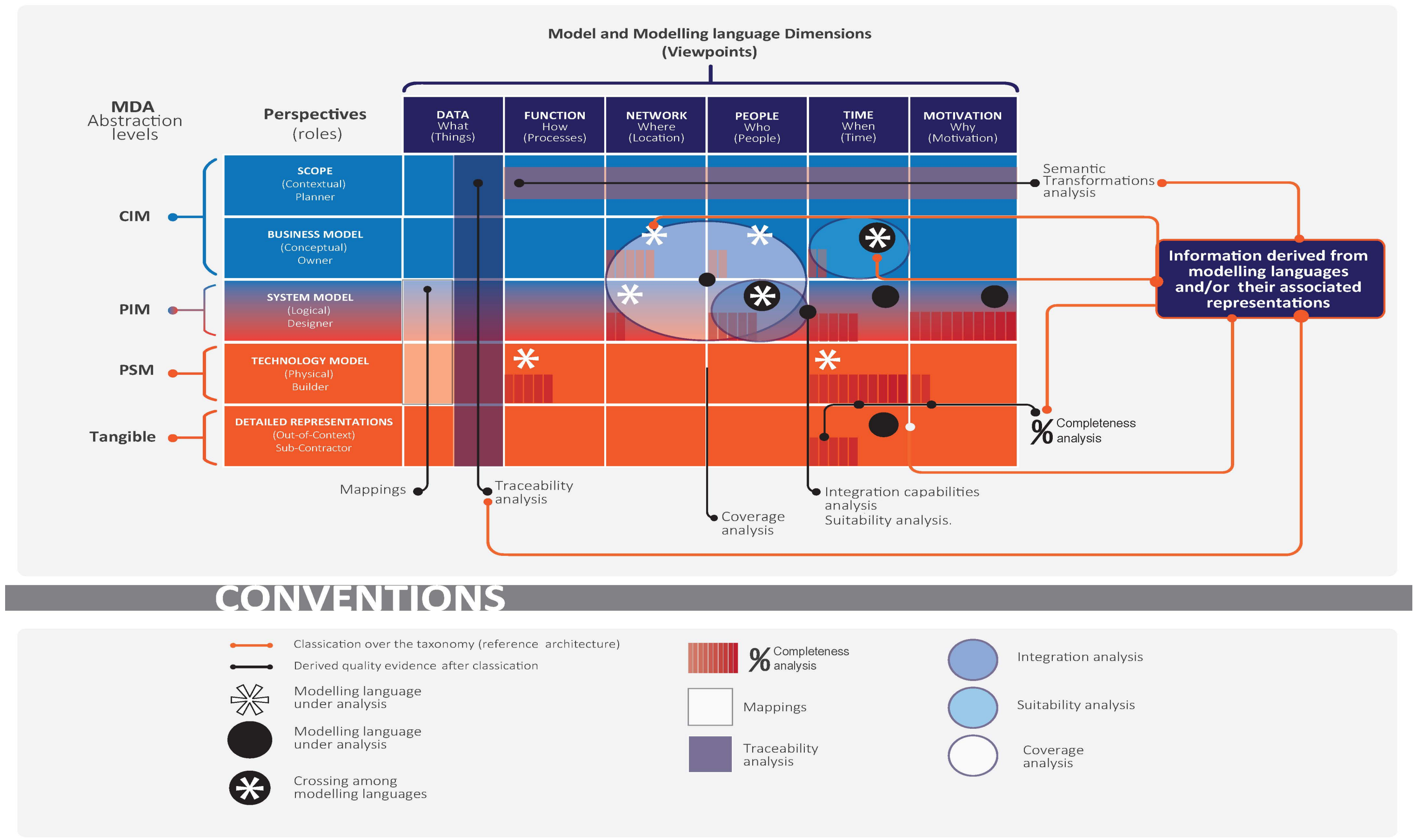

Model and Modelling language Dimensions (Viewpoints)
MDA Abstraction levels
Perspectives (roles)
DATA What (Things)
FUNCTION How (Processes)
NETWORK Where (Location)
PEOPLE Who (People)
TIME When (Time)
MOTIVATION Why (Motivation)
CIM
PIM
PSM
Tangible
SCOPE (Contextual) Planner
BUSINESS MODEL (Conceptual) Owner
SYSTEM MODEL (Logical) Designer
TECHNOLOGY MODEL (Physical) Builder
DETAILED REPRESENTATIONS (Out-of-Context) Sub-Contractor
Semantic Transformations analysis
Information derived from modelling languages and/or their associated representations
Completeness analysis
Mappings
Traceability analysis
Coverage analysis
Integration capabilities analysis
Suitability analysis.
CONVENTIONS
Classication over the taxonomy (reference architecture)
Derived quality evidence after classication
Modelling language under analysis
Modelling language under analysis
Crossing among modelling languages
Completeness analysis
Mappings
Traceability analysis
Integration analysis
Suitability analysis
Coverage analysis

Fig. 1 Summary of the support provided by the reference taxonomy for quality evaluation in MDE contexts.

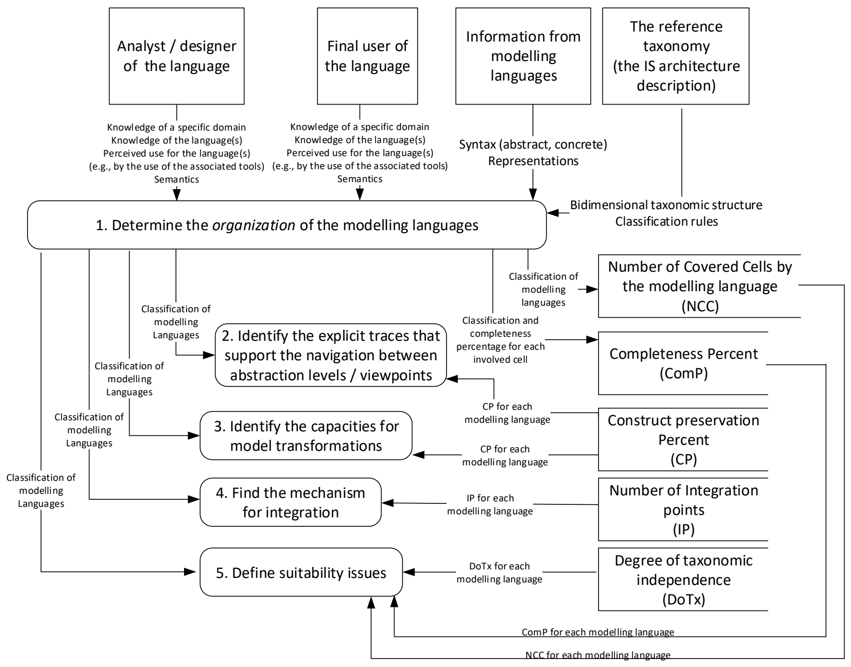

Analyst / designer of the language
Final user of the language
Information from modelling languages
The reference taxonomy (the IS architecture description)
Knowledge of a specific domain
Knowledge of the language(s)
Perceived use for the language(s)
(e.g., by the use of the associated tools)
Semantics
Knowledge of a specific domain
Knowledge of the language(s)
Perceived use for the language(s)
(e.g., by the use of the associated tools)
Semantics
Syntax (abstract, concrete)
Representations
Bidimensional taxonomic structure
Classification rules
1. Determine the *organization* of the modelling languages
Classification of modelling languages
Number of Covered Cells by the modelling language (NCC)
Classification of modelling Languages
2. Identify the explicit traces that support the navigation between abstraction levels / viewpoints
Classification and completeness percentage for each involved cell
Completeness Percent (ComP)
Classification of modelling Languages
CP for each modelling language
Construct preservation Percent (CP)
Classification of modelling Languages
3. Identify the capacities for model transformations
CP for each modelling language
Classification of modelling Languages
4. Find the mechanism for integration
IP for each modelling language
Number of Integration points (IP)
5. Define suitability issues
DoTx for each modelling language
Degree of taxonomic independence (DoTx)
ComP for each modelling language
NCC for each modelling language

Fig. 2 Summary of the MMQEF method.

Quality for modelling languages in MDE could be defined as the degree to which a given modelling language (with its artifacts) meets an IS concern, considering its location inside an abstraction level with a clear purpose (or viewpoint) and an explicit traceability for deriving technical implementations (as part of an IS development process). MMQEF defines the quality of a modelling language as the degree of its fulfillment with essential principles of IS that are defined in the Zachman reference architecture for IS.

As an evaluation method, MMQEF defines activities in order to derive quality analytics based on the classification applied to modelling languages. Figure 2 summarizes the MMQEF method using a *Data-Flow Diagram* (DFD) level 1 notation to show the main evaluation activities in accordance with the support previously presented in Figure 1.

MMQEF derives analytic procedures that support the detection of quality issues in modelling languages, such as the suitability of modelling languages, traces between abstraction levels, specification for model transformations, and integration between modelling proposals. The most important quality question that is managed by the taxonomy is the support of the *essential modelling* of IS concepts and their associated abstraction level. This is done by contrasting the modelling artifacts with the minimum information that is expected in each cell of the taxonomy.

2.4 Main features of MMQEF

Although the modelling levels that are defined in the taxonomy are closer to the levels of the MDA specification (i.e., the Computation-Independent Model or CIM, the Platform-Independent Model or PIM, and the Platform-Specific Model or PSM), the taxonomy is a neutral mechanism for reasoning about models used in the development of IS.

Thus, even a model-driven approach that is not in compliance with MDA can be classified by the taxonomy as long as it considers at least one of the following descriptions: the specification of the system (in terms of the *what* or *how* of the granularity levels) and/or the description of the surrounding *situation* in which the system will be used (i.e., the *why* of the system). The taxonomy considers the *situation* as an organizational perspective.

Through the application of taxonomic analysis, some key features of MMQEF are derived. One feature is the management of the model transformations as a *controlled process*, i.e., the transformations and mapping of models take place as a direct consequence of the addition of information in accordance with the interaction between abstraction levels and questions.

The *model mapping* feature (which is mentioned in the first version of the MDA guide (OMG, 2003)) occurs as a result of the structural changes in the models that cross from higher to lower abstractions. Information that comes from the Computation-Independent Model (CIM) level are enriched with constraints that are associated to lower levels so that it is possible to get enough information for the implementation of higher models in a technical (computational) environment.

Similarly, *transformations* between the information of two different columns must be sufficiently justified in order to support the derivation of models from different essential properties (e.g., *time-location*, *data-process*).

Among the main features in the design of modelling languages, there must be an explicit rationale about *why* and *how* information from a column can derive/generate/support information for another column. In addition, the evidence of traceability can be obtained from the information classified in the taxonomy.

The classification act requires the explicit rationale of the *technical issues implementation* of the model artifacts under consideration. The Platform-specific models (PSM) row of Figure 1 expresses models that are transformed to any specific technological platform: programming languages, development environments, supporting platforms (frameworks, engines, APIs), and hardware and network configurations.

Therefore, when the classification is performed, the associated technical details must be considered to ensure the full functional implementation of the IS (i.e., the transformation of the models to executable artifacts on computational platforms). This is the implication of the *abstraction level zero* that the reference taxonomy defines in the lower row. This *functional implementation* feature must be considered at some point to indicate the feasibility of the modelling effort from a computational perspective.

To make the development of the *system* and *situations* descriptions compatible with current methodologies for software and system construction, descriptions can be mapped to specific tasks and artifacts associated to roles that participate in the development process of an IS; e.g., the organizational information can be grouped in a business analyst or business expert role, the *what* description can be grouped in the system analyst-designer role, and the *how* description can be grouped in the system designer and developer role.

Since the taxonomy focuses on the use of models and the key importance that models have in projects developed under the model-driven paradigm, descriptions must have models (resulting from modelling languages) to address the concerns involved in an IS.

The taxonomic analysis supports *quality inferences* for modelling languages (with their model elements[1]). This distinction is important because quality evidence come from different sources regarding the type of artifact under analysis; for modelling languages, quality is based on their properties; and for modelling elements, quality is based on their derived and recognized use.

2.5 The FCA method and its support for the taxonomic analysis of MMQEF

The *Formal Concept Analysis* (FCA) approach is a mathematical method for data analytics, representation of knowledge, and information management (Priss, 2006). It is based on the ordered set theory. FCA provides a natural method for defining concepts in a model so that these concepts are associated to others through shared features. The main goal of the FCA method is that *concepts* are all the pairs of *objects* and *attributes* that have a mutual dependence.

The foundational support of the *formal concept* is that a *concept* is determined by its *extension* (i.e., the collection of objects that are covered by this concept) and its *intention* (the set of properties or attributes that are included in this concept). Therefore, a *formal concept* is conformed in an incidence relationship called *formal context* between a set of objects and a set of attributes through any closing operator.

According to (Wolff, 1993), the mutual dependence between objects and attributes is defined as a *formal context* (a partially ordered triplet) $K = (O, A, I)$ where: O represents a formal set of objects (e.g., the classes in a software model); A is a set of attributes that may or may not have objects; and I is a relation of incidences that shows the association between an object and an attribute; I is defined as $I \subseteq O \times A$. I is expressed as a binary relation *(o,a)*, it is interpreted as *the o-object has the a-attribute* (Ganter and Wille, 1999).

A *formal context* can be represented as a matrix with crossings, where the rows are O, the columns are A, and the incident relation I is a series of crossings between rows and columns. The FCA verifies the closeness between concepts (i.e., the *semantic distance*) by operations of containment and overlapping depending on the incidences that were found. This analysis receives a matrix of objects (rows) and attributes (columns) as input. If a specific feature is identified, a mark is assigned in the corresponding cell of the attribute that possesses it. The output of the analysis is a *concept lattice* (connected graph), where nodes are the concepts and attributes under analysis, and the lines represent the semantic distance between them. Figure 3 summarizes the FCA method.

In order to operationalize the quality evaluation procedure mentioned in Section 2.3, we consider the reference taxonomy as a *formal context* where

[1] These refer to the specific elements that appear with the use of a modelling language, according to the definition presented in (Object and Reference Model Subcommittee of the Architecture Board, 2005).

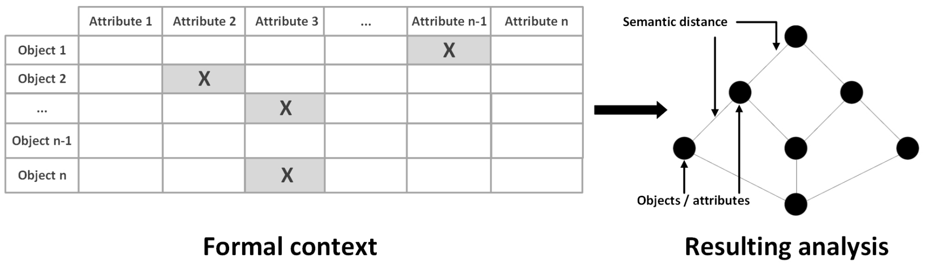


Fig. 3 Summary of the FCA method (with figures taken from (Wolff, 1993)).

$O := \{scope/contextual, business\_model/conceptual, system\_model/logical,$
$technological\_model/physical, detailed\_representation/out - of - context\}$
$A := \{data/what, function/how, network/where, people/who, time/when,$
$motivation/why\}$

and the incidences $(o, a) \in I$ are derived when the information of modelling languages is classified using the taxonomic structure. Thus, the FCA method processes the grammar constructs provided by the modelling languages that are involved in an IS development process (semantic constructs, diagrams, etc.), generating a connected graph or *concept lattice* as the output, where the relations between elements are described. The resulting lattice derives inferences about the application of modelling languages. In the classification of the modelling artifacts over the reference taxonomy, some types of association of concepts are identified:

- Association 1: $for\ a \in A \mid (o, a) \in Ifor\ only\ one\ o \in O \wedge O \subseteq K$.
- Association 2: $for\ A' \subseteq A \mid (o, a) \in Ifor\ O' \subseteq O$.
- Association 3: $for\ O' \subseteq O \mid (o, a) \in Ifor\ A' \subseteq A$.

Due to these associations, the rules of the taxonomy (Section 2.2) must be contrasted with the rules of the FCA method for analyzing concepts in order to harmonize the two set of rules for generating concrete analyses over the modelling artifacts. This was achieved by modifying the procedure that generates concept lattices, so that the taxonomic independence that is defined for each row of the reference taxonomy (R4 and R6) could be not combined during the FCA parsing procedure. Due to associations 2 and 3, the semantic distance of objects and attributes will be linked to the same node in the resulting concept lattice.

In addition, to meet R5 of the taxonomy (*each cell is unique*), a *contribution value* was established to indicate the *percentage of completeness* that the modelling artifact contributes in answering the coverage of the abstraction level-philosophical question pair for a specific cell. The contribution value is mainly used in cases when two or more modelling languages support one cell. This value is based on the opinion or judgment of experts (i.e., the modelling language analysts) who define the degree to which modelling artifacts (e.g., model elements, diagrams, or meta-elements) support a cell based on the specific purpose of that artifact.

2.5.1 The EMAT Tool

EMAT (*EMF Modelling Analytics Tool*) is a software tool that is implemented to support the evaluation of quality in modelling languages through the taxonomic analysis proposed in the MMQEF method. EMAT is a plugin for the Eclipse Modelling Project[2], specifically the Eclipse Modelling Framework (EMF). The plugin supports the analytic procedure with the reference taxonomy presented in Section 2.3 and the FCA method for the management of the semantics data derived from the taxonomic analysis.

We identify a common question about why it is necessary to develop another FCA tool, taking into account previous FCA tools (e.g., Concept Explorer[3], and Toscana[4]). These types of FCA tools support analysis of objects/attributes in generic contexts.

EMAT is not another FCA tool. It focuses on providing the required technical support to evaluate modelling languages through taxonomic analysis and operationalizing the generation of lattices that contain objects and attributes derived from information of the modelling artifacts. EMAT implements the FCA support for the taxonomic analysis previously mentioned. This FCA analysis considers the application of the seven rules of the reference taxonomy. The output of EMAT is not only a drawing of a connected graph, it is also a *conceptual model* of the semantic closeness among objects/attributes of modelling languages. Quality inferences can be deduced and also automated from this model.

Unlike traditional FCA tools, EMAT works within a MDE technical environment. As a further consequence, we hope this integration promotes the reduction of the subjectivity criteria that are traditionally associated to the quality evaluation processes. EMAT allows model-driven practitioners to perform reasoning over shared semantic lattices.

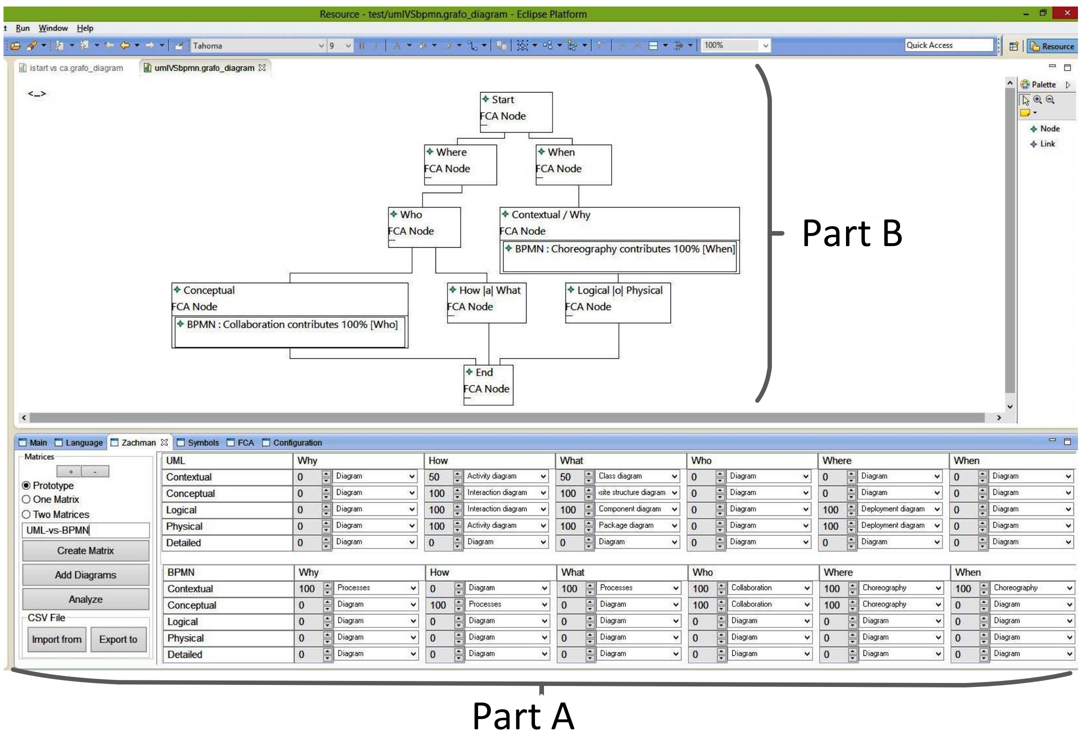
In the current version of EMAT, the resulting lattices are rendered directly over the work area of EMF. However, lattices are not only a visual output with graphical information of the semantics, they are also *conceptual models* that contain data about the semantic closeness of modelling languages. In this way, quality inferences (that are derived from the taxonomic analysis of MMQEF) are formally supported and potentially automatable by the application of some query analysis and/or formal methods.

2.5.2 The taxonomic evaluation procedure using EMAT

EMAT offers a view into the Eclipse working area to classify the elements of the modelling languages under analysis in each cell of the taxonomy. For each language under evaluation, the analyst (i.e., the role that represents the designer of the language or the final user of the language) can indicate whether the elements of the language fit in a cell with its associated *coverage value*; it is a percentage between 0-100 that indicates the degree of coverage that these elements contribute to answering the specific question in the considered cell. For each cell, two input values are generated *(modelling_language_element, percentage)* for the FCA analysis; these values mark the *incidence* relationship in the matrix.

In the current version of the tool, a graphical interface was implemented to evaluate one or more modelling languages versus the reference taxonomy, i.e., a FCA analysis is performed by overlapping the matrixes (each one with the specific

[2] https://eclipse.org/modeling/
[3] http://conexp.sourceforge.net/
[4] http://toscanaj.sourceforge.net/



Matrices

[✓ Prototype] One Matrix Two Matrices

UML-vs-BPMN

CSV File

UML	Why		How		What		Who		Where		When	
Contextual	0	Diagram	50	Activity diagram	50	Class diagram	0	Diagram	0	Diagram	0	Diagram
Conceptual	0	Diagram	100	Interaction diagram	100	ısite structure diagram	0	Diagram	0	Diagram	0	Diagram
Logical	0	Diagram	100	Interaction diagram	100	Component diagram	0	Diagram	100	Deployment diagram	0	Diagram
Physical	0	Diagram	100	Activity diagram	100	Package diagram	0	Diagram	100	Deployment diagram	0	Diagram
Detailed	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram

BPMN	Why		How		What		Who		Where		When	
Contextual	100	Processes	0	Diagram	100	Processes	100	Collaboration	100	Choreography	100	Choreography
Conceptual	0	Diagram	100	Processes	0	Diagram	100	Collaboration	100	Choreography	0	Diagram
Logical	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram
Physical	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram
Detailed	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram	0	Diagram

Fig. 4 Example of the taxonomic evaluation for the UML and BPMN modelling languages performed in the EMAT tool.

set of incidences for each modelling language under analysis). This evaluation produces relationships of containment in the resulting concept lattice. In this concept lattice, the contributions of each language for each cell are reported, differentiating which of the languages is closer to 100% of the contribution. This is the *completeness analysis* that was depicted in Figure 1.

Figure 4 presents an example of an analysis that was performed over the UML and BPMN modelling languages. For this case, we use the information from the diagrams associated to both languages, indicating the coverage provided by each diagram in each cell of the taxonomic structure (Figure 4, *Part A*). Each diagram has a structure to classify it. Afterwards, a concept lattice (Figure 4, *Part B*) is automatically generated as the result of the application of the FCA method with the modifications for preserving the taxonomic rules.

EMAT allows the configuration of options for the automatic generation of the concept lattice. Because of its mathematical foundations, the FCA properties must be fulfilled. Therefore, EMAT provides a configuration panel where the user can apply the FCA analysis by selecting between two alternatives, either fulfilling the essential modelling of the taxonomy or applying each FCA rule to each matrix. Each alternative generates a connected graph.

Finally, to support the associations reported in Section 2.5, we use three conventions to identify them (Figure 4, *Part A*):

- The / convention represents that the object is the only one that has an attribute (i.e., a single association between an abstraction level with a philosophical question or vice versa).
- The | *obj* | convention indicates that two or more abstractions have the same associated attributes.

  The | *att* | convention defines that two or more questions are associated with one or more abstractions.

Depending on the selected configuration, the FCA analysis can be performed by taking into account R3 of the taxonomy (each column has a basic model), or applying the FCA method to look for the semantic similarities between columns reported from the classification of modelling artifacts.

2.5.3 How should a resulting lattice be interpreted ?

The graphical interpretation of the lattice in EMAT is equivalent to an *ordinal context* (Ganter and Wille, 1999), which provides a hierarchy for analyzing concepts from a set of objects and attributes that compose them. Each concept of the obtained hierarchy represents a set of objects that share the same values or meanings for a certain set of attributes.

The formal concepts are defined as a pair of a set of objects (*extension*) and another set of attributes (*intention*). The *extension* is all the objects that share the given attributes. The *intention* is all the attributes that share some given objects. Formal concepts can be ordered partially due to the containment relationship among their sets of objects and attributes. This order produces a hierarchized system in which *sub-concepts* and *super-concepts* appear. These are visualized as nodes and links.

EMAT applies the notion of *object*, *attribute*, *concept*, *sub-concept*, and *super-concept* to interpret an obtained lattice. This interpretation must be done top-down, from the start to end nodes, and taking into account that the attributes (i.e., the questions in the reference taxonomy) are those nearest to the start node, and the objects (i.e., the abstraction levels in the reference taxonomy) are nearest to the end node.

An object has attributes, and, in turn, an attribute could be in many objects. This produces a hierarchized lattice in which the *sub-concepts* are the highest nodes in the lattice, and the *super-concepts* are the lowest The relations among nodes are binaries. They detect if a node has a link with another node, and if a node is a *super-concept* or a *sub-concept* with respect to another node and its associated position.

An additional consideration is needed in the taxonomic analysis because EMAT provides a *completeness percentage* to indicate the degree of support of a modelling artifact for a specific concept. This percentage is depicted as internal nodes inside a conceptual node.

Figure 5 presents an illustrative lattice that was generated intentionally in EMAT, which has the *Physical*, *Why*, and *When* attributes, and the *Contextual* node that is an object. *Contextual* has some internal nodes that describe the relations between nodes.

Contextual and *When* have a relation of 50% that is covered through the BPMN Business Process diagram. This same relation is covered by the UML Class diagram. Afterwards, a relation among the *Contextual* concept and the *Why* attribute

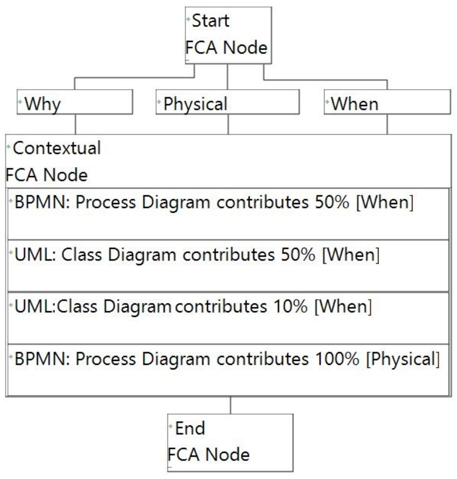

Start
FCA Node
Why
Physical
When
Contextual
FCA Node
BPMN: Process Diagram contributes 50% [When]
UML: Class Diagram contributes 50% [When]
UML:Class Diagram contributes 10% [When]
BPMN: Process Diagram contributes 100% [Physical]
End
FCA Node

Fig. 5 Example of a lattice generated automatically in EMAT.

is covered only by the UML Class diagram with a level of 10%. Finally, there is a relation between *Physical* and *When* with a level of completeness of 100% through the BPMN Business Process diagram. This relation appears inside the *Contextual* node because, in the classification with EMAT, the analyst indicated that these BMPM diagram support both levels (*Contextual* and *Physical*).

From the lattice shown in Figure 5, it can be deduced that BPMN is the most appropriate language for this modelling task because its diagrams cover most of the levels and questions involved in the taxonomic analysis.

2.6 MMQEF in practice

In this section, we present two demonstrations of the application of MMQEF. Modelling scenarios were identified for both examples with more than one modelling language.

2.6.1 Quality analysis of the UML and BPMN modelling languages

Following the example that was described in Section 2.1, Figure 6 presents the application of the MMQEF method to analyze the UML and BPMN modelling languages.

We chose these two languages due to the taxonomic analyses that were previously made separately for the two languages. These previous analyses used the reference taxonomy to classify the diagrams associated to each language. For example, the analysis on UML presented in (Frankel et al., 2003) was proposed from the first release of the MDA specification. The BPMN analysis described in (Zhao et al., 2012) evaluates the suitability of this language for linking the business with the information systems.

Both languages were also evaluated in combination and by using a taxonomy, as reported in (Aagesen and Krogstie, 2015); however, in that work, the selected

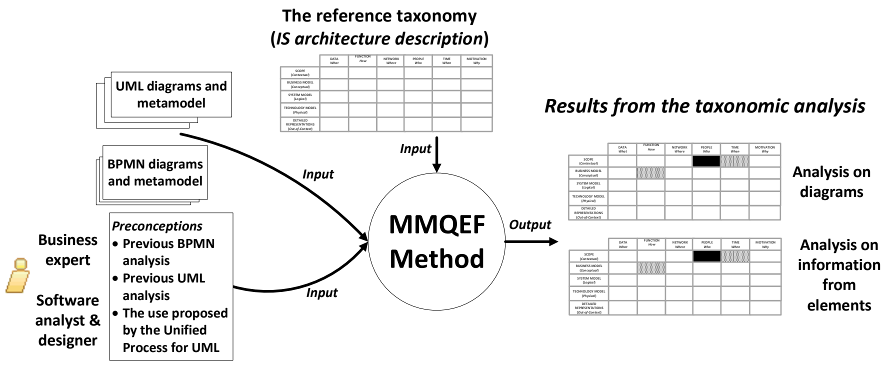

The reference taxonomy
(IS architecture description)
UML diagrams and metamodel
BPMN diagrams and metamodel
Business expert
Software analyst & designer
Preconceptions
• Previous BPMN analysis
• Previous UML analysis
• The use proposed by the Unified Process for UML
Input
Input
Input
MMQEF Method
Output
Results from the taxonomic analysis
Analysis on diagrams
Analysis on information from elements

Fig. 6 Summary of the application of the MMQEF method on UML and BPMN modelling languages.

taxonomy only considers concepts related to the modelling of processes. Therefore, the UML language is limited to the process-modelling support that is commonly associated to the activity diagrams.

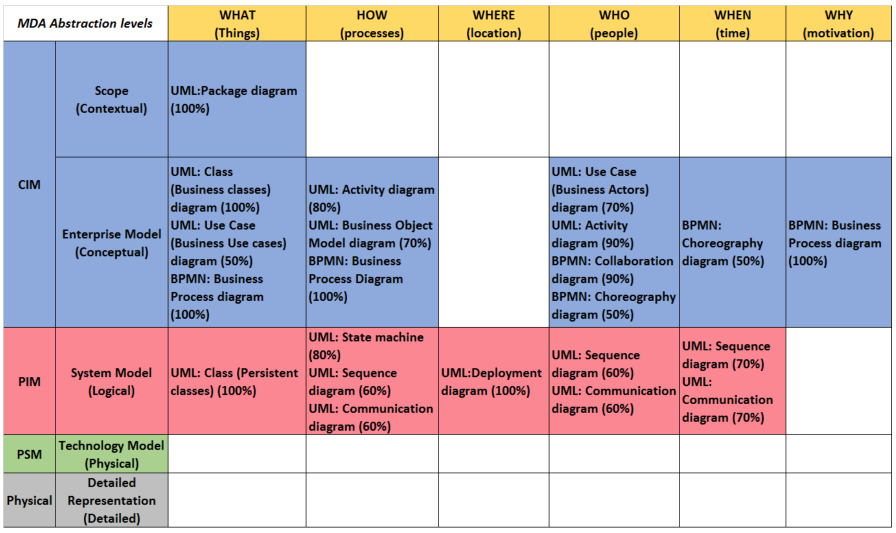
We made the classification of the modelling artifacts from the specifications of both languages provided in the OMG web site[5] and the previous analysis using the reference taxonomy. For this case, Figure 7 presents the classification of the diagrams that are associated to the languages under analysis, and Figure 8 presents the classification of some modelling elements that are relevant to this analysis.

These classifications were made in accordance with the perceived use of the modelling artifacts regarding the taxonomic cells (i.e., artifacts are classified based on the information that each cell can capture from them). The percentages associated to each element are assigned based on the perceived completeness of each modelling artifact for the specific cell. Reported values were assigned by an expert judgment from the perceived information of diagrams (Figure 7) and metamodel elements (Figure 8) associated to both modelling languages. The classification of Figure 8 considers more elements and some different values regarding the precise description that are in the metamodels' specification.

The classification presented in Figure 7 is the input for the EMAT tool. To begin the FCA analysis, EMAT looks for objects and attributes without any relation to incidence. In this case, the *Physical* and *Detailed* objects are eliminated. Then, EMAT looks for objects and attributes that are unique (i.e., those that have only one incidence). In this case, the *Contextual* object only has an incidence with the *What* attribute through the *UML package diagram*. This object is added to the *What* attribute using the */att/* label to indicate that *What* only has the *Contextual* object. This grouping produces the new concept *[What/att/Contextual]*.

In the same way, the *Why* attribute has only one incidence with the *Conceptual* object through the *BPMN Business process* diagram. Therefore, this attribute is added to the *Conceptual* object using the */obj/* label to indicate that the *Conceptual* object only has the *Why* attribute. It produces the new concept

[5] Specifications available at http://www.omg.org/spec/UML/2.5/ (UML), and http://www.omg.org/bpmn/index.htm (BPMN).

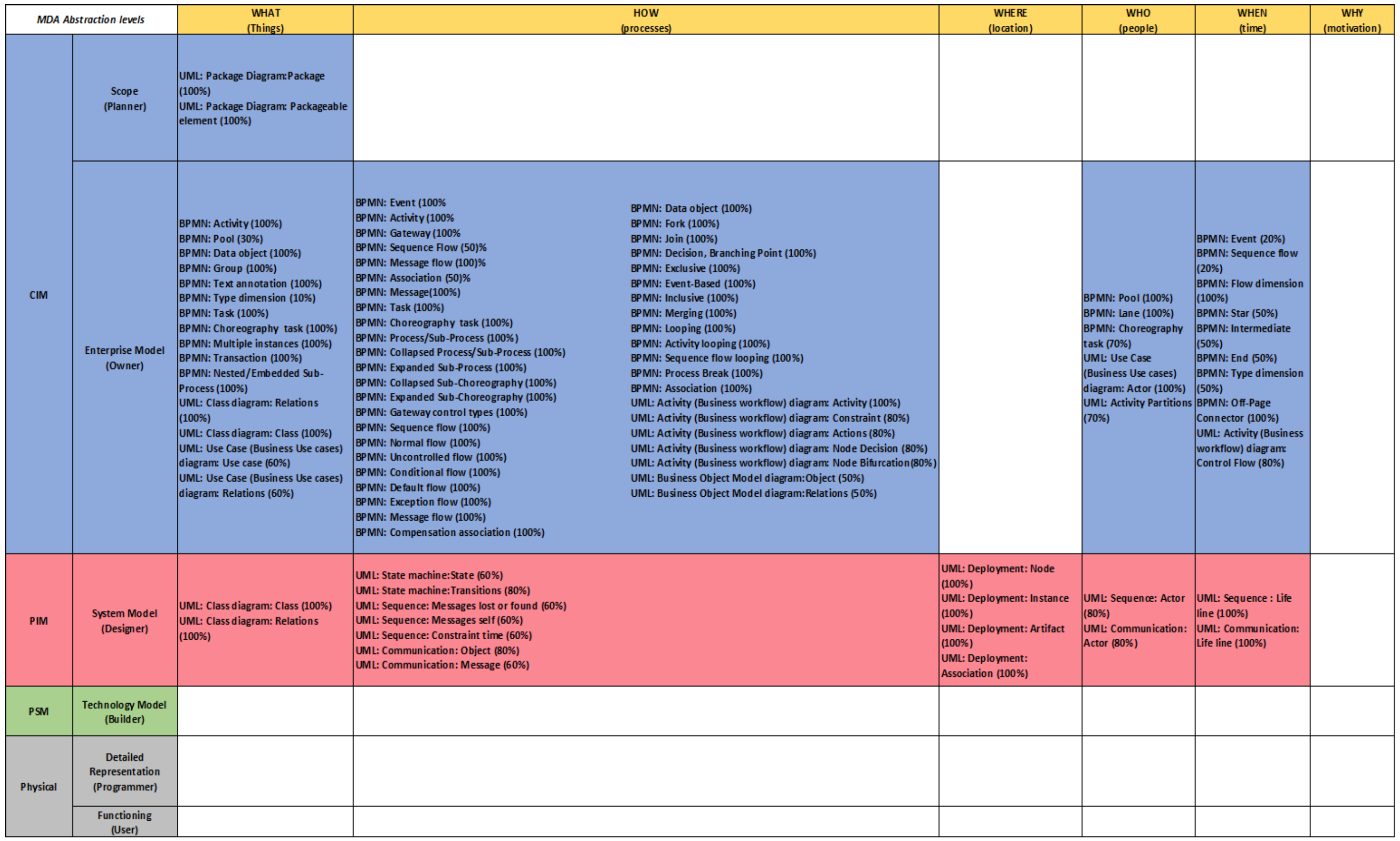


MDA Abstraction levels		WHAT (Things)	HOW (processes)	WHERE (location)	WHO (people)	WHEN (time)	WHY (motivation)
CIM	Scope (Contextual)	UML:Package diagram (100%)					
	Enterprise Model (Conceptual)	UML: Class (Business classes) diagram (100%) UML: Use Case (Business Use cases) diagram (50%) BPMN: Business Process diagram (100%)	UML: Activity diagram (80%) UML: Business Object Model diagram (70%) BPMN: Business Process Diagram (100%)		UML: Use Case (Business Actors) diagram (70%) UML: Activity diagram (90%) BPMN: Collaboration diagram (90%) BPMN: Choreography diagram (50%)	BPMN: Choreography diagram (50%)	BPMN: Business Process diagram (100%)
PIM	System Model (Logical)	UML: Class (Persistent classes) (100%)	UML: State machine (80%) UML: Sequence diagram (60%) UML: Communication diagram (60%)	UML:Deployment diagram (100%)	UML: Sequence diagram (60%) UML: Communication diagram (60%)	UML: Sequence diagram (70%) UML: Communication diagram (70%)	
PSM	Technology Model (Physical)						
Physical	Detailed Representation (Detailed)						

Fig. 7 Taxonomic analysis of the diagrams of the UML and BPMN modelling languages.

[Conceptual/obj/Why]. This same procedure is performed for the *Where* attribute, resulting in the new *[Logical/obj/Where]* concept.

Afterwards, EMAT looks for objects with common attributes and vice versa. This analysis considers the new concepts that were formulated in the previous step. If EMAT determines that two attributes are related to the same objects, it shows the *—att—* label. Finally, EMAT looks for objects that contain other objects for the same (or fewer) attributes. These are the *super-objects*. The same is performed to look for *super-attributes*.



MDA Abstraction levels		WHAT (Things)	HOW (processes)	WHERE (location)	WHO (people)	WHEN (time)	WHY (motivation)
CIM	Scope (Planner)	UML: Package Diagram:Package (100%) UML: Package Diagram: Packageable element (100%)					
	Enterprise Model (Owner)	BPMN: Activity (100%) BPMN: Pool (30%) BPMN: Data object (100%) BPMN: Group (100%) BPMN: Text annotation (100%) BPMN: Type dimension (10%) BPMN: Task (100%) BPMN: Choreography task (100%) BPMN: Multiple instances (100%) BPMN: Transaction (100%) BPMN: Nested/Embedded Sub-Process (100%) UML: Class diagram: Relations (100%) UML: Class diagram: Class (100%) UML: Use Case (Business Use cases) diagram: Use case (60%) UML: Use Case (Business Use cases) diagram: Relations (60%)	BPMN: Event (100% BPMN: Activity (100% BPMN: Gateway (100% BPMN: Sequence Flow (50)% BPMN: Message flow (100)% BPMN: Association (50)% BPMN: Message(100%) BPMN: Task (100%) BPMN: Choreography task (100%) BPMN: Process/Sub-Process (100%) BPMN: Collapsed Process/Sub-Process (100%) BPMN: Expanded Sub-Process (100%) BPMN: Collapsed Sub-Choreography (100%) BPMN: Expanded Sub-Choreography (100%) BPMN: Gateway control types (100%) BPMN: Sequence flow (100%) BPMN: Normal flow (100%) BPMN: Uncontrolled flow (100%) BPMN: Conditional flow (100%) BPMN: Default flow (100%) BPMN: Exception flow (100%) BPMN: Message flow (100%) BPMN: Compensation association (100%) BPMN: Data object (100%) BPMN: Fork (100%) BPMN: Join (100%) BPMN: Decision, Branching Point (100%) BPMN: Exclusive (100%) BPMN: Event-Based (100%) BPMN: Inclusive (100%) BPMN: Merging (100%) BPMN: Looping (100%) BPMN: Activity looping (100%) BPMN: Sequence flow looping (100%) BPMN: Process Break (100%) BPMN: Association (100%) UML: Activity (Business workflow) diagram: Activity (100%) UML: Activity (Business workflow) diagram: Constraint (80%) UML: Activity (Business workflow) diagram: Actions (80%) UML: Activity (Business workflow) diagram: Node Decision (80%) UML: Activity (Business workflow) diagram: Node Bifurcation(80%) UML: Business Object Model diagram:Object (50%) UML: Business Object Model diagram:Relations (50%)		BPMN: Pool (100%) BPMN: Lane (100%) BPMN: Choreography task (70%) UML: Use Case (Business Use cases) diagram: Actor (100%) UML: Activity Partitions (70%)	BPMN: Event (20%) BPMN: Sequence flow (20%) BPMN: Flow dimension (100%) BPMN: Star (50%) BPMN: Intermediate (50%) BPMN: End (50%) BPMN: Type dimension (50%) BPMN: Off-Page Connector (100%) UML: Activity (Business workflow) diagram: Control Flow (80%)	
PIM	System Model (Designer)	UML: Class diagram: Class (100%) UML: Class diagram: Relations (100%)	UML: State machine:State (60%) UML: State machine:Transitions (80%) UML: Sequence: Messages lost or found (60%) UML: Sequence: Messages self (60%) UML: Sequence: Constraint time (60%) UML: Communication: Object (80%) UML: Communication: Message (60%)	UML: Deployment: Node (100%) UML: Deployment: Instance (100%) UML: Deployment: Artifact (100%) UML: Deployment: Association (100%)	UML: Sequence: Actor (80%) UML: Communication: Actor (80%)	UML: Sequence : Life line (100%) UML: Communication: Life line (100%)	
PSM	Technology Model (Builder)						
Physical	Detailed Representation (Programmer)						
	Functioning (User)						

Fig. 8 Taxonomic analysis of the modelling elements of the UML and BPMN modelling languages

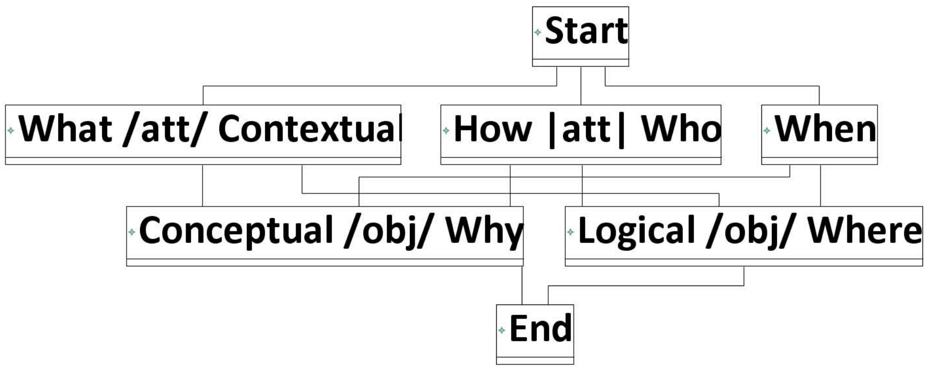


Fig. 9 Lattice generated from the classification shown in Figure 7.

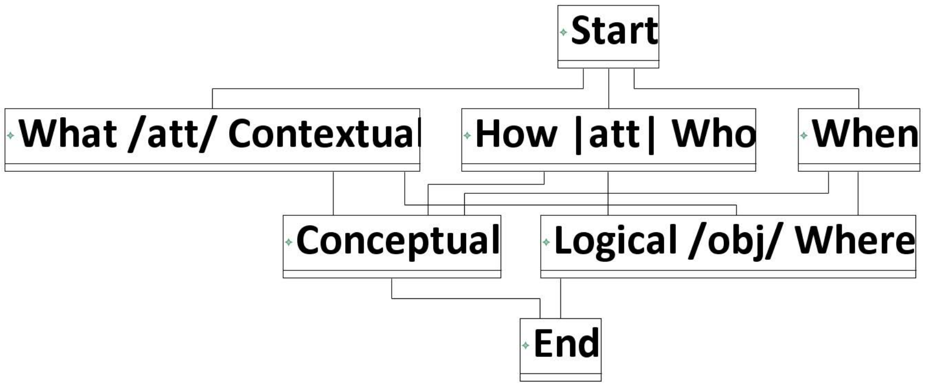


Fig. 10 Lattice generated from the classification shown in Figure 8.

Finally, EMAT generated the lattice shown in Figure 9. Following the interpretation process that is presented in Section 2.5.3, the following inferences are detected from the classification of diagrams:

- When the *What* viewpoint is defined, it also defines the *Contextual* level.
- When the *Contextual* level is defined, it also defines the *Why* viewpoint.
- When the *Logical* level is defined, it implicitly defines the *Where* viewpoint.
- The *How* and the *Who* viewpoints are indistinctly defined with the use of the involved modelling languages and their diagrams.
- The *When* viewpoint has not modelled consistently with respect to the other viewpoints and abstractions under analysis. In other words, the modelling of *When* is very different with respect to the other selected cells in the taxonomy.

EMAT applies a similar procedure for the classification of modelling elements shown in Figure 8. This generates the lattice shown in Figure 10. In this case, the consistency issues are in the *When* and *Conceptual* viewpoints.

From the taxonomic analysis on UML and BPMN, MMQEF infers the following quality issues that must be addressed for integrally modelling the main concerns of an IS project:

- BPMN does not cover the data modelling of business processes.

– The *Where* viewpoint is not integrally covered at the CIM levels for the two modelling languages.
– There is no explicit specification to manage the traceability from the *Conceptual* to the *Physical* abstraction levels. For this reason, decisions about the derivation of code or executable components are related to specific tools. The modelling languages do not explicitly define this generation.
– It is not clear whether or not the models associated to the *Conceptual* level have any relation to the models of the *Logical* level. It can be interpreted as an attempt to directly generate software infrastructure from *Conceptual* models, or a lack of alignment among these levels for modelling complementary concerns (from the perspective of an integral process of code generation based on the preservation of the main modelling constructs).
– Instead of competing initiatives, UML and BPMN could complement each other to model the *Conceptual* abstraction level and its associated viewpoints. BPMN can take advantage of the UML stereotype proposed in the Unified Process for covering complementary concerns at business levels, such as business goals, business rules, business workers, business key concepts, and entities.
– The *Time* viewpoint does not have explicit support. It is a complementary property of the modelling artifacts that were classified.
– The information associated to the *Who* in the taxonomic analysis does not allow the generation of specific user supports at lower levels (e.g., rules for access controls, user management, etc). Similar to the *Time* viewpoint, the information from modelling artifacts that satisfies the *Who* viewpoint relies on specific properties of modelling artifacts.

2.6.2 Quality analysis of the OO-Method and CA integration

In this section, we use the MMQEF method to evaluate the integration of two previous modelling methods that were generated from (and also supported by) the PROS Research Centre: the OO-Method (Pastor and Molina, 2007; Pastor et al., 2013), and the Communicational Analysis (CA) method (España et al., 2009; España Cubillo, 2012).

The OO-Method is a model-driven, object-oriented software development method that generates complete applications from conceptual models. This method is a pioneer in the model-driven field, and one of the first proposals for obtaining software products from source conceptual schemas. The OO-Method uses a methodological process based on the object-orientation paradigm, in which its models (i.e., the object model, the dynamic model, the functional model, and the presentation model) contain the static, dynamic, and presentation properties of a modelled system. The OO-Method underlies the *Olivanova* framework, which defines a compiler of conceptual schemas and a model execution system. Currently, the OO-Method is technologically supported by the *Integranova* platform[6].

CA is a communication-oriented requirements engineering method that is focused on the specification and modelling of the communicative messages and events that are associated to business processes. It is possible to generate OO-Method models from these models. CA has an associated tool, the GREAT modeller (Rueda et al., 2015); this tool is based on Eclipse EMF and models business processes, their

[6] Tool available at http://www.integranova.com/ .

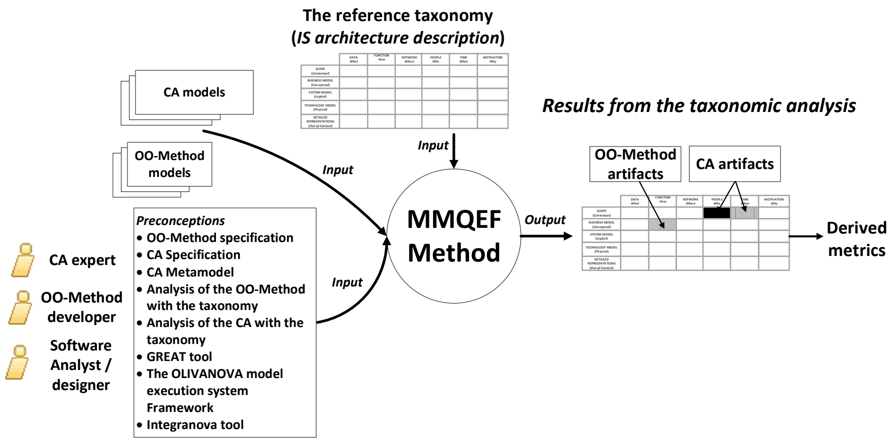

The reference taxonomy
(IS architecture description)
CA models
OO-Method models
Input
Input
Input
MMQEF Method
Output
Results from the taxonomic analysis
OO-Method artifacts
CA artifacts
Derived metrics
CA expert
OO-Method developer
Software Analyst / designer
Preconceptions
• OO-Method specification
• CA Specification
• CA Metamodel
• Analysis of the OO-Method with the taxonomy
• Analysis of the CA with the taxonomy
• GREAT tool
• The OLIVANOVA model execution system Framework
• Integranova tool

Fig. 11 Summary of the application of the MMQEF method to the integration of the OO-Method and the CA methods.

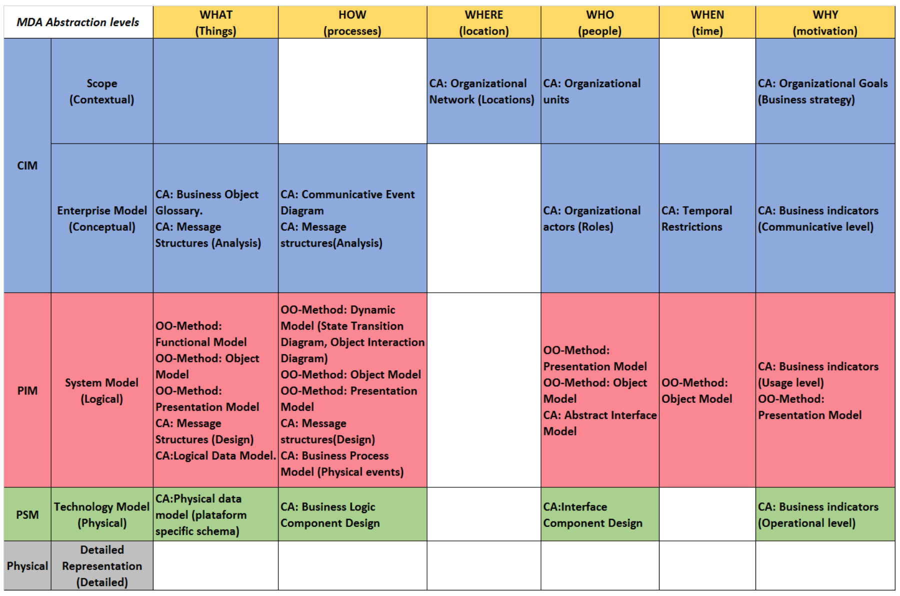
communicative events and messages, and generates OO-models through transformations of these communicative models.

The methodological integration of these two modelling methods was proposed in order to enrich the OO-Method with *requirement modelling* capabilities (González et al., 2011; Pastor and España, 2012). Thus, the OO-Method adds modelling support at the CIM level of the MDA specification (the Contextual and Conceptual levels of the reference taxonomy), complementing its native support for the PIM (Logical) and PSM (Physical) levels. This enrichment allows requirements and their associated transformations to conceptual schemes to be defined in an automated way. The integration of the OO-Method and CA is a clear demonstration of the *Requirements2Code* metaphor (Pastor et al., 2013), which uses a well-defined set of models and model transformations for this purpose.

Another feature that supports our analysis of the OO-Method and CA methodological integration is the evidence of previous reports where both methods were analyzed independently with the reference taxonomy. The resulting analyses were reported in (de la Vara et al., 2007) (for OO-Method) and (España Cubillo, 2012) (for CA). Taking into account these two analyses, we use each independent analysis to overlap them and derive crosses in the cells of the reference taxonomy with the diagrams of both modelling approaches. Therefore, the *contribution values* were assumed in similar percentages regarding the number of elements of both approaches that support each one of the cells involved in the analysis.

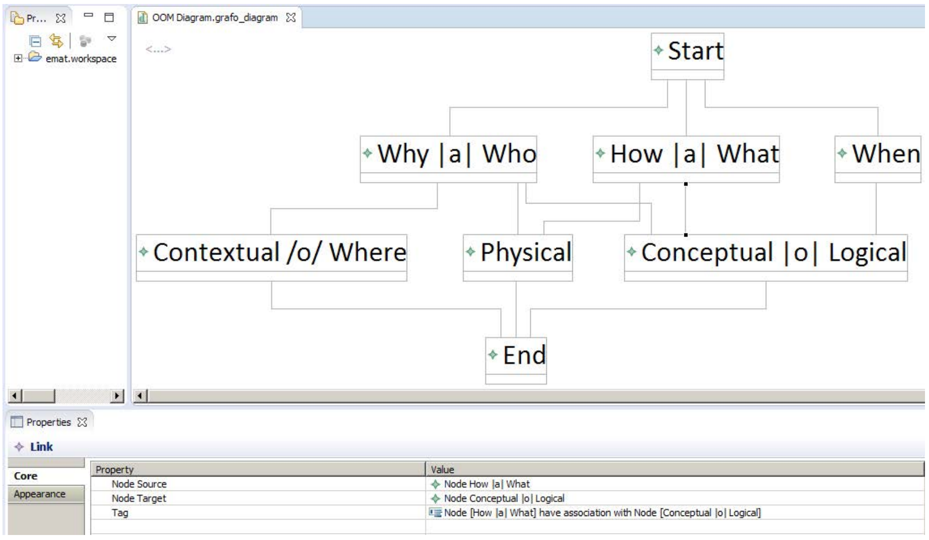
To start the analytic procedure, we classify the modelling artifacts from both methods. These come mainly from their associated diagrams. Figure 11 summarizes the application of the MMQEF method in this analysis. The inputs come from the conceptual specifications of the methods, their related publications, the previous individual classifications, and the information derived from their associated tools (Integranova and GREAT modeller, respectively).

Figure 12 presents the obtained classification. Derived data from representations are placed in the cells in accordance with the information that can be captured by each one. Taking into account that the *Physical* level is covered by



MDA Abstraction levels		WHAT (Things)	HOW (processes)	WHERE (location)	WHO (people)	WHEN (time)	WHY (motivation)
CIM	Scope (Contextual)			CA: Organizational Network (Locations)	CA: Organizational units		CA: Organizational Goals (Business strategy)
	Enterprise Model (Conceptual)	CA: Business Object Glossary. CA: Message Structures (Analysis)	CA: Communicative Event Diagram CA: Message structures(Analysis)		CA: Organizational actors (Roles)	CA: Temporal Restrictions	CA: Business indicators (Communicative level)
PIM	System Model (Logical)	OO-Method: Functional Model OO-Method: Object Model OO-Method: Presentation Model CA: Message Structures (Design) CA:Logical Data Model.	OO-Method: Dynamic Model (State Transition Diagram, Object Interaction Diagram) OO-Method: Object Model OO-Method: Presentation Model CA: Message structures(Design) CA: Business Process Model (Physical events)		OO-Method: Presentation Model OO-Method: Object Model CA: Abstract Interface Model	OO-Method: Object Model	CA: Business indicators (Usage level) OO-Method: Presentation Model
PSM	Technology Model (Physical)	CA:Physical data model (plataform specific schema)	CA: Business Logic Component Design		CA:Interface Component Design		CA: Business indicators (Operational level)
Physical	Detailed Representation (Detailed)						

Fig. 12 Taxonomic analysis of the diagrams of the OO-Method and CA methods.



Property	Value
Node Source	Node How |a| What
Node Target	Node Conceptual |o| Logical
Tag	Node [How |a| What] have association with Node [Conceptual |o| Logical]

Fig. 13 Lattice generated from the classification shown in Figure 12.

the code and infrastructure that is generated by the Integranova platform, the methodological integration has a coverage of 79.167% of the CIM-PIM-PSM levels, highlighting the coverage at CIM cells. The lattice of Figure 13, which was generated automatically by EMAT, shows this coverage and the closeness semantics of the OO-Method and CA integration.

The integration of OO-Method and CA establishes a very complete modelling approach that models business concerns and guarantees their traceability to software platforms through the native mapping that is previously implemented in the OO-Method. This traceability takes attributes and properties belonging to the communicative messages and events and derives conceptual models at the OO-Method level. Works such as (España et al., 2012) report previous applications of this derivation procedure. The traceability relations have the capacity to potentially support the generation of other OO-Method models such as the Presentation model.

However, the integration of the two modelling methods lacks coverage of the *Where* viewpoint, which is delegated to the specific constraints that are imposed by the code generation frameworks that are used by the technical environment of the OO-Method. Nevertheless, taking advantage of the *Organizational network* diagram of the CA method, underlying models can be identified to address specific concerns associated to the *location* question. This would provide an interesting opportunity to model new IS phenomena regarding this viewpoint, such as service platforms, cloud deployment, micro-services, ubiquitous interfaces, etc.

3 The formal support of the taxonomy for the quality evaluation analysis of modelling language artifacts

The main feature of the reference taxonomy that is used by MMQEF is the classification of modelling languages that fit each other in a systemic way (Wegmann et al., 2008). Thus, the sufficiency of the taxonomy must be evaluated to determine the support of this feature and the derived reasoning about quality on modelling languages. To do this, we apply an ontological evaluation procedure proposed in (Siau and Rossi, 1998). In this procedure, the elements of the taxonomy are matched w.r.t. constructs from previous IS methods in order to verify the *essential notions* to be met by any well-constructed taxonomy. These IS methods are proposed in (Guarino and Welty, 2000)(Welty and Guarino, 2001).

The evaluation methods define four meta-properties for the understanding, comparison, and integration of taxonomies: *identity, unity, essence*, and *dependence*. The *identity* meta-property distinguishes a specific instance of a certain class from other instances of that class by means of a characteristic property which is unique for this instance. The *unity* meta-property distinguishes the parts of an instance from the rest of the world by means of a unifying relation that binds the parts, and only the parts, together. The *essential* meta-property discusses which properties of an instance change / do not change over time, and how an instance can be reidentified after some time. Finally, the *dependence* meta-property asks about the several relations of the instances.

The reference taxonomy supports all four meta-properties as follows:

– The scope of each cell (by the classifiers that are extracted from the *abstractions-questions* combination) offers enough information to define an *identity* for a specific IS concern under modelling. This *identity* allows a modelling language artifact to be classified through an ontological reasoning that relates its intention with the identity information of the cell. Simultaneously, the whole/part notion is managed by this taxonomy through the rules for integrity in columns and rows (R2 to R6 of the taxonomy).

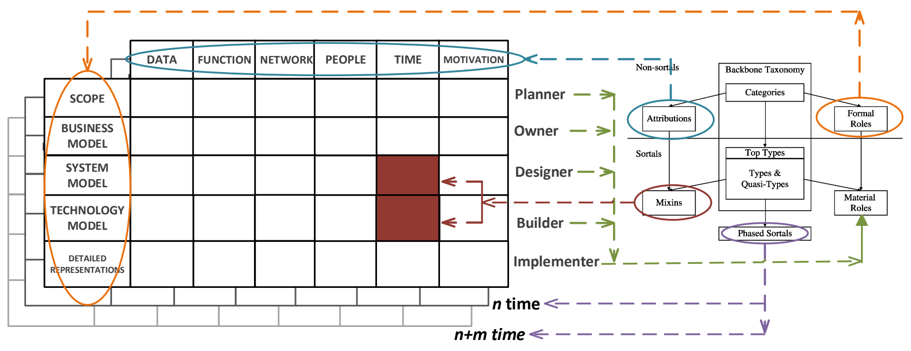


Fig. 14 Zachman taxonomy matching the *Backbone taxonomy*(Guarino and Welty, 2000)(Welty and Guarino, 2001).

– *Unity* of the taxonomy at the column level is achieved by the adherence of the artifacts that are classified in the columns with the predefined generic model of each column (R2). For the rows of the taxonomy, *unity* is achieved by the alignment of each cell with the principles of each abstraction level in order to create a single row model that consists of multiple (and different) models of viewpoints (questions) with a consistent semantic link.
– For the *essential* meta-property, the classifiers of the taxonomy offer the evaluation mechanism required to identify the evolution of modelling language artifacts through the management of the inserted changes during the lifecycle of that artifact. This is the key factor in managing the traceability of models.
– For the *dependency* meta-property, the classifiers and rules of the framework define relations of dependency among the classified artifacts, which systematically harmonize with the independence of each cell.

In addition, the *backbone taxonomy* concept is defined in the IS methods used for the ontological evaluation of taxonomies. This is a set of special properties that are associated to a taxonomy for imparting structure in an ontology, facilitating human understanding, and enabling integration of knowledge. The *backbone taxonomy* is composed of the following concepts:

– *Formal roles:* these are properties that express the part that is played by one entity in an event, often exemplifying a particular relationship between two or more entities.
– *Material roles:* these are elements that inherit identity conditions from some type. They represent roles that are constrained to particular types of entities.
– *Phased sortals:* they correspond to a certain temporal phase of their instances.
– *Attributions:* these represent values of attributes (or qualities) like color, shape, etc.
– *Mixins:* these properties intuitively represent various combinations (disjunctions or conjunctions).

Figure 14 presents the mapping of the reference taxonomy mentioned in our work with the *backbone taxonomy*. For this case, *formal roles* are mapped to the roles that are associated to the abstraction levels (originally defined as *perspectives*). The *material roles* are those that are involved in the modelling act itself (i.e., either defining/using/supporting a modelling language), such as business expert, system analyst, designer, architect, developer, ethnographer, security expert, etc. They are defined as roles that are associated with IS views and inherit from roles of abstraction levels.

The *phased sortals* concept refers to the different times in which analysis can be performed. For this case, we have temporal analysis when the modelling language crosses between the MDA abstraction levels that the reference taxonomy supports. Another *phased analysis* occurs when the taxonomy is applied in posterior stages of the IS modelling process over time, e.g., to check the progress of the IS modelling and development (R7 of the taxonomy). *Attributions* are defined by the essential model of each column and the semantic coherence of each row. Both elements define the attributes of the modelling artifacts under classification. *Mixins* are the combination of properties from abstractions and questions that are expressed as cells.

3.1 The Zachman framework as a taxonomic theory

While the above theoretical evaluations demonstrate the sufficiency of the taxonomy to derive reasoning and understanding, an analysis is required from a more taxonomical perspective to verify if the reference taxonomy that MMQEF uses could be considered as a *taxonomy theory* (i.e., a theory that prescribes how to classify objects of interest, explain similarities and differences among objects, and derive analysis).

To do this, we use a prescriptive framework formulated in (Muntermann et al., 2015) for determining whether or not the Zachman framework qualifies as a taxonomic theory. This is done through a three-condition procedure as follows:

- *Condition 01:* The candidate taxonomy must be formally represented, and it must meet a four-evaluation criteria analysis (*Usefulness, Clarity of Classification, Completeness and Exhaustiveness,* and *Expandability*).
- *Condition 02:* It must include components that describe the theory.
- *Condition 03:* It must provide a foundation to develop other theories.

To meet *Condition 01*, there are previous reports about the formal representation of the taxonomy; one example can be found in (Martin and Robertson, 1999). In addition, the above criteria are met as follows:

- *Usefulness:* the taxonomy serves to identify and classify architectural representations (conceptual models) that relate concepts in an IS to the representations in underlying computational platforms. The taxonomical proposal is an IS architecture framework that takes advantage of architectural representations for understanding an IS.
- *Clarity of Classification:* the taxonomy clearly defines how to classify conceptual models and also defines the characteristics of each category in which models can be placed.
- *Completeness and Exhaustiveness:* the taxonomy defines the main abstraction levels and viewpoints for fully understanding an IS that is developed with conceptual models.

– *Expandability:* the taxonomy establishes a recursive logic that supports the classification for complex concepts.

To meet *Condition 02*, the main components of the theory (i.e., the bi-dimensional structure, and the rules and principles for classification) are described in (Zachman, 1987; Sowa and Zachman, 1992).

Condition 03 is met through the functions and analytic procedures that must be done to place modelling languages in the taxonomy, and, therefore, to derive quality inferences. The taxonomy provides the conceptual foundation to support the methodological and technological framework of the MMQEF method.

Because the Zachman-based taxonomy that is used in the MMQEF method satisfies the three conditions of (Muntermann et al., 2015), it can be considered to be a taxonomic theory with a conceptual foundation that is sufficient to support quality evaluation at model-driven levels.

4 Discussion and trade-off of the method

Most of the current MDE challenges require methodological tools that are aligned with model-driven principles. Tools without conceptual support affect the adoption of MDE (which have been extensively reported (Giraldo et al., 2016a)). In addition to technical support, methodological guidance is required in order to perform analytic reasoning on models.

One of the most relevant features of this guidance is the possibility to reason about *when* a given artifact is *model-driven compliant* (i.e., the assumption of whether or not a modelling artifact has a clear purpose for an IS concern, independently of notational justifications (e.g., a language profiling, any language with new graphical elements, and similar)).

The reference taxonomy provides a foundational architectural description of IS from an organizational perspective. It natively uses models to support all the issues (cells) that are required to fully understand an IS (from organizational levels to technical implementation levels). The reference taxonomy meets the conceptual model defined in the ISO 42010 standard (ISO/IEC/IEEE, 2011) for architecture frameworks, architecture descriptions, architecture description languages, elements, correspondences, decisions, and rationale.

Following the ISO 42010 standard, the taxonomy provides an architecture description for IS and the associated architectonical practices. We believe that there is a strong correlation between the model-driven paradigm and the IS principles that the taxonomy proposes. Therefore, the reference taxonomy allows the assessment of the current MDE quality challenges.

MMQEF provides the required support for the quality evaluation of modelling languages. This includes metrics for models and modelling languages, orientation derived from the analysis of the taxonomy, and information for making decisions for specific situations (e.g., a decision about the convenience or suitability of a specific modelling language). Some advantages of MMQEF are:

– The method is in compliance with recognized ontological frameworks and standards for IS due to the use of the reference taxonomy.
– An explicit and standardized *visual support* is provided to analyze modelling initiatives. This is similar to an ontological evaluation; however, instead of

using a subjective conception of a domain, the method uses a universal IS architecture that has been accepted by the most recognized IS standards.
- It facilitates the formulation of metrics for the *coverage* of a model artifact or a modelling language in accordance with the cells involved in the different abstraction levels and modelling dimensions of the taxonomy. In addition, orientations and guidelines can be obtained based on the associated reasoning and the intentions of specific modelling efforts or communities.
- This method gives conceptual and methodological support for evaluating the suitability of a modelling language in accordance with its purpose, the identified coverage, and the abstraction levels and modelling dimensions that are involved.
- The method also considers the information that can be obtained from the diagrams (e.g., the resulting artifact from the notation and the semantics of a modelling language) with respect to the intentions of use of the modelling artifact and the intentions of the modelling effort.
- The integration capacities that are offered by modelling languages can be explicitly identified.
- Support is obtained for the management of mappings, transformations, and the traceability relations between modelling artifacts. The taxonomy makes the type of the analyzed relation and the required specifications to support processes of models transformations explicit.
- The taxonomy defines the most granular elements of modelling (i.e., the minimal information that must be considered in a modelling effort). These granularity levels are related to the viewpoints (Henderson-Sellers and Gonzalez-Perez, 2010) (in this case, the philosophical questions).
- As a conceptual and methodological tool, MMQEF (through EMAT) provides a shared knowledge repository for reasoning, analyzing, and communicating quality issues on modeling artifacts and modelling languages, with their implications in a real IS project.

4.1 Why another quality framework for MDE?

Because MMQEF solves some open challenges regarding modelling language quality evaluation. In (Giraldo et al., 2016b) we listed some open pending challenges in the quality evaluation of modelling languages. MMQEF address each one of these challenges as follows:

- *Language/model according to MDE (MDE compliant)*: MMQEF allows the purpose of the modelling languages under analysis to be verified by locating them into the specific cells of the reference taxonomy, and thus, identifying the association of the languages with the abstraction levels, the capacities of the languages to integrate with others, the support for model transformations, and the generation of concrete functional platforms.
- *Multiple modelling languages*: MMQEF evaluates the suitability of a set of modelling language to support specific IS phenomena. For each language under analysis, MMQEF determines the completeness, coverage, and integration capacities provided by the languages.
- *Explicit management of abstraction levels*: the reference taxonomy that is used in the MMQEF method allows the abstraction levels involved in an IS project

to be explicitly considered, from the business to the functional implementation level.

- *Metrics over models*: MMQEF defines five metrics with their associated decision criteria to support reasoning about modelling languages previously classified in the reference taxonomy.
- *Models Transformations as a managed process*: in a models transformation process, the method allows evaluating whether or not the chosen source-target modelling languages are appropriate.
- *Semantic in the diagram*: MMQEF allows managing the artifacts that are associated to modelling languages, which result from the interaction of the users with the languages (i.e., the interaction with the models, the navigation through structures of related models, the simulation of expected behaviors, and queries over models).
- *Agile ontological analysis*: the method provides a precise and prescriptive set of task and activities to evaluate the quality of modelling languages.
- *The management of quality issues in modelling languages as technical debt evidence*: we project to use the semantic models that are generated by the EMAT tool of the MMQEF method as the input to calculate technical debt on modelling languages using previous services for the calculus and management of technical debt in software development environments.

Because it is not just another framework, since it can be used in combination with previous approaches. MMQEF does not attempt to be another isolated framework; it can be used in combination with other frameworks. The analytic procedure (supported in the classification of modelling artifacts) allows quality dimensions and properties that are formulated in previous frameworks to be addressed. Most of these dimensions and properties are empirically evaluated with respect to the expectations and intentions of the authors of the frameworks or the analysts.

For example, Table 1 (which is extracted from the table originally formulated in (Krogstie, 2012)) presents the support of MMQEF for the SEQUAL framework, which is one of the most important quality frameworks for the model-driven and model-based fields. This table presents a summary of the SEQUAL properties for modelling languages. This table also contains the support of MMQEF to address the items of SEQUAL from our perspective and the descriptive precepts of SEQUAL. The last column of Table 1 (MMQEF support) describes the activities of MMQEF that allow SEQUAL properties to be addressed.

Specific properties of SEQUAL for modelling languages are supported by the classification and evaluation activities defined in MMQEF. The quality evaluation of MMQEF is framed within organizational, system, and technological perspectives.

The visual language provided by the taxonomy makes the relationship of the IS domain with the modelling language(s) under analysis explicit, including the support of the language for modelling organizational issues of an IS project (the higher levels of the taxonomy). This is the *domain* and *organizational appropriateness* of SEQUAL.

Classification of information from modelling languages and the posterior evaluation activities support properties that are related to the interpretation of actors who are involved in the modelling act (the *participant* and *modeler* appropriate-

Quality type and characteristics	*Means*			*MMQEF support*
	Beneficial existing quality	Model and language properties	Modelling techniques and tool support	
Semantic validity	Physical	Domain appropriateness	Statement insertion	*Activity 1*
Completeness	Syntactic	Participant appropriateness	Statement deletion	*Activities 1 and 2*
		Modeller appropriateness	Behavioural meta-modelling	*Activities 1 to 5*
		Language extension mechanisms	Meta-model adaptation	*Activity 3*
		Formal semantics		*Activity 1*
		Modifiability	Driving questions	*Activities 1 and 5*
		Analyzability	Model reuse	*Activities 1, 4, and 5*

Table 1 Support of the MMQEF method for the quality levels, goals, and means of the SEQUAL framework.

ness). Lower levels of the taxonomy address issues about the technical implementations through which the modelling effort (made with modelling languages) is computationally supported (the *tool appropriateness*).

The reference taxonomy defines a conceptual artifact for reasoning about the application of modelling approaches to model IS concerns in an organizational context. MMQEF takes advantage of this reasoning to perform quality evaluation at higher levels such as semantics, deontic, social, and pragmatics (including the understanding of both human and tool). Therefore, the organizational adoption of modelling initiatives is also considered to be part of the quality evaluation assessment. Quality dimensions such as the goals of modelling, the explicit knowledge of the audience, and interpretations are also easily addressed by MMQEF and reported in EMAT.

Other quality evaluation methods can also complement their analyses with the combined use of MMQEF and EMAT. For example, the quality goals proposed in the 6C approach (Mohagheghi et al., 2009) (*Correctness, Completeness, Consistency, Comprehensibility, Confinement*, and *Changeability*) can be accurately expressed through their association with the taxonomic analysis and the metrics defined in MMQEF. Another example is the *Physics of Notations* (PoN) work (Moody, 2009), which can complement its notational scope with the semantic analysis of MMQEF. Thus, PoN cognitive principles such as the *Semantic transparency*, *Semiotic clarity*, *Cognitive integration*, *Cognitive fit*, and *Complexity management* can be easily analyzed with the support provided by MMQEF.

In addition, the MMQEF method proposes the EMAT tool to manage the data of the semantics that is derived from the taxonomic analysis. This is a clear advantage of MMQEF over other quality frameworks because they do not provide any concrete (native) tool to support their quality assessment.

4.2 Related works

Some works previously used the taxonomic framework to reason about modelling languages. (Molina et al., 2014) presents a scenario in which the framework was used as a conceptual tool to systematically integrate a set of modelling languages that are used in a development process of a groupware software. In this way, the authors achieve a harmonization of modelling languages without detriment to the expectation of the participant roles. The main benefit of this harmonization for an IS project is the generation of computational platforms that cover the multiple expectations of the IS, each of which is modelled with its own resources of its associated viewpoints.

In (Frankel et al., 2003), a systemic combination between MDA and the reference taxonomy was proposed. MDA explicitly supports the taxonomy through the definition of the abstraction levels and the foundations of mappings in the top-down relations among these levels. The taxonomy complements MDA with the definition of the *modelling dimensions* (from the philosophical questions) that supports the mapping and transformations of models at conceptual levels, which are independent of their implementation on a specific model transformation language.

However, currently, there are no reports about the applicability of the reference taxonomy as a model evaluation tool that supports inferences and analytics on models. Some works that propose the use of the taxonomic structure for modelling-related tasks are evident. For example, the authors in (Kingston and Macintosh, 2000) make suggestions about modelling approaches for a medical domain; these suggestions are made by performing a classification of modelling alternatives in the taxonomy. The authors also propose the use of individual perspectives of the taxonomy as *user interfaces* for a knowledge distribution system. In the analysis with the taxonomy reported in (Noran, 2003), a set of modelling languages is suggested to populate each cell in accordance with the purpose of the languages and the specific tasks that are associated to each cell.

The support of the taxonomy for inferences at ontological levels has also been reported. (Kingston, 2008) describes how the taxonomy was used for managing multi-perspective modelling in an ontology development process. The resulting reasoning comes from the classification activity on the type of knowledge that is addressed. The R7 rule of the taxonomy (recursivity) was used in (Garner and Raban, 1999) to propose an IS context management approach that provides a dynamic validation of user requirements. The classifiers of the taxonomy were used by (de Graaf et al., 2014) to address an ontological approach in specific architecture scenarios.

A similar work about relations among viewpoints is reported in (Romero et al., 2009), where the authors take advantage of the RM-ODP viewpoints to propose a generic model-driven approach for the specification and realization of correspondences (relationships) among these viewpoints. However, unlike the reference taxonomy, RM-ODP focuses on the development of the architecture so that the rationale and tradeoffs of the architecture do not belong to the RM-ODP model (Tang et al., 2004).

With regard to the operationalization of analytic procedures for evaluating modelling languages, a work is reported in (Shuman, 2010) which proposed a checklist for reviewing issues of operational executable architectures. Thus, the features of modelling languages (diagrams or specific graphical constructs) were classified and measured according to the items that were extracted from the DoDAF framework. Unlike our work, this operationalization was proposed as a spreadsheet file and was not included in a MDE technical environment (Section 2.5.1).

5 Conclusions and further work

In this work, we have presented an analytic procedure proposal for evaluating the quality of modelling languages through the use of taxonomic analysis. The classification activity is used to answer when any modelling initiative is model-driven compliant (i.e., *when* it is in MDE) through its alignment with IS concerns defined in a reference taxonomy that is extracted from a recognized IS architecture description. With the FCA mathematical method, we demonstrated the formality of the reference taxonomy for supporting quality evaluation procedures on modelling artifacts and for managing the semantic data that is generated in these procedures. The EMAT tool for operationalizing these quality procedures was also reported.

As further work, we are improving the tool by adding complementary visualization options to interpret the conceptual lattices obtained (e.g., radial graphs). In addition, we will populate the tool with more examples of taxonomic analysis in order to provide more precise guidance for potential users of the tool. The quality evaluation method by taxonomic analytics will be specified in more detail to demonstrate its advantages and its potential for the MDE and IS communities and practitioners.

Acknowledgements

F.G. would like to thank COLCIENCIAS (Colombia) for funding this work through the COLCIENCIAS Grant call 512-2010. This work has been supported by the Generalitat Valenciana Project IDEO (PROMETEOII/2014/039), the European Commission FP7 Project CaaS (611351), and ERDF structural funds. F.G. would to thank César A. Cataño and Juan D. Fernández for their support in the implementation of EMAT tool.

References

Aagesen, G. and Krogstie, J. (2015). *BPMN 2.0 for Modeling Business Processes*, pages 219–250. Springer Berlin Heidelberg, Berlin, Heidelberg.

da Silva, A. R. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139 – 155.

de Graaf, K., Liang, P., Tang, A., van Hage, W., and van Vliet, H. (2014). An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, 65(7):1053 – 1064.

de la Vara, J. L., Díaz, J. S., and Pastor, O. (2007). Integración de un entorno de producción automática de software en un marco de alineamiento estratégico (in spanish). In *Anais do WER07 - Workshop em Engenharia de Requisitos, Toronto, Canada, May 17-18, 2007*, pages 68–79.

España, S., González, A., and Pastor, Ó. (2009). *Communication Analysis: A Requirements Engineering Method for Information Systems*, pages 530–545. Springer Berlin Heidelberg, Berlin, Heidelberg.

España, S., Ruiz, M., and González, A. (2012). Systematic derivation of conceptual models from requirements models: A controlled experiment. In *2012 Sixth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12.

España Cubillo, S. (2012). *Methodological integration of Communication Analysis into a model-driven software development framework.* PhD thesis.

Frankel, D. S., Harmon, P., Mukerji, J., Odell, J., Owen, M., Rivitt, P., Rosen, M., and Soley, R. M. (2003). The Zachman Framework and the OMG's Model Driven Architecture.

Ganter, B. and Wille, R. (1999). *Concept Lattices of Contexts*, pages 17–61. Springer Berlin Heidelberg, Berlin, Heidelberg.

Garner, B. and Raban, R. (1999). Context management in modeling information systems (is). *Information and Software Technology*, 41(14):957 – 961.

Giraldo, F. D., España, S., and Pastor, O. (2016a). Evidences of the mismatch between industry and academy on modelling language quality evaluation. *CoRR*, abs/1606.02025.

Giraldo, F. D., España, S., Pastor, Ó., and Giraldo, W. J. (2016b). Considerations about quality in model-driven engineering. *Software Quality Journal*, pages 1–66.

González, A., España, S., Ruiz, M., and Pastor, Ó. (2011). *Systematic Derivation of Class Diagrams from Communication-Oriented Business Process Models*, pages 246–260. Springer Berlin Heidelberg, Berlin, Heidelberg.

Goulão, M., Amaral, V., and Mernik, M. (2016). Quality in model-driven engineering: a tertiary study. *Software Quality Journal*, pages 1–33.

Guarino, N. and Welty, C. A. (2000). A formal ontology of properties. In *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, EKAW '00, pages 97–112, London, UK, UK. Springer-Verlag.

Henderson-Sellers, B. and Gonzalez-Perez, C. (2010). Granularity in conceptual modelling: Application to metamodels. In Parsons, J., Saeki, M., Shoval, P., Woo, C., and Wand, Y., editors, *Conceptual Modeling ER 2010*, volume 6412 of *Lecture Notes in Computer Science*, pages 219–232. Springer Berlin Heidelberg.

ISO/IEC/IEEE (2011). Iso/iec/ieee systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1–46.

Kingston, J. (2008). Multi-perspective ontologies: Resolving common ontology development problems. *Expert Systems with Applications*, 34(1):541 – 550.

Kingston, J. and Macintosh, A. (2000). Knowledge management through multi-perspective modelling: representing and distributing organizational memory. *Knowledge-Based Systems*, 13(23):121 – 131.

Krogstie, J. (2012). *Quality of Models*, pages 205–247. Springer London, London.

Kruchten, P. (2000). *The Rational Unified Process: An Introduction, Second Edition.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition.

Laware, G. and Kowalkowski, F. (2005). The business value of taxonomies and ontologies for web & knowledge management practices. pages 41–47. cited By 0.

Martin, R. and Robertson, E. L. (1999). Formalization of multi-level zachman frameworks (technical report no. 522). Technical report, Computer Science Department, Indiana University.

Mohagheghi, P., Dehlen, V., and Neple, T. (2009). Definitions and approaches to model quality in model-based software development a review of literature. *Information and Software Technology*, 51(12):1646 – 1669. Quality of {UML} Models.

Molina, A. I., Giraldo, W. J., Ortega, M., Redondo, M. A., and Collazos, C. A. (2014). Model-driven development of interactive groupware systems: Integration into the software development process. *Science of Computer Programming*, 89, Part C(0):320 – 349.

Moody, D. (2009). The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779.

Muntermann, J., Nickerson, R., and Varshney, U. (2015). Towards the development of a taxonomic theory. In *Proceedings of the Twenty-first Americas Conference on Information Systems (AMCIS), Puerto Rico, 2015*, AMCIS'15. AIS Electtronic Library (AISeL).

Noran, O. (2003). An analysis of the zachman framework for enterprise architecture from the {GERAM} perspective. *Annual Reviews in Control*, 27(2):163 – 183.

Object and Reference Model Subcommittee of the Architecture Board (2005). A Proposal for an MDA Foundation Model, ormsc/05-04-01. Technical report, Object Management Group.

Olivé, A. (2001). Taxonomies and derivation rules in conceptual modeling. In *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, CAiSE '01, pages 417–432, London, UK, UK. Springer-Verlag.

OMG (2003). Mda guide version 1.0.1.

OMG (2014). MDA Guide revision 2.0.

Pastor, Ó. and España, S. (2012). *Full Model-Driven Practice: From Requirements to Code Generation*, pages 701–702. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pastor, O., Insfrán, E., Pelechano, V., Romero, J., and Merseguer, J. (2013). *OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods*, pages 139–152. Springer Berlin Heidelberg, Berlin, Heidelberg.

Pastor, O. and Molina, J. C. (2007). *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Priss, U. (2006). Formal concept analysis in information science. *Annual Rev. Info. Sci & Technol.*, 40(1):521–543.

Romero, J., Jaen, J., and Vallecillo, A. (2009). Realizing correspondences in multi-viewpoint specifications. In *Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE International*, pages 163–172.

Rueda, U., España, S., and Ruiz, M. (2015). GREAT process modeller user manual. *CoRR*, abs/1502.07693.

Shuman, E. A. (2010). Understanding executable architectures through an examination of language model elements. In *Proceedings of the 2010 Summer Computer Simulation Conference*, SCSC '10, pages 483–497, San Diego, CA, USA. Society for Computer Simulation International.

Siau, K. and Rossi, M. (1998). Evaluation of information modeling methods - a review. In *HICSS (5)*, pages 314–322.

Smith, R. (2013). On the value of a taxonomy in modeling. In Tolk, A., editor, *Ontology, Epistemology, and Teleology for Modeling and Simulation*, volume 44 of *Intelligent Systems Reference Library*, pages 241–254. Springer Berlin Heidelberg.

Sowa, J. F. and Zachman, J. A. (1992). Extending and formalizing the framework for information systems architecture. *IBM Syst. J.*, 31(3):590–616.

Tang, A., Han, J., and Chen, P. (2004). A comparative analysis of architecture frameworks. In *Software Engineering Conference, 2004. 11th Asia-Pacific*, pages 640–647.

Wegmann, A., Kotsalainen, A., Matthey, L., Regev, G., and Giannattasio, A. (2008). Augmenting the zachman enterprise architecture framework with a systemic conceptualization. In *Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, EDOC '08, pages 3–13, Washington, DC, USA. IEEE Computer Society.

Welty, C. and Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. *Data Knowl. Eng.*, 39(1):51–74.

Wolff, K. E. (1993). A first course in formal concept analysis. *StatSoft*, 93:429–438.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Syst. J.*, 26(3):276–292.

Zhao, L., Letsholo, K., Chioasca, E.-V., Sampaio, S., and Sampaio, P. (2012). Can business process modeling bridge the gap between business and information systems? In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1723–1724, New York, NY, USA. ACM.