Journal of Universal Computer Science, vol. 26, no. 9 (2020), 1230-1264
submitted: 8/7/19, accepted: 11/9/20, appeared: 28/9/20

Using the Scientific Method as a Metaphor to Understand Modeling

Emilio Rodríguez-Priego, Francisco J. García-Izquierdo
Ángel Luis Rubio
(Departamento de Matemáticas y Computación
Universidad de La Rioja, La Rioja, Spain
emilio.rodriguez@unirioja.es, francisco.garcia@unirioja.es, arubio@unirioja.es)

Abstract: Although modeling is used to address complex problems, it is difficult to study modeling itself with an easy to understand model. Many authors have proposed such a model of modeling, but a consensus on the meaning of the basic modeling concepts has yet to materialize. We claim that any proposal regarding the fundamentals of modeling should address several objectives, such as to focus on the concept of model and define what it is, how a model is created and how it relates to the entities it models or to explain the relationship between model and other basic concepts such as metamodel or (modeling-)language. In this paper, we present some of the most important elements of our proposal, named Scientific Method approach to Modeling (SMM). Our proposal uses the Scientific Method as a metaphor to explain the mechanisms of modeling, since it provides well-known mechanisms constantly utilized when developing or understanding models: validation, analysis, synthesis and analogy. Inspired by these mechanisms, our proposal addresses the notion of model by including several constructors that allow us to explain better several complex modeling mechanisms extensively discussed in the literature, such as the metamodel notion.
Key Words: theory of modeling, scientific method, process of modeling, quasi-model, model, metamodel, modeling language
Category: D2.2, H.1, M.4

1 Introduction

The term model is certainly one of the most frequently used in any humanistic or scientific discipline [Magnani et al., 2017, Frigg and Hartmann, 2020]. One of the youngest scientific disciplines, computer science, has utilized different types of models almost since its inception: computational models, data models, software models, conceptual models, domain models, formal models, logic models, information models, mathematical models, etc.

Some of these models are scientific models, i.e. “models that can be found in a wide range of scientific contexts and disciplines” [Magnani et al., 2017]. The nature of this kind of models is the heart of the Philosophy of Science [Frigg and Hartmann, 2020]. The model-based science deals with the use of models for different purposes such as, for example, the ones collected in [Magnani et al., 2017]: simulation, representation, explanation of scientific theories, etc. In contrast, in this paper we present a proposal that takes inspiration from the method used by scientists, scientific method [Andersen and Hepburn, 2016], to explain what a model is, regardless it is used in a scientific context or not.

Formal approaches do also exist, e.g. [Williams, 2013, Henderson-Sellers, 2012, Hodges et al., 1997], wherein mathematics aspire to provide a sufficient conceptual and procedural grounding to explain the essence of modeling. However, these approaches to modeling have not been universally accepted, as is also the case in less formal arenas where the debate over what exactly a model is continues. In fact, before we continue, let us underscore that our intention in this study is to address those non-formal approaches [Muller et al., 2012].

Therefore, it is clear that there are several contexts for the application and study of modeling. Herein, we will mainly focus on that of computer science, although, thanks to the principles established in Section 2, we will not limit the applicability, in whole or in part, of our approach to other contexts.

Despite the mentioned omnipresence of models, in general, modeling is still not a well-understood discipline [Frigg and Hartmann, 2020, Rodriguez-Priego et al., 2010]. It is paradoxical that, although modeling helps us to understand complex problems, taking on the study of modeling itself through an easy to use and understand model is exceedingly difficult. Many authors have proposed such a model of modeling [Magnani et al., 2017, Williams, 2013, Henderson-Sellers, 2012, Hodges et al., 1997, Kühne, 2006, Group, 2014, Gonzalez-Perez and Henderson-Sellers, 2007, Seidewitz, 2003, Stachowiak, 1973, Favre, 2004, Rensink, 2005, Gonzalez-Perez, 2018], so much that the literature includes many articles that define the basic terms of modeling. However, a comparative analysis of these studies reveals considerable inconsistencies, ambiguities and unsolved problems representing red flags for those who are working with modeling for the first time [Rodriguez-Priego et al., 2010].

Two circumstances contribute to the situation presented herein. Firstly, no theory has been universally accepted as the basis of modeling [Magnani et al., 2017, Frigg and Hartmann, 2020]. If such a theory did exist, the myriad of debates held by renowned researchers regarding the foundations of modeling would not continue to occur (some of them have already lasted for over ten years [Magnani et al., 2017, Frigg and Hartmann, 2020, Kühne, 2006, Atkinson and Kühne, 2001, Atkinson and Kuhne, 2003, Atkinson and Kühne, 2002, Kühne, 2009, Atkinson et al., 2000, Kühne, 2010]). Second, there is a lot of modeling approaches to software engineering that are primarily based on object orientation modeling [Kühne, 2006, Group, 2014, Atkinson et al., 2000, Group, 2016, Overbeek, 2006, Group, 2017, Aßmann et al., 2006, Bézivin, 2005]. This has resulted in a somewhat constraining conceptual discourse, as it might seem that any modeling theory in the field of software engineering must necessarily be based on object orientation. Based on our deep study in [Rodriguez-Priego et al., 2010], we have come to the conclusion that, while recognizing the importance of object orientation as a valuable software engineering paradigm, a general modeling theory should enrich the discussion, and allow the inclusion of more types of models and not only those based on the object oriented principles.

Against this background, our proposal to address modeling is structured around several research questions: what is a model?; how is a model created?; how does a model relate to the entities it models?; what are the relationships between the concept of model and other basic concepts such as metamodel or (modeling-)language?; what are the differences and similarities of our approach with the object orientation modeling paradigm?; and finally, how does our approach contributes to clarifying the problems and confusions discussed in the related literature over the course of years? Finding answers to these questions is the goal of our study, so that these answers give rise to a new and consistent approach to modeling that could be used for a better comprehension, description and development of software engineering models, among others. These answers are addressed throughout the forthcoming sections. Thus, Sections 4 and 5 cover the concept of model: its definition, creation mechanisms, types and nature. In Section 7 we elaborate on how our approach addresses certain concepts, such as submodeling and metamodeling. The relationship between model, language and metalanguage should be studied thereafter, but due to space limitations, we can only provide an outline in this regard (see Section 7.4). The reasons why our proposal represents a contribution to the debates in the literature (in particular by comparing it to the object-oriented approach) are discussed in Sections 6 and 7.3.

2 The Principles behind our approach

Our proposal, which we have named Scientific Method approach to Modeling (hereinafter SMM), is governed by three principles: the *Generality principle*, the *Independence principle*, and the *principle of Scientific Method inspiration*.

The *Generality* principle demands an approach that is so basic that it can be adopted, a priori, by any discipline. Therefore, our intention is that SMM can be used to discuss any type of model (software, conceptual, scientific, etc.) that we have just presented in the Introduction. In other words, SMM could potentially be applicable to both system models and domain models, as they are commonly understood. However, it should be noted that our specific objective is to model software systems. Given that it is a general approach, it must clarify and interpret the meaning of other modeling theories, as well as concepts such as class, type, kind, set, universal, law, etc. that traditionally appear in modeling theories. However, a complete interpretation of these concepts in light of our new proposal is beyond the scope of this article for reasons of space. Herein, we focus on comparing the concepts to SMM to the concepts of the object oriented modeling theory as it is the most widely utilized.

Model is not the only term that appears when modeling is studied. From different perspectives and with different terminology, there are three interrelated concepts that are key to approaching any modeling process: model, system and language. Clearly, when it comes to working with a model, you need to understand the meaning of the term model (this being basically the leitmotiv of this article). However, it is even more evident that a model is modeling something and that that "something" is frequently called system. Finally, and sometimes even unconsciously, when we work with a model, we are doing so by knowing, manipulating, speaking or reading expressions of a language. The proposal herein also assumes that the interrelationships among these three concepts represent the most elemental core of the modeling process. However, understanding these interrelationships is challenging because they delve deeper into the realms of linguistics, semiotics, mathematics, philosophy, or epistemology. In addition, these three terms are often used together when trying to define model, making it difficult to understand that concept. The second of our theory's principles, *Independence*, pursues to articulate a definition of model completely independent from language and system terms. Thus we can respond affirmatively to open questions such as whether a model can exist without being expressed in a language, or if it is possible to model entities that are not a system. Despite this achieved independence, which represents one of the contributions of this study, we recognize that the three concepts are indeed related, forming a kind of triangle similar to what is known as semiotic triangle, or Ogden-Richards' triangle [Ogden and Richards, 1923], which other authors have reformulated as modeling triangle [Sykes, 2003, Gupta and Sykes, 2001]. In the vertices of the different versions of this triangle appear three concepts whose exact denomination varies from one version to another: real object, concept and symbol. Figure 1 displays our proposal for the modeling triangle, with the name that we assign to the three mentioned concepts (respectively, original, model and expression) and the relationships that our theory establishes between them. Throughout the article, we will develop the exact meaning of the terms and the scope of their relationships. For now, suffice it to say that, in our proposal, this modeling triangle is simpler than that previously mentioned: it places the concept of model in the middle, and applies the mechanisms of scientific method to relate model with the other vertices.

It is interesting to note that the problem of independence of language arose very early, in the process of selecting a language style for our proposal. We have opted for less formal language, though more formal expressions are included in the definitions, so as to facilitate the comprehension on the issues addressed. We understand, as does [Harel and Rumpe, 2000], that the use of exclusively mathematical language is not synonymous with formal language; nor does the use of natural language necessarily entail a lack of precision.

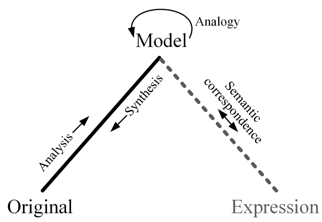

Figure 1: Our proposal for the modeling triangle

Although there is no consensus among scientists on its definition [Andersen and Hepburn, 2016], *scientific method* [Gower, 2012] provides certain well-known mechanisms including, among others, induction/prediction, analysis/synthesis or analogy, whose role in scientific development has been debated for centuries. These ideas are essential to understanding our proposal, not because we have employed it to explain our ideas herein, but rather because it can be used as a metaphor to explain the mechanisms of modeling. In the context of this paper, we will refer especially to the mechanisms of *analysis*, *synthesis* and *analogy.* As indicated by [Andersen and Hepburn, 2016], we consider that, in analysis, a phenomena was examined to discover its basic explanatory principles (inductive method); in synthesis, explanations of a phenomena were constructed from first principles (hypothetical-deductive method). On the other hand, for us, analogy is "to refer to some relation of similarity and/or difference between a model and the world, or between one model and another" [Hesse, 2000]. Despite the fact that, as [Andersen and Hepburn, 2016] points out, "a frequently seen argument is that research based on the H-D method is superior to research based on induction", we state as a working hypothesis that both processes are equally important, as well as that of analogy. As we will show later in this paper, when we present our full models' notion, both processes are compatible, and they can refer to the same model.

In many cases modeling tasks are blurred by the specific terminology of the scientific domain to which modeling is applied, making them more difficult to identify and connect with more basic and general ideas. We uphold that positioning scientific method's basic terms in the very foreground of modeling terminology allows an easily recognizable common vocabulary to be shared, thus facilitating the comprehension of modeling concepts. We claim that scientific method may be used as a metaphor to explain modeling for three reasons: first, we have borrowed those scientific method terms (analysis, synthesis, analogy...) [Andersen and Hepburn, 2016], which will be consistently applied throughout the article to addresses the notions of model, metamodel, etc. Second, these concepts will allow us to make a classification of models based on what we will call *constructors*, which are based on scientific method activities. These constructors create models (through analysis or analogy) or new *originals* corresponding to the models (through synthesis). Finally yet importantly, the process of building the models that will be described in the constructors is, as in scientific method, an iterative process. This underscores the fact, not generally considered in the literature on modelling, that obtaining a model or an original is not generally achieved at the first attempt, but rather after various iterations. This iterative feature of modelling allows the validation of the correspondence between models and their originals, or between models (in the case of analogy).

3 Related Work

The literature on modeling is very extensive and has evolved over time. Perspectives vary greatly and thus far researchers have not reached a consensus regarding the terminology [Muller et al., 2012, Börstler et al., 2012]. Evidence of this can be found in our previous article [Rodriguez-Priego et al., 2010], wherein an in-depth review of 200 references about modeling was conducted. That is the main reason why compiling a related work section about modeling is a challenging task. Even if we used tools like References-enriched Concept Maps [Rodriguez-Priego et al., 2013], the size of the paper would increase greatly. Consequently, we are now focusing on a brief overview of the bibliography of the principal contributions to the theory of modeling in general, while we refer the reader to a more extensive analysis of the related work that can be consulted in the supplementary material [Rodriguez-Priego et al., 2019].

Kühne [Kühne, 2006], in collaboration with other authors such as Atkinson [Atkinson and Kuhne, 2003] has been one of the most prolific authors in this field. His principal contributions have been the distinction between type and token models, linguistic and ontological instantiation, and between classification and inheritance. Gonzalez-Perez and Henderson-Sellers [Gonzalez-Perez and Henderson-Sellers, 2007, Eriksson et al., 2013] have also made various contributions regarding the classification of models according to mappings (isotypical, prototypical, metatypical) powertypes and clabjects. Bézivin has also published many articles (e.g., [Bézivin, 2005, Bézivin and Gerbé, 2001]) pertinent to modeling theory focused on OMG. He distinguishes between the relationships instance-of and conforms-to. And, in his classic article [Bézivin, 2005] he compares OO with modeling to affirm that they are both a paradigm (in the field of software engineering). In a similar vein, he has also developed tools based on Eclipse [Foundation, 2015] by working with metamodels based on OMG. Seidewitz [Seidewitz, 2003] proposes an approach that has greatly influenced modeling theory. The distinction he draws between the formal view (model theory) and the non-formal (modeling theory) should be underscored. By starting with a mathematically oriented definition, he addresses the principal modeling issues: descriptive vs. prescriptive, metamodel, modeling language, minimal model, relating them to the OMG approach. Stachowiak [Stachowiak, 1973] and Ludewig [Ludewig, 2003] have also been of great influence in the field as they outlined the three primary characteristics of models: reduction, mapping, pragmatic. Favre [Favre, 2004, Favre, 2005] focuses his discussion on megamodels (models of MDE) and gives equal attention to the central issues of model, metamodel, and modeling language by defining the primary relationships among these concepts. Most notable is his statement regarding the key question of distinguishing between descriptive and prescriptive models: 'who, of the model or the system, have the truth' [Favre, 2004]. Rensink [Rensink, 2005] offers a different approach that defines model as a 'subject' with an associated 'member-of' function. He also formally defines metamodel and modeling language and compares models with ontologies and theorizes on the relationships of similarity (description, representation, instance-of, is-a, subset). Muller et al. [Muller et al., 2012] focus on the intention of a model and then goes on to define a formal notation to demonstrate the relationships between the purpose of a model and the purpose of the modeled object.

4 Models

Undoubtedly, the first concept any contribution to the theory of modeling must address is the very notion of model. Our intention herein is not to propose a ground–breaking new definition of model, but rather primarily refine and complete some aspects that the existing definitions lack. To this end, our approach elaborates primarily on two existing definitions: first, Seidewitz's definition [Seidewitz, 2003], which was later reformulated by Gonzalez-Perez and Henderson-Sellers [Gonzalez-Perez and Henderson-Sellers, 2007]. Both essentially consider a model to be a set of sentences about a system or a subject under study respectively. Inspired by scientific method, we wonder if a group of sentences is sufficient to produce a model, especially because, if this is the case, such a model would only allow a modeler to verify that a certain thing complies with the model, or not. Such a notion of model disregards other aspects, such as that of analysis and synthesis, which we claim to be also fundamental in the notion model, and that are related to the purpose of the model and its creation process. In fact, we think that the aforementioned definitions address a simpler concept, which constitutes the heart of a model, but is not properly a model. That's why we named this concept quasi–model. We should also note that, although we address this concept first, the quasi–model is not the first step in the modeling procedure proposed herein. We have yet to discuss how we obtain a quasi–model, though the reader can probably already infer that the scientific method activities provides the method to do so.

4.1 Quasi–models

Let entity be something that has a distinct, separate existence, though it need not be a material existence, being E the set of all the entities. Let a statement d be an assertion whose compliance can be evaluated for different entities $e \in E$, $d : E \rightarrow B$, being $B = \{true, false\}$; e.g., the sentence 'it has one or more legs' is a statement. We can evaluate this statement for each entity which we want to assess to see if it has one or more legs. Depending on the entity, the statement d is evaluated as true or false.

Definition 1. ***Quasi–model***. We define *quasi–model* as an entity Qm determined by a *validity function* $\mathcal{V}_{Qm}$ that, considering a set of statements DQm, is defined as $\mathcal{V}_{Qm} : E \rightarrow B | \mathcal{V}_{Qm}(e) = \bigwedge_{d \in DQm} d(e)$. We call *quasi–model statements* the statements belonging to DQm.

Definition 2. ***Original***. We name *quasi–model original*, or simply *original*, an entity $o \in E | \mathcal{V}_{Qm}(o) = true$, i.e., an original is an entity for which the quasi–model statements are verified.

Definition 3. ***Domain***. We name *domain* of the quasi–model Qm, the set $O_{Qm} = \{o \in E | \mathcal{V}_{Qm}(o) = true\}$, consisting of all quasi–model originals.

Essentially a quasi–model Qm is an entity associated with a set of statements DQm whose certainty about other entities called originals (o) can be determined by a validity function $\mathcal{V}_{Qm}$. This function, and its constituting statements, is the key element that relates a quasi–model of something to that thing. Consequently, we could say that a certain Qm is the quasi–model of its domain O_{Qm}, and that this domain must be non-empty for the quasi–model Qm to actually exist as such (i.e. so it is consistent).

The statements will include references to certain features of the originals, features that distinguish these originals from other entities that do not belong to the domain. We call these references *quasi–model features*. For example, when making a quasi–model of tables, in a statement like 'it has one or more legs', 'legs' is the quasi–model feature that refers to the originals' characteristic of having legs. As we will see in Section 5, our concept of feature focuses on the modeler view, and it subsumes other approaches like property [Gonzalez-Perez, 2018, Orilia and Swoyer, 2020] or attribute [Gonzalez-Perez, 2018] which can be found in the literature from different perspectives such as the philosophical [Orilia and Swoyer, 2020] or the conceptual [Gonzalez-Perez, 2018] points of view. Each entity to be determined as an original should provide a concrete value for each quasi–model feature, thus allowing a *modeler* to evaluate the statements for that original. Note that, in addition, the statements establish a set of relationships among the quasi–model features that must also be verified in the originals. For example, if we add the statement 'it has a flat board perpendicular to the legs' to the above quasi–model of table, this statement determines a position relationship between the quasi–model features 'board' and 'legs', relationship that also exists in the actual tables to which the quasi–model refers. Some authors (for example [Gonzalez-Perez and Henderson-Sellers, 2007, Stachowiak, 1973, Klir, 2013]) have studied more thoroughly this characteristic of quasi–models (models using their terminology), referring to it as a homomorphism that is established between a model and its domain.

The notion of quasi–model can be illustrated with some simple examples. The first completes the above initiated quasi–model of table. Its validity function $\mathcal{V}_{Qm}$ could be defined with the following statements: d_1='it has a flat horizontal board' and d_2='one or more legs' that d_3='hold the board', being d_4='the legs located under the horizontal surface' and 'perpendicular to it'. All entities that meet these statements, i.e. the evaluation of $\mathcal{V}_{Qm}$ is true for them, are originals of the table quasi–model, and therefore they are tables. The quasi–model features in this case are the legs and the board. Around these features, the modeler builds the statements that check the existence of legs, their position relative to the board, etc. For example, the table on which I am writing this paper is an original of this quasi–model, in contrast to my dog who, despite having four legs, does not even provide a value for the 'board' feature. A second example, taken from the software engineering field, is the quasi–model of date. In this case, the validity function could have more or less statements depending on the desired degree of precision. For the sake of simplicity, and omitting the details regarding the specification of the number of days in February, lets us consider the following: d_1='it has three numbers, d, m and y'; d_2='y is an integer'; d_3='m is an integer ranging from 1 to 12'; d_4='d is an integer that, if m is 1, 3, 5, 7, 8, 10 or 12, ranges from 1 to 31; if m is 4, 6,9 or 11, ranges from 1 to 30; and if m is 2, it ranges from 1 to 28' (notice that, e.g. the number m and its range are two of the quasi–model features). Let's now consider a more mathematical quasi–model: that of even numbers. In this case the statements are d_1 = '$x \in \mathbb{N}$' and d_2 = '$x \bmod 2 = 0$'. Our last example is the quasi–model of the Sun. Our lack of knowledge of astrophysics prevents us from listing the statements constituting the validity function of this quasi–model. However, our intention in presenting this example is not to detail it, but rather, to bring up the fact that certain quasi–models have only one original. We will return to this example when we present the concept of *singleton model*.

Before proceeding, let us clarify an aspect that seems to contradict one of the primary driving principles behind our approach; the independence principle, which aims to independently address the three interrelated concepts of the model-system-language triangle. However, so far, we have only presented the quasi–model notion, and then we have given several examples in which the statements are expressed using some languages, specifically, the English 'natural language' or the mathematical language. Moreover, we even contaminated the previous quasi–model definitions, 'saying' that they are the quasi–models 'of something', by using a word or a phrase that provides a great implicit semantic load. As we discuss later, the relationship between model and language is very complex. Several authors even bind both terms in their definitions of model, by indicating that models are expressed by modeling languages [Seidewitz, 2003, Group, 2016, Aßmann et al., 2006, Favre, 2005, Ober and Prinz, 2006, Mellor et al., 2004]. This interdependence is the source of many inconsistencies and ambiguities, dealt with profusely in the literature on modeling [Rodriguez-Priego et al., 2010, Seidewitz, 2003, Aßmann et al., 2006, Favre, 2005, Ober and Prinz, 2006, Kurtev, 2007], and from which we want to escape. For us, model and language are related but independent concepts, to the extent of considering the possibility that there are quasi-models for which a sufficiently precise language has not yet been found to express them fully. For example, humans can recognize faces in complicated conditions but we can hardly express in a spoken o written language how we realize this task. Using a language to express the statements of the quasi–model examples is nothing but a necessity imposed by the need to present them. We could have even expressed them using UML, and still be referring to the same quasi–models. Likewise, we could have omitted the names of the quasi–models, leaving the reader to identify them, but this would have no influence on the fact that they still refer to tables, dates, even numbers and the Sun.

A quasi–model cannot be considered as a true model yet. Unlike Seidewitz [Seidewitz, 2003], we claim that a model is not simply a set of statements about something. It is our belief that modeling benefits from the application of scientific method activities. The statements of a quasi–model are related to the verification aspect of scientific method, which means that the quasi–model can be used by the modeler only to check if a certain entity is an original of a quasi–model. Nevertheless, just as formulating the hypotheses is not the first step in scientific method, quasi–model statements do not represent the first step in modeling in our proposal. Though for expository reasons we have described quasi–models first, they actually come about during a process of analysis, analogy or synthesis (again the parallels with scientific method are evident). These processes are successive approximations that conduct iterative validations until the desired result is obtained. We claim that, and this is one of our main contributions, analysis, analogy and synthesis, or at least one of them, must be included in the definition of model, along with the aspect of validation. By doing so, we delve into the notion of utility or purpose of a model. What good is a model if it cannot be used to describe and/or to prescribe? It is our understanding that these descriptive and prescriptive aspects of modeling are strongly related to the aforementioned scientific method processes of analysis, analogy and synthesis.

4.2 The Analytical process: Analytical models

Analysis is the process by which a modeler develops a set of statements that characterize a quasi–model, while also giving rise to the corresponding model, as we will soon discuss. Thanks to this *analytical process*, the modeler selects a set of entities, the sample, and, by means of an abstraction process, draws from them certain common features of interest to him. These features allow the modeler to propose, by means of an induction mechanism, the set of statements that constitute the quasi–model validity function. Bear in mind that the modeler does not usually obtain the set of statements 'in the first attempt', nor does he state them randomly. Instead, observing the sample elements iteratively, the modeler develops *hypotheses* about those elements. As Favre indicates [Favre, 2004], it is worth noting that in the analytical process, the sample elements 'represent the truth'. This means that, if a certain hypothesis is found to be false for any of the sample elements, it must be discarded and, if possible, replaced by another that could be true. If the sample is a finite set, the analytical process ends when the modeler has checked that all hypotheses are correct with respect to every sample element. If the sample has an infinite set of elements, the modeler selects a representative and finite subset of the sample using it to validate the hypotheses. In both cases, at the end of the iteration over the sample, the hypotheses become the statements of the validity function of a resultant quasi–model, and consequently the sample elements can be named originals. An important point is that the modeler not only chooses the hypotheses, but also the sample for which the quasi–model is to be developed. This means that, not only must each and every hypothesis be true for all the sample entities, but at least one hypothesis must be not true for the other entities not belonging to the model domain. Otherwise, the hypothesis would be too general to define the quasi–model and more restrictive ones should be ascertained.

Although the described analytical process represents the general way of analyzing entities to build models, modelers apply it differently in each case. For example, the observation of a set of faces by a dentist results in a face quasi–model with many features regarding the mouth and the teeth. The same set of faces observed by an ophthalmologist could result in a facial quasi–model with extremely detailed eyes. It all depends on how the modeler perceives the sample. Another example, which is obtained from computer science areas, is that of considering how a certain relational schema can be processed by different reverse engineering tools to find conceptual quasi–models made up of either related classes (typically expressed using UML) or entities and relationships (usually expressed using ER). While there is always an analysis of a given sample, the way the analysis is conducted is different in each case. That is why different sets of statements are obtained. The modeler chooses the hypotheses about the sample and, in doing so, and this must be emphasized, the modeler intrinsically associates the set of statements with the details of its analysis process. In fact, we claim that the analysis process has led the modeler to something richer than a mere quasi–model, i.e. a *model*, which, in addition to the developed statements, includes the peculiarities of the analysis process as one of its constituent parts. We call the result of this process *analytical models*.

Definition 4. ***Analytical model***. We define *analytical model* m as an entity characterized by (1) the specific iterative process, which we call *analytical constructor* (and denote AC_m,) by which a modeler states, selects and checks the validity of a set, D_m, of statements d (initially called *hypotheses*) about a set of entities (called *sample*), and (2) the *validity function* $\mathcal{V}_m$ defined as $\mathcal{V}_m : E \to B | \mathcal{V}_m(e) = \bigwedge_{d \in D_m} d(e)$ that results from that iterative process. We name *model statements* those statements (all the previously introduced terms for quasi–model can be ported to model: *model originals*, *domain*, *model features*).

Note that an analytical model has only one analytical constructor AC. This restriction stems from the fact that a model is built in a unique way, where this building procedure is an intrinsic part of the model. This does not prevent another modeler, using a different AC and even a different sample, from creating the same set of statements (same $\mathcal{V}_m$). However, for us, it is a different model, because it was built differently. For example, a database analyst could design a database model to store people data (name, address, etc.), simply observing the world and selecting the characteristics of interest to build a relational table with a column for each characteristic (AC_m). Similarly, a hypothetical automatic tool could obtain the same table by processing automatically the information contained in a city's public census (a different AC_m). From the SMM point of view, although both tables, which constitute the statements of each model, are the same, here we have two different models, since another constituent part of them, such as the AC_m, is different (they have been built differently).

The association of the set of statements and its corresponding AC_m under the notion of model will allow the fine-tuning of that resulting model. As we have commented above, for analytical models, the entities of the sample hold the truth. Suppose then that, after the model development, the modeler discovers new entities that should be originals of the model but do not meet its validation function. This is a symptom of the model not being sufficiently accurate. It must be reanalyzed, by running its analytical constructor again, which remember, captures the specific details of the analysis process of the model. This new analysis of the extended sample produces a new set of statements for the validity function of the new model, which should now take into account the new entities. Similarly, if an entity that meets the model's validation function,

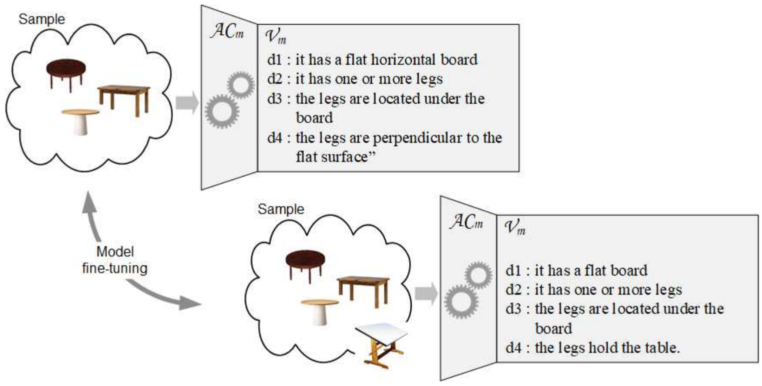

Figure 2: Representation of an analytical model (the model of table). We graphically represent an analytical model using a polyhedron, which resembles a funnel that represents the analytical constructor, attached to a rectangle that represents the validity function (containing the model statements). The sample passes through the funnel to build the model statements. The figure also illustrates the model fine-tuning process by which the model statements are changed to consider new originals.

but that should not be considered a model's original, appears, then the model should be reanalyzed again to reformulate the statements to make them more restrictive. Another possibility is that the originals change, such as in the case of the influenza virus model, which constantly mutates. This mutation forces the reanalysis of the model, so that it remains valid for the development of effective vaccines. The reanalysis can be conducted using the same techniques as in the previous model (running the same analytical constructor), or others, if the former are no longer adequate. In any case, a new model is obtained and, interestingly, the former runs out of physical originals, but retains the abstract ones (in Section 5 we discuss the physical or abstract nature of originals).

Let us revisit the previous section's examples to illustrate the AC notion. Those quasi–models correspond to analytical models, i.e., they were created using certain analytical constructors that we can specify. For the first example, the modeler starts from a set of tables in different shapes and with different number of legs (see Figure 2). The modeler abstracts certain features of this sample such as the board, the legs and their position with respect to the board, and so on, while discarding others such as color, dimensions or properties of the used materials. After obtaining the set of statements described in the example, the modeler may realize that the legs of some tables, not initially included in the sample, are not perpendicular to the board, a fact that would lead him to modify the model statements (the process is illustrated in Figure 2). Consequently, a new model of table is obtained. Similar specifications can be given for the other examples.

The analytical constructor is extremely important to understand the difference between a quasi–model and an analytical model. However, the quasi–model alone is commonly identified as the analytical model without paying attention to its constructor. One of the reasons for this is that, whereas the quasi–model statements could be easily expressed using a language (perhaps a modeling language), the analytical constructor is often difficult to express. The analytical process is a mental process as innate to humans as other mechanisms of deduction, inference and relation. Everyone is born with the ability to analyze, but sometimes we cannot explain how we do it, or why we draw certain conclusions and not others. In some cases, as in the previous one of tables, it is easy to describe. In others, such as the case of the aforementioned reverse engineering tools, the description of the process is given in the form of software or it can be tackled through approaches such as machine reasoning [Bottou, 2014], model building and pattern recognition [Lake et al., 2017], by transforming object diagrams into class diagrams [Kästner et al., 2018] or by the use of machine learning techniques for data analysis to get interpretable models [Vellido et al., 2012]. But other times, when a model must be communicated, we can easily transmit the statements, but conveying the details of how to perform an analysis is extremely difficult, and so examples are provided as a sort of approximate specification. E.g., we can tell a little child several times the statements corresponding to a model of an adjustable wrench, but we will most likely fail until we show the child an example. These examples help to explain the relationship between model statements and model originals. The receiver of the communicated model realizes this process, which is also analytical and therefore becomes the AC_m of his the received model.

4.3 The Synthetical process; Synthetic models and Full models

So far, having dealt exclusively with analytical models, the only evident utility of a model is to use it instead of the set of originals it represents, as a description of its domain, or to study (validate) certain properties of those originals. This is the case of the aforementioned model of the Sun. However, this is only one of the uses that a model can have. Models are also useful for building plans for the synthesis of new entities that did not exist when the model was created, just the opposite approach of analytical models.

Synthesizing involves utilizing certain raw products that are processed in a certain way, giving rise to a new entity. Similarly to the analytical case, a certain *synthesis process* may require an iterative method to be fully obtained. After a first version of the process that allows preliminary entities to be synthesized, these are verified by the modeler to check if the result is completely satisfactory.

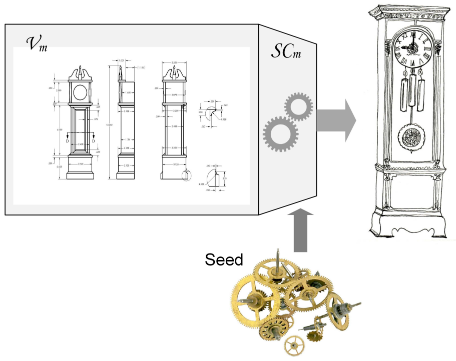

Figure 3: Representation of a synthetic model (the model of a clock). The graphical representation of a synthetic model includes a polyhedron, which resembles a funnel that represents the synthetic constructor, attached to a rectangle that represents the validity function (containing the model statements). The figure also illustrates the seeds that feed into the synthetic constructor to build a new original.

The results of this verification can be used in an iterative fine-tuning of the synthesis process. Here, the principal element is the final synthesis process itself, which embodies a model of non-analytical nature, because it does not have an initial sample to analyze and its objective is not to describe, but to generate new originals (prescribe). All generated originals will have something in common, which are nothing more than the model statements, which could now be interpreted as the post-conditions of the synthesis process associated with the model. These statements constitute the model validity function, which must be met by every generated original (otherwise they would not be model originals). The question as to how a modeler obtains the first version of the entities to be synthesized must still be addressed. Sometimes they are the product of invention: a modeler imagines an original of the model to be constructed and analyzes it to obtain a first version of the model statements to be created and which will be refined later on through successive validations. Other times, it is another scenario of application of scientific method. As Gower explains in [Gower, 2012] synthesis often begins with a prior process of analyzing existing propositions, which then generates the first hypotheses to be iteratively refined. In modeling, this would consist of applying general theories or recommendations, or even more abstract models, which, once particularized, would become the preliminary version of the synthetical process to be created. In our proposal, these mechanisms are based on the notion of metamodel, which is addressed in subsection 7.2.

Definition 5. ***Synthetic model***. We define *synthetic model* m as an entity characterized by (1) a set of at least one *synthetic constructor*, denoted SC_m, each of which is a process whose purpose is to create new model originals from sets of other entities called *model seed*, and (2) a *validity function* $\mathcal{V}_m$ defined as $\mathcal{V}_m : E \rightarrow B|\mathcal{V}_m(e) = \bigwedge_{d \in D_m} d(e)$ where D_m is a set of statements that must be met by every generated original.

There are many examples of these models, that we call *synthetic models*, such as the design of a new clock, when a clockmaker specifies its characteristics and the steps to build it (Figure 3) or similarly, a recipe that explains how to prepare a dessert. Note that we cannot use any arbitrary set of entities to generate an original. Each synthetic constructor will require that its seeds meet certain requirements that make them suitable for the construction process. Thus, the model of clock will require the necessary pieces to build the new clock, and the dessert recipe will specify the sort and precise amounts of the ingredients used in its elaboration. Similarly to the analytical constructor, the internal formulation of a synthetic constructor may be more or less complex, and more or less accurate. We will encounter cases in which the formulation lacks precision and is a rather vague building guidance, and cases that provide a rigorous algorithm implementation.

Models are not always synthetic models. The Sun model we presented above is a pure analytical model: we use it only for descriptive purposes. Likewise, there are purely synthetic models such as the previous model of dessert recipe. However, it is not uncommon that a modeler who develops an analytical model enriches it with a synthetic constructor. This could be considered another result of the AC_m, along with the model statements. If we consider the examples presented in subsection 4.2 again, we can see that this could be the case for some of them. Thus, we could specify a synthetic constructor able to generate originals of the model of table. In this case, a suitable seed would be, for example, the combination of 'a square board of 1 m^2' and four units of '80 cm stick'. The synthetic constructor generates an original of table, placing the board on the legs, in such a way that all the previously specified statements of the table model are met. Note that we do not mean that the constructor has to build the table itself. To our mind, both models and originals are abstract entities, though they may be related to real physical things. We will postpone this discussion until a later section about the physical or abstract nature of originals and models. For now, it is enough to say that the synthetic constructor of the model of table is not a machine, but a mental process that allows model users to imagine tables. Skipping to the next example, we could also specify at least two synthetic constructors to generate originals of the model of date. One of them needs a seed made up of three integer numbers, where the second and the third must be valid month and day numbers. The other SC_m takes a single integer number representing the number of milliseconds since a certain reference in time (such as

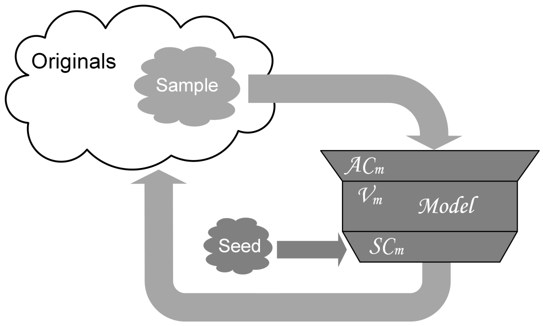

Figure 4: Graphical and schematic representation of a full model.

January 1, 1970, 00:00:00 GMT). Both constructors enrich the previous model of date, thus it is no longer only analytical.

Models presented in the previous paragraph are simultaneously synthetic and analytical. Their originals existed before creating the model, but new originals can be constructed using their added synthetic constructors, as Figure 4 illustrates. This enrichment of models has been treated by other authors [Gonzalez-Perez and Henderson-Sellers, 2007, Rothenberg, 1991]. In SMM, we call this last type of models *full models*.

Definition 6. ***Full model.*** We define *full model* m as an entity characterized by an analytical constructor AC_m, a non-empty set of synthetic constructors SC_m and a validity function $\mathcal{V}_m$ defined as $\mathcal{V}_m : E \rightarrow B | \mathcal{V}_m(e) = \bigwedge_{d \in D_m} d(e)$ where D_m is a set of statements that must be met by every model original.

4.4 Models and analogy

So far, we have shown how modelers can create analytical or synthetic models. However, we claim that another common way of modeling is by analogy [Gentner, 1983]. The process is similar to the previous analytical process, but with one important difference. Now the starting point is not a sample of entities that become model originals, but another model whose originals have analogue properties with the originals of the model to be obtained. Consider, e.g., the model of airplane. The different models, which eventually led to the airplane, took birds as a reference to develop the model. The result was that the airplane model was analog to the bird model (bird's wings - airplane wings, bird's body - airplane body, bird's legs - airplane landing gear, etc.). The obtained model was not analytical since, though it was produced by means of an iterative process similar to the analytical process, its sample did not exist when the model construction began. It was not a model of existing originals (the bird model did exist, but the

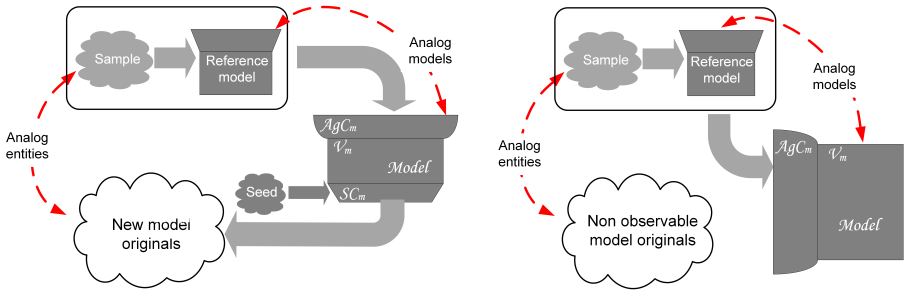

Figure 5: Two analogical models, the first is also synthetic. The figure shows that the input to the analogical constructor is another model (reference model), previously existing or purposely built from a sample of analog entities. Note that, unlike analytical models, this sample, from which model construction starts, does not belong to the set of model originals. Instead, the sample entities are analog to the model originals.

airplane model originals did not exist prior to the airplane model). It is therefore a kind of model that by analogy describes and, in this case, synthesizes originals that are similar (analog) to the originals of the analog model to which it refers.

Definition 7. ***Analogical model***. We define *analogical model* m as an entity characterized by (1) the specific iterative process, which we call *analogical constructor* and denote AgC_m, by which a modeler selects and checks the validity of a set of statements, initially called *hypotheses*, stated by positive analogy *[Hesse, 2000]* to another model called *reference model*, and (2) the validity function $\mathcal{V}_m$ defined as $\mathcal{V}_m : E \rightarrow B | \mathcal{V}_m(e) = \bigwedge_{d \in D_m} d(e)$ that results from that iterative process.

Note that analogical models are usually synthetic models, since, as in the previous case of the airplane, they are developed to build completely new originals based on other existing analog originals. However this is not always true. The analogy is also used when, despite the existence of entities to be modeled, the modeler cannot directly observe them. Thus, there is no feasible set of potential originals that can be utilized as the sample for the analytical process, and the modeler has to resort to an analogous model for the hypothesis statement. For example, this is the case of the Rutherford's atom model, which could have been created from the reference model of a planetary system [Gentner, 1983, Goldstone and Son, 2005, French, 2002] and whose validation function must use indirect tests to check the validity of the hypotheses regarding the sample.

It is important to note that in this case the link between the model and its

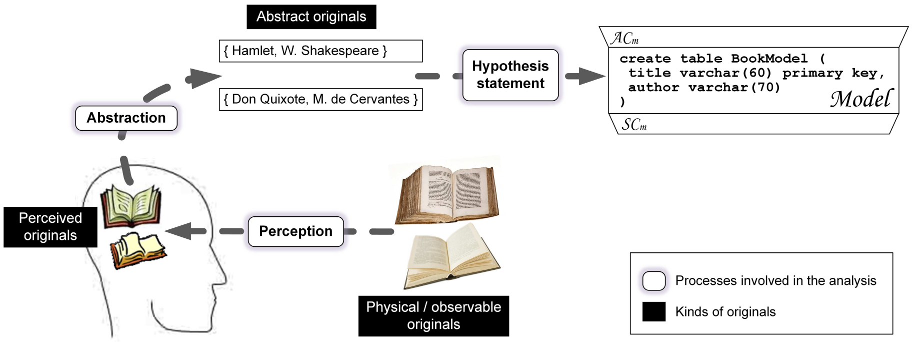

Figure 6: Process of analysis that leads to a model of a book. The figure illustrates the different types of originals, from the physical/observable to the final model. The AC of this model results from the application of the relational database theory to create a database schema that contains data about books. This model has a SC that generates new relational tuples corresponding to new book originals.

domain is not created by means of the analytical constructor, but indirectly via the analogical constructor and its reference model. When this happens, analogical models are utilized in the same way as analytical models are: as a description, by analogy, of their domains.

5 About the physical or abstract nature of modeling

The distinction between the physical and the abstract is one of the most confusing issues in the modeling literature. Usually, when researchers talk about originals, real and abstract entities are conflated. But they are not really the same. So far we have talked about samples of originals, seeds and models, assuming, as does [Klir, 2013], that these terms belong to a conceptual layer, and therefore they are abstract, this being consistent with the fact that we use them to refer to real entities. The question is: what is the relationship between the concept (original, model, seed) and the actual entity which it refers to?

5.1 Nature of originals

When analyzing (executing a model AC) we start with a real observable entity, physical or not (see Figure 6). The modeler observes it, obtaining, in the first step, an 'image' of it provided by his sensors. If the modeler is human, this image is a mental image, and the sensors are the senses. If the modeler were a machine, the representation of the original would be digital. The obtained image depends on the capabilities of the modeler sensors. E.g., a person who observes a flower through his eyes gets a raw image of it, whereas the same physical flower observed by a blind person through his nose leads to an 'olfactory' image. It also depends on the modeler's will or need to perceive more or less entity details. To develop a certain model, a modeler might be content with a cursory glance of the entities, or require a more precise observation to create a more complete model of them. We call this image *perceived entity* as opposed to the actual entity we call *physical/observable entity* (we use the terms physical or observable interchangeably, the latter being more general). Then, based on that raw and meaningless set of characteristics obtained by the sensors that constitute the perceived entity, the modeler, through a process of simplification, chooses those features deemed relevant to his model (the *model features*) giving rise to what we call *abstract entity*. The analytical process continues by stating certain hypotheses about this and other abstract entities that belong to the sample, as we explained before. If the process succeeds, the hypotheses become the statements of a model; each abstract entity becomes an *abstract original*; each perceived entity becomes a *perceived original*; and the observable entity becomes a *observable original*. We do not consider the transition from the observable to the perceived original to be an abstraction, because the loss of details that occurs in this step is due solely to the sensors' inability to capture all the physical characteristics of the observable original. Note that for us and other authors [Sykes, 2003, Gupta and Sykes, 2001, Klir, 2013], the process of abstraction starts from the perceived original, because the modeler cannot mentally work with the physical original. Thus, the perceived is the one that merits consideration as the true original.

The situation is similar when it comes to the synthesis. Here, the starting point to obtain the final original is a synthetic model and a set of seeds. Thanks to the synthetic constructor, the modeler can imagine a new original produced from the model, obtaining a mental image which is similar to the obtained from the perception of observable originals. Therefore, this new original belongs to the same category as the perceived ones, though we do not name it as such, but *conceived* original. The previous considerations about the originals can apply again regarding the seeds. The seeds used in the synthesis are not physical, but abstract seeds, which correspond to perceived seeds. E.g., from the previous model of table, a synthesizer modeler could imagine a new table, by giving value to the features of the model (four concrete legs and one board), then obtaining an abstract original that leads to a conceived original. The fact that an abstraction still exists between the model and the conceived original is due to the perceived seeds adding more information to the mere abstract original (the material of the legs, the color of the board, etc.). The modeler is definitely obtaining an enriched mental image of the new table he is synthesizing. Similarly to analytical models, the true original in this case is the conceived one, which can become a physical original if the synthesizer uses a device, complementary to the sensor, called *actuator* [Kelly and Tolvanen, 2008] (e.g., a production machine).

The last consideration concerns the physical or abstract nature of the models. Are there physical models? From our definitions it can be deduced that models are always abstract, which does not mean that they cannot 'reside' on physical entities (indeed, they must do so in order to be useful). In fact, models reside in the modeler. The two most common examples of modelers, humans and digital systems, illustrate this aspect. Therefore, it is incorrect to talk about physical models, but rather we should refer to physical modelers (human or digital).

5.2 Singleton models

The confusion caused by not distinguishing between the above mentioned types of originals can be made worse when we deal with what we call *singleton models*: models that have only one (perceived/ conceived) original. In this case it is possible to confuse the model with the abstract original, as there is a one-to-one relationship between them. SMM clarifies this confusion. An abstract original cannot be a model because it is mainly a set of features, with value, relevant to the original, which are not associated to any validation function, nor an analytical, analogical and/or synthetic constructor. The above mentioned model of the Sun is an example of singleton model.

One might think that singleton models can only be analytical, because, if we give them a synthetic constructor, they could generate more originals, and therefore cease to be singleton. Revisiting the aforementioned synthetic model of clock (Figure 3), we realize that it is a singleton model that, using a set of seeds, prescribes how to create a single conceived clock (always the same conceived clock). From that conceived clock, an actuator could generate multiple physical originals of the clock. But for us, the relevant original is the conceived, which is unique.

Other authors have given other names to singleton models. Kühne [Kühne, 2006] talks about 'token-models' (as opposed to 'type-models'), [Hesse, 2006] names them 'feature projection model' (vs. 'placeholder projection models') and [Gonzalez-Perez and Henderson-Sellers, 2007] employs the term 'isotypical mappings' (vs. 'prototypical/metatypical mappings'). Kühne concludes that these models have different characteristics from the rest. We claim that no such differentiation exists. The singleton models are standard models, with the proviso that they have a single original.

6 Discussion about models

Just as we set as our objective in the introduction, our Definition 6, full model, already refers to three scientific method characteristic activities: analysis, synthesis and validation. Our definition includes the definitions proposed by Seidewitz and Gonzalez-Perez and Henderson-Sellers [Gonzalez-Perez and Henderson-Sellers, 2007, Seidewitz, 2003], except that in our case a model is not just a set of statements about something. Mere statements do not fully capture all the aspects with which the analytical, synthetic and analogical constructors contribute to our definition of model. To our mind, the constructors are essential in our notion of model, since they establish a permanent binding between a model and its originals, whether they are derived from a synthesis, and/or used as a basis for the model analysis. An example that shows this fact in a simple way is the notion of class in an object-oriented programming language. An OOP class (to be called such) needs at least one constructor. Without this constructor, an OOP class would be just a quasi-model, that could not generate its instances (originals in SMM terms). Note also that a complete OOP class is a remarkable example of synthetic model.

Our classification (analytical, synthetic, analogical and full) allows to revise under a new perspective what the literature calls backward/descriptive and forward/prescriptive models [Gonzalez-Perez and Henderson-Sellers, 2007, Seidewitz, 2003, Favre, 2004, Bézivin, 2005, Sánchez et al., 2009, Jouault et al., 2009]. Pure analytical/analogical models are used to describe a set of originals (the model domain), where the model can represent each original. Synthetic models are used to build new originals. In other words, in analytical/analogical models (backward/descriptive), originals exist before and, consequently, they prevail over the model. Conversely, in synthetic models (forward/prescriptive), the model mandates how to generate originals. Thus, it seems reasonable to suppose that, in the case of analytical/analogical models, if the model changes, it is because the originals/the reference model have changed before (they have changed, or more significant originals have appeared in the initial sample). In the case of synthetic models it is just the opposite, if the modeler changes the model, the generated originals will change accordingly (or they will no longer be model originals). Another property commonly ascribed to a model is its representational character. It is common to say that a model is a representation (even with the added character of 'substitution') of something [Gonzalez-Perez, 2018]. Under our interpretation, this is not an intrinsic characteristic of the concept of model, since a representation can only be of something pre-existing. Therefore, it could occur in the case of analytical models, in which the originals exist prior to the model, which, once created, could be used to represent its originals. An analogous interpretation is possible in the case of analogical models, but not in the case of synthetic models.

It is important to note that both original and model are interdependent concepts, i.e., one cannot exist without the other. For an entity to be considered as a model it must be related at least to one original, this being possible either because the model can be obtained using its analytical constructor, or because the original is built using the model synthetic constructor. This does not mean, as other authors say [Favre, 2004, Bézivin, 2005, Asikainen and Männistö, 2009], that, depending on the way it is used, an entity can be an original or a model. It does not mean either that 'everything is a model', as [Bézivin, 2005] states. We do not agree with these points of view. Originals and models should not be confused. Only models have analytical or synthetic constructors and validity functions. Only when we deal with models of models (metamodels) can a model be deemed an original, and vice versa. What we can say is that the statement 'everything is a model' should be rephrased as 'everything can be a physical/observable original of a model'. Similarly we could say that 'everything can have its associated singleton model'.

SMM is not even remotely based on the object-oriented modeling theory (hereinafter OOMT). SMM models may use object-orientation concepts or not, but object-orientation is not the basis of the approach. SMM does not need the class concept; instead, the model concept is used directly. There is no replication of concepts either [Atkinson and Kühne, 2001, Atkinson and Kühne, 2008], as there is only a general model definition. SMM does not require every entity to be an original of something. An entity can be easily identified as an original of its model by means of the validation function; and the mechanisms for creating a model for a set of originals, or an original of a certain model, are clearly specified by the model constructors.

In our opinion, SMM improves understanding of some approaches such as multilevel modeling [Atkinson and Kühne, 2001, Atkinson and Kühne, 2002, Balaban et al., 2018, Lara et al., 2014, Atkinson et al., 2015]. Although SMM does not require a layered model structure, different architectures can be defined as long as the basic rules of our proposal are followed. For instance, a model whose originals simultaneously include other models and the singleton models of the originals of the other models is valid in SMM, as long as a validity function and some constructors for such a model can be defined. A new study would be necessary to fully analyze the various multilevel modeling proposals according to the SMM perspective. Thus, we propose this task for future research.

SMM goes further by considering not only the relationship between models and original, but also the different types of originals: observable (physical entity), perceived/conceived and abstract. This distinction helps avoid confusion between the different cardinalities produced between models and originals. For example, according to SMM, token models proposed by Kühne [Kühne, 2006] are not actually models (they lack constructors), but rather abstract originals. In OOMT, due to the synthetic nature of the class concept, the construction of class-based models always leads to synthetic models from the perspective of SMM, though these instances cannot correspond to what SMM calls original (read, perceived/conceived original), but rather, to what it calls abstract original. SMM is more general, since it provides for the existence of purely analytical models. Moreover, in SMM, a model can be bidirectional (full model). In a full model, changes in the originals affect the model, and vice versa.

7 The notions of Submodel and Metamodel

We now address two essential issues that have been broadly discussed by many modeling experts: submodeling, including inheritance as a particular case of it, and metamodeling [Kühne, 2006, Gonzalez-Perez and Henderson-Sellers, 2007, Seidewitz, 2003, Kühne, 2009].

7.1 Submodels and inheritance

The relationship between modeling and inheritance has become a central issue in the literature [Atkinson and Kühne, 2002, Kühne, 2009, Atkinson et al., 2000, Kühne, 2010, Overbeek, 2006, Aßmann et al., 2006, Bézivin, 2005, Shan and Zhu, 2008, Büttner and Gogolla, 2004, Merunka, 2003, Barbero et al., 2007]. The point is how inheritance can be interpreted from a pure theory of modeling point of view. As we see below, inheritance is a special case of a more general concept: submodeling.

Definition 8. ***Submodel***. It is said that a model m' is a submodel of $m \Leftrightarrow O_{m'} \subseteq O_m$, $m, m' \in M$, being M the set of all models, and O_m the set of originals of m. On the other hand, m is said to be a supermodel of m'.

A trivial example is the model of a piece of furniture, one of whose submodels is the above mentioned model of table.

Definition 8 states that everything that is true for every original of a model must be also true for every original of its submodels (but not necessarily vice versa). This can be expressed in terms of the validity functions of the models in such a way that if m' is a submodel of m, $\forall o \in E$ if $\mathcal{V}_{m'}(o)$ then, $\mathcal{V}_m(o)$. Note that this does not mean that the statements of model m (Dm) must be included in the statements of model m' (Dm). In fact, Definition 8 says nothing about the conditions to be met either by the statements or the constructors of both models. Any existing relationship between these elements of both models leads to particular cases of submodeling. E.g., we can find that SCm and SCm' can be independent processes, and cases where SCm is used by SCm' as a subprocess, (e.g., that is the case of nested calls to superclasses constructors in inherited classes in an object-oriented programming language). ACm and ACm' can also be related in a similar way, e.g., when the analysis peculiarities of ACm are included in ACm'. However, the most typical case in practice is when the statements of a model m are a subset of the statements of another model m'. We claim that this corresponds to the traditional inheritance, making clear that, if the statements are included, then $\mathcal{V}_{m'}(o) \rightarrow \mathcal{V}_m(o)$, $O_{m'} \subseteq O_m$, and therefore m' is a submodel of m. Our proposal is consistent even under an interpretation of inheritance as a mechanism, in which one subject receives ('inherits') the properties of another. Since the statements of a model include the properties of their originals, in the case that all the statements of a model are included in another model, the properties of the first one are 'inherited' in the second one. However, we insist that to deal with submodeling, as we understand it, the statements inclusion is not required. We can point to cases for which the verification of the validity function of a model imply the verification of the validity function of another (super)model without the statements inclusion being necessary. Examples demonstrating this abound, though usually they are not interpreted in this way. There are cases in which the model statements do not include the features of the supermodel, as they are unnecessary or superfluous. However, in the model originals level, these features are implicitly contained. E.g., consider the cases of mammals and football players (obviously a football player is a mammal, although it would be unusual for a modeler to include such feature in an analytical model of football players).

7.2 Metamodels

In essence, and unlike other authors who give it a central role in their theories of modeling [Bézivin and Gerbé, 2001, Mellor et al., 2004, Gitzel and Hildenbrand, 2005, Gašević et al., 2007, Terrasse et al., 2006] or identify it with a modeling language [Gonzalez-Perez and Henderson-Sellers, 2007, Seidewitz, 2003, Group, 2017, Aßmann et al., 2006, Favre, 2005, Ober and Prinz, 2006, Kurtev, 2007], for us a metamodel is nothing but a model whose originals are also models.

Definition 9. ***Metamodel***. It is said that a model mm is a metamodel $\Leftrightarrow \forall o \in O_{mm}, o \in \mathcal{M}$, being $\mathcal{M}$ the set of all models. We name the originals of the models belonging to O_{mm}, *metaoriginals* of mm.

Metamodel examples are not difficult to find, even referring to aspects of everyday life. Thus, we could propose a metamodel for recipes, being the aforementioned dessert recipe one of its originals. This metamodel should specify how to develop a new recipe, and its statements will refer to the usual features of recipes: ingredients, method, plating, etc. This is a full metamodel, made by analyzing the structure of a sample of existing recipes (this is the metamodel AC); it can be used also as a framework for issuing new recipes (metamodel SC). Also, in software engineering we often find metamodels. Consider, e.g., the Entity-Relationship model [Chen, 1976], which, despite its name including the

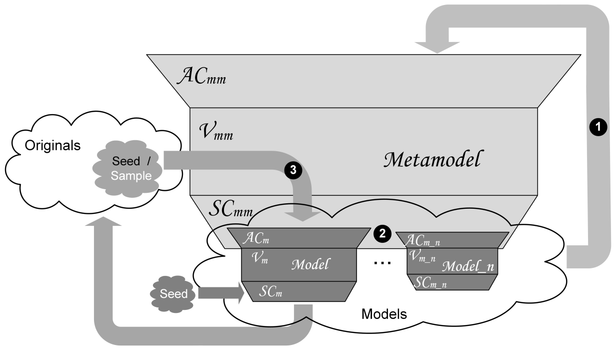

Figure 7: Illustration of how (1) metamodels can be obtained from models by applying the metamodel $ACmm$, and how (2) new models can be synthesized from a metamodel running its $SCmm$. When generated models are analytical, each $SCmm$ execution is equivalent to the ACm of the generated model. In fact, the $SCmm$ execution requires a seed made up of entities that can be interpreted as the sample that feeds (3) the ACm of the generated model. All the models synthesized from the same metamodel, can be used to refine the definition of the metamodel through its analytical constructor (1).

word model, is considered by the community as a metamodel since it serves to generate ER models.

Note that Definition 9 could be iteratively applied, being possible that the originals of a metamodel mm were in turn metamodels, i.e., mm would be a meta-metamodel. Continuing the above example, a model of manufacturing processes of any kind (recipes, clothing, handicrafts, etc.) is a meta-metamodel of our dessert recipe. In the literature, this succession of metamodels is studied using the notion of degree of a metamodel, which leads to theories of modeling based on levels (e.g., the OMG M0 to M3 levels [Group, 2017] or the EIA/CDIF levels [Flatscher, 2002]).

The analytical constructors of metamodels ($ACmm$) are like those of models in general (Figure 7, path 1). Sometimes, the analytical process described above will be applied to a sample of existing models, trying to obtain general commonalities between them. E.g., this is what happens, when, with the intention of creating a new UML-profile, a set of models expressed in UML is taken to analyze their characteristics. Other times, the metamodel is not analytical but synthetic. The author of the metamodel, through a process of creative thinking, states the first version of the metamodel and uses it to generate a set of originals-models that are re-analyzed to refine the initial metamodel (as explained in the Synthetical process Section). This could be the process followed in the above example, whereby Chen devised its Entity-Relationship metamodel.

Thus far, we have consciously articulated the notion of metamodel in such a simple way to convey to the reader the idea that a metamodel is not much more complex than any other model. In fact, we do not intend to claim any originality with Definition 9 which is shared with other references [Kühne, 2006, Group, 2014, Gonzalez-Perez and Henderson-Sellers, 2007, Aßmann et al., 2006]. However, SMM does contribute to metamodeling theory with the discussion about certain special characteristics of metamodels that shed light on how models in general are generated. To this aim, SMM draws on the validity function (statements) and the constructors of metamodels, studying their relationships with the constructors of the models that are originals of those metamodels.

In this regard, SMM states that, being a metamodel a model whose domain is made up of models, it is a requisite of any metamodel, and concretely of its statements, to address, among other things, the constructors of the models that correspond to the metamodel. Consequently, the statements of a metamodel that leads to analytical models must specify a sort of meta-analytical constructor (not to be confused with the metamodel analytical constructor, $ACmm$), a kind of framework for the ACm of the models that belong to the metamodel domain; i.e., it must specify the way a modeler analyzes a given sample to generate a certain model of that metamodel. In short, it must indicate how to model. Similarly, if the metamodel generates synthetic models, it should clarify how its originals, which are models, generate their originals, or meta-originals from the point of view of the metamodel (meta-synthetic constructor). Note that this is a distinguishing feature of metamodels not present in other models.

We recognize that the reader might find it difficult to apply this last claim to the commonly accepted set of software engineering metamodels. These metamodels are indeed described, but their specifications do not refer to the ACm and SCm of the models of its domain. Nevertheless, it is a fact that modelers are able to construct models corresponding to those metamodels, e.g. Chen's ER, and once constructed, modelers know how to create originals of those models. If they are able to do so it is because the metamodel must have specified how. To explain this apparent dilemma we must consider that, frequently, the meta-analytical and meta-synthetic constructors to which we allude above are not explicitly expressed in that metamodels. On the contrary, these meta-constructors are conveyed to the modeler using examples that, e.g. in the case of the meta-analytical constructor, specify inputs (domains to be modeled) and outputs (models according to the metamodel). E.g., in the description of Chen's ER metamodel, or even in the Codd&Date relational model, or in the metamodel associated to UML abstract syntax, there are many examples that illustrate the essential features for each of these metamodels and how they are used to model (examples of uses of entity, attribute...; or examples of relation, tuple...; or examples of class, actor...). Note that these vague specifications give freedom to modelers to analyze/model in different ways, this being the reason for different modelers using the same metamodel to model the same domain differently. The above does not impede other metamodels from precisely specifying how their meta-constructors are. As an example, we may consider a reverse engineering program utilized to obtain a conceptual model from the study of a certain relational database schema. The program developer has in mind a metamodel of these conceptual models that are going to be generated. This metamodel specifies its meta-analytical constructor so precisely that it allows him to write a computer program that, each time it is executed to analyze a DB schema, generates a model according to the metamodel. In fact, that concrete execution of the reverse engineering program is the ACm of the generated model.

We must reflect on the synthetic constructor of a metamodel ($SCmm$), which is the process by which the models of a metamodel are generated. When a certain $SCmm$ generates analytical models, and only in this case, there is a remarkable relationship already suggested at the end of the previous paragraph. Each time a modeler runs that metamodel $SCmm$, a new analytical model is created (its statements and constructor). Then, part of an execution of that $SCmm$, the part that creates the model statements, is, by definition (Definition 4), the ACm of the new model (situation illustrated by Figure 7, bullet-point 2). In addition, the seed that feeds each execution of the $SCmm$ can also be viewed from the analysis perspective. Due to the generated model being analytical, the seed must be a set of entities that plays the role of sample of originals for the ACm of the new model (Figure 7, bullet-point 3). Summing up, we can say that when a modeler creates a model by analysis under the influence of a metamodel, he is running the $SCmm$ of that metamodel. On the other hand, when the generated models are synthetic, there is also a relationship between these SCm of the models and the $SCmm$ of their metamodel. Notice that the latter must generate, besides the statements, the synthetic constructors SCm of the created models, so that they can generate their originals.

We cannot strictly say that all analytical models are always generated running a $SCmm$. However, intuition suggests so. It is strange to think that a modeler does not base his way of analyzing on some pattern (metamodel). Actually, the features of a metamodel (see Section about Quasimodels), which embody the metamodel statements, represent the features we want to extract from the sample when we analyze. These meta-features guide the process of analysis. For instance, in the case of the object-orientation metamodel, when analyzing a domain, a modeler identifies specific classes, attributes, relationships, etc., elements that correspond to the features of the object-orientation metamodel. As we have noted before, models are not always generated running a $SCmm$, because by iteratively applying the reasoning we get an infinite stack of meta-metamodels.

A root metamodel must exist, from which everything comes. This is a recurring issue in the literature [Gonzalez-Perez and Henderson-Sellers, 2007, Seidewitz, 2003, Gitzel and Hildenbrand, 2005, Gitzel et al., 2007]. We claim that this root model is the Model of System, which aims to model everything using a set of things and a set of relationships between those things [Klir, 2013]. This model is an essential part of the human being, naturally 'implanted' in our minds, and that underlies other innate human mechanisms of deduction, inference and relation. It is the basis of any human analysis. Unfortunately, for reasons of space, we cannot analyze in detail the role of the model of system in our theory of modeling.

7.3 Discussion about submodels and metamodels

Other theories of modeling do not study submodels and metamodels as SMM does. The SMM definition of model neither uses nor needs the concept of inheritance. Instead, SMM introduces submodeling that it is more general. Furthermore, SMM defines metamodel very simply: models whose originals are also models. And whats more, the distinction between submodeling and metamodeling lies in their definitions, unlike, e.g., Kühne [Kühne, 2006], who suggests that transitivity is the main difference between these concepts. On the contrary, we propose that the principal distinguishing feature is its statements.

As to the issue of modeling levels, SMM is not intended to re-edit the four [Group, 2017], or more, levels approach. Instead, it focuses exclusively on the original-model relationship. Although SMM defines metamodel, it does not require that every model be an original of a metamodel, nor does it require the existence of a root model. In fact, many models have been created by means of an analytical process and utilized over time without relying on the concept of metamodel. However, SMM also allows us to study how a recursive root model could be [Gonzalez-Perez and Henderson-Sellers, 2007, Seidewitz, 2003, Gitzel and Hildenbrand, 2005]. Models are usually created by humans and therefore a model that deserves to be called root model must be an essential constituent of the way humans think of and see the world. For us this root model is the Model of System, as stated by Klir [Klir, 2013], which models everything using a set of things and a set of relationships between those things. Interest has been on the rise in solving these problems by rearchitecturing the four-level OMG model through various approaches known as multilevel modeling (MLM) [Atkinson and Kühne, 2001, Atkinson and Kühne, 2002, Balaban et al., 2018, Lara et al., 2014, Atkinson et al., 2015]. In our opinion, SMM improves understanding of these approaches. Although SMM does not require a layered model structure, different architectures can be defined as long as the basic rules of our proposal are followed. For instance, a model whose originals simultaneously include other models and the singleton models of the originals of the other models is valid in SMM, as long as a validity function and some constructors for said model can be defined. A new study would be necessary to fully analyze the various MLM proposals according to the SMM perspective. Thus, we propose this task for future research.

Our last consideration is that some modeling theories [Clark et al., 2015, Fuentes et al., 2003] only deal with notation issues, thus contributing to the confusion between the notions of modeling language and metamodel, and even between the notions of language and model. SMM helps to separate these notions, and aligns us with other references that do not put metamodels on the same level as 'models of modeling languages' [Group, 2014, Group, 2017, Bézivin, 2005, Harel and Rumpe, 2000, Bézivin and Gerbé, 2001]. But the fact that we have not needed the notion of language to discuss models and metamodels does not mean that there is no relationship between them.

7.4 An excerpt on languages and models

Although we are aware that it requires more detailed explanations, which we cannot include here due to space limitations, we could not finish this article without making a brief comment on the relationship between model and language terms, since this relation appears frequently in the literature [Kühne, 2006, Seidewitz, 2003, Rensink, 2005, Atkinson and Kuhne, 2003, Harel and Rumpe, 2000, Favre, 2005, Asikainen and Männistö, 2009].

We think of languages as communication tools that allow modelers to express their models in order to transmit them to others. Thanks to SMM we can define language more precisely from the point of view of modeling. For that purpose, SMM introduces two constructions: the linguistic metamodel and the semantic correspondence. Linguistic models originals are expressions that can be mapped to models thanks to a semantic correspondence. A certain linguistic model together with its semantic correspondence constitutes a Language.

Much of the misunderstanding between modeling languages and modeling itself in the literature is because modelers, who are usually human, need to use a language to formulate the statements of a model. Therefore, the expressions of that language are often confused with the models they express (e.g., it is not unusual to refer to class diagrams as models). But what actually happens is that there is only a semantic relationship between an expression (specification of the model statements) and its expressed-model by means of the languages semantic correspondence. Deepening on this correspondence between expressions and models, SMM can explain the reasons for this confusion. This analysis allows us to enunciate the notion of Modeling Language, which is a type of language that meets certain conditions: there exists a metamodel for all the models that take part in its semantic correspondence, and there exists a homomorphism between the features of that metamodel and the features of the modeling language linguistic model. That homomorphism is the key to understand the aforementioned confusion.

8 Conclusions

SMM contributes to the study of the foundations of modeling in an innovative way that both complements and at times clarifies some of the work conducted so far on other modeling approaches, especially those non-formal, such as OOMT. This study makes several contributions, all of them rooted in our proposal to incorporate concepts borrowed from scientific method into the definition of model: analysis, synthesis and analogy. Until now, these concepts had not been explicitly considered in the notion of model, despite being naturally present in modeling. The incorporation of these aspects into the definition of model in the form of analytical, synthetic and analogical constructors enriches the concept of model with a nuance of utility that we have found to be essential to explaining our view of several complex modeling mechanisms, which are extensively discussed in the literature. Clear examples of this can be found in our definitions of submodel and metamodel; our analysis of the types of originals and their physical or abstract nature; the extension of the classical typology of analytical/synthetic models (usually called descriptive/prescriptive) to include two new types: analogical and full models; or the clarification of the fact that the quality of being descriptive or prescriptive is not a role, as most authors believe. Thanks to the independence principle, our notion of model is free of references to the concepts of language or system. Though due to lack of space we havent dealt with the role of language in modeling, SMM allows to analyze the relationship between model and language, so as to explain the mechanisms by which the two closely interact. This has allowed us to tease out the confusion that appears in the literature regarding these terms, especially concerning modeling languages and metalanguages.

Acknowledgements

This work was partially supported by the Spanish Ministerio de Economía, Industria y Competitividad (project EDU2016-79838-P), and by the Agencia de Desarrollo Económico de La Rioja (project IRIS 4.0, 2017-I-IDD-00057).

References

[Andersen and Hepburn, 2016] Andersen, H. and Hepburn, B. (2016). Scientific method. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2016 edition.

[Asikainen and Männistö, 2009] Asikainen, T. and Männistö, T. (2009). Nivel: a metamodelling language with a formal semantics. *Software & Systems Modeling*, 8(4):521–549.

[Aßmann et al., 2006] Aßmann, U., Zschaler, S., and Wagner, G. (2006). Ontologies, meta-models, and the model-driven paradigm. In *Ontologies for software engineering and software technology*, pages 249–273. Springer.

[Atkinson et al., 2015] Atkinson, C., Gerbig, R., and Fritzsche, M. (2015). A multilevel approach to modeling language extension in the enterprise systems domain. *Information Systems*, 54:289–307.

[Atkinson and Kühne, 2001] Atkinson, C. and Kühne, T. (2001). The essence of multilevel metamodeling. In *International Conference on the Unified Modeling Language*, pages 19–33. Springer.

[Atkinson and Kühne, 2002] Atkinson, C. and Kühne, T. (2002). Rearchitecting the uml infrastructure. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 12(4):290–321.

[Atkinson and Kuhne, 2003] Atkinson, C. and Kuhne, T. (2003). Model-driven development: a metamodeling foundation. *IEEE software*, 20(5):36–41.

[Atkinson and Kühne, 2008] Atkinson, C. and Kühne, T. (2008). Reducing accidental complexity in domain models. *Software & Systems Modeling*, 7(3):345–359.

[Atkinson et al., 2000] Atkinson, C., Kuhne, T., and Henderson-Sellers, B. (2000). To meta or not to meta-that is the question. *Journal of Object Oriented Programming*, 13(8):32–35.

[Balaban et al., 2018] Balaban, M., Khitron, I., Kifer, M., and Maraee, A. (2018). Multilevel modeling: What's in a level? A position paper. In *Proceedings of MODELS 2018 Workshops*, pages 693–697.

[Barbero et al., 2007] Barbero, M., Jouault, F., Gray, J., and Bézivin, J. (2007). A practical approach to model extension. In *European Conference on Model Driven Architecture-Foundations and Applications*, pages 32–42. Springer.

[Bézivin, 2005] Bézivin, J. (2005). On the unification power of models. *Software & Systems Modeling*, 4(2):171–188.

[Bézivin and Gerbé, 2001] Bézivin, J. and Gerbé, O. (2001). Towards a precise definition of the omg/mda framework. In *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, pages 273–280. IEEE.

[Börstler et al., 2012] Börstler, J., Kuzniarz, L., Alphonce, C., Sanders, W. B., and Smialek, M. (2012). Teaching software modeling in computing curricula. In *Proceedings of the final reports on Innovation and technology in computer science education 2012 working groups*, pages 39–50. Haifa.

[Bottou, 2014] Bottou, L. (2014). From machine learning to machine reasoning. *Machine learning*, 94(2):133–149.

[Büttner and Gogolla, 2004] Büttner, F. and Gogolla, M. (2004). On generalization and overriding in uml 2.0. In *OCL and Model Driven Engineering, UML 2004 Conf. Workshop, O. Patrascoiu, Ed. Univ. Kent*, pages 1–15. Citeseer.

[Chen, 1976] Chen, P. P.-S. (1976). The entity-relationship modeltoward a unified view of data. *ACM transactions on database systems (TODS)*, 1(1):9–36.

[Clark et al., 2015] Clark, T., Sammut, P., and Willans, J. (2015). Applied metamodelling: a foundation for language driven development. *arXiv preprint arXiv:1505.00149*.

[Eriksson et al., 2013] Eriksson, O., Henderson-Sellers, B., and Ågerfalk, P. J. (2013). Ontological and linguistic metamodelling revisited: A language use approach. *Information and Software Technology*, 55(12):2099–2124.

[Favre, 2004] Favre, J.-M. (2004). Foundations of model (driven)(reverse) engineering–episode i: Story of the fidus papyrus and the solarus. In *Post-proceedings of Dagsthul Seminar on Model Driven Reverse Engineering*. Citeseer.

[Favre, 2005] Favre, J.-M. (2005). Foundations of meta-pyramids: Languages vs.

metamodels–episode ii: Story of thotus the baboon1. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[Flatscher, 2002] Flatscher, R. G. (2002). Metamodeling in eia/cdif—meta-metamodel and metamodels. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 12(4):322–342.

[Foundation, 2015] Foundation, T. E. (2015). Eclipse modeling framework. https://www.eclipse.org/modeling/emf/.

[French, 2002] French, R. M. (2002). The computational modeling of analogy-making. *Trends in cognitive Sciences*, 6(5):200–205.

[Frigg and Hartmann, 2020] Frigg, R. and Hartmann, S. (2020). Models in science. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2020 edition.

[Fuentes et al., 2003] Fuentes, J. M., Quintana, V., Llorens, J., Génova, G., and Prieto-Díaz, R. (2003). Errors in the uml metamodel? *ACM SIGSOFT Software Engineering Notes*, 28(6):3–3.

[Gašević et al., 2007] Gašević, D., Kaviani, N., and Hatala, M. (2007). On metamodeling in megamodels. In *International Conference on Model Driven Engineering Languages and Systems*, pages 91–105. Springer.

[Gentner, 1983] Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170.

[Gitzel and Hildenbrand, 2005] Gitzel, R. and Hildenbrand, T. (2005). A taxonomy of metamodel hierarchies. Technical report, University of Mannheim. https://madoc.bib.uni-mannheim.de/993/.

[Gitzel et al., 2007] Gitzel, R., Ott, I., and Schader, M. (2007). Ontological extension to the mof metamodel as a basis for code generation. *The Computer Journal*, 50(1):93–115.

[Goldstone and Son, 2005] Goldstone, R. L. and Son, J. Y. (2005). *Similarity*. Cambridge University Press.

[Gonzalez-Perez, 2018] Gonzalez-Perez, C. (2018). *Information Modelling for Archaeology and Anthropology*. Springer.

[Gonzalez-Perez and Henderson-Sellers, 2007] Gonzalez-Perez, C. and Henderson-Sellers, B. (2007). Modelling software development methodologies: A conceptual foundation. *Journal of Systems and Software*, 80(11):1778–1796.

[Gower, 2012] Gower, B. (2012). *Scientific method: A historical and philosophical introduction*. Routledge.

[Group, 2014] Group, O. M. (2014). Model driven architecture 2.0. Technical report, OMG. https://www.omg.org/mda/.

[Group, 2016] Group, O. M. (2016). Meta object facility (mof) 2.5.1. Technical report, OMG. https://www.omg.org/mof/.

[Group, 2017] Group, O. M. (2017). Unified model language infraestructure 2.5.1. Technical report, OMG. https://www.omg.org/uml.

[Gupta and Sykes, 2001] Gupta, P. and Sykes, J. A. (2001). The conceptual modeling process and the notion of a concept. In *Information modeling in the new millennium*, pages 53–68. Idea Group.

[Harel and Rumpe, 2000] Harel, D. and Rumpe, B. (2000). Modeling languages: Syntax, semantics and all that stuff part i: The basic stuff. MCS00-16.

[Henderson-Sellers, 2012] Henderson-Sellers, B. (2012). *On the mathematics of modelling, metamodelling, ontologies and modelling languages*. Springer Science & Business Media.

[Hesse, 2000] Hesse, M. (2000). Models and analogies. *A Companion to the Philosophy of Science: Malden, MA, Blackwell Publication*, pages 299–307.

[Hesse, 2006] Hesse, W. (2006). More matters on (meta-) modelling: remarks on thomas kühnes matters. *Software & Systems Modeling*, 5(4):387–394.

[Hodges et al., 1997] Hodges, W. et al. (1997). *A shorter model theory*. Cambridge university press.

[Jouault et al., 2009] Jouault, F., Bézivin, J., and Barbero, M. (2009). Towards an advanced model-driven engineering toolbox. *Innovations in Systems and Software Engineering*, 5(1):5–12.

[Kästner et al., 2018] Kästner, A., Gogolla, M., and Selic, B. (2018). From (imperfect) object diagrams to (imperfect) class diagrams: New ideas and vision paper. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 13–22.

[Kelly and Tolvanen, 2008] Kelly, S. and Tolvanen, J.-P. (2008). *Domain-specific modeling: enabling full code generation*. John Wiley & Sons.

[Klir, 2013] Klir, G. J. (2013). *Facets of systems science*, volume 7. Springer Science & Business Media.

[Kühne, 2006] Kühne, T. (2006). Matters of (meta-) modeling. *Software & Systems Modeling*, 5(4):369–385.

[Kühne, 2009] Kühne, T. (2009). Contrasting classification with generalisation. In *Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling-Volume 96*, pages 71–78.

[Kühne, 2010] Kühne, T. (2010). An observer-based notion of model inheritance. In *International Conference on Model Driven Engineering Languages and Systems*, pages 31–45. Springer.

[Kurtev, 2007] Kurtev, I. (2007). Metamodels: Definitions of structures or ontological commitments. In *Workshop on TOWERS of models. Collocated with TOOLS Europe*, volume 2007.

[Lake et al., 2017] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

[Lara et al., 2014] Lara, J. D., Guerra, E., and Cuadrado, J. S. (2014). When and how to use multilevel modelling. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(2):1–46.

[Ludewig, 2003] Ludewig, J. (2003). Models in software engineering–an introduction. *Software and Systems Modeling*, 2(1):5–14.

[Magnani et al., 2017] Magnani, L., Bertolotti, T., and Heeffer, A., editors (2017). *Springer handbook of model-based science*. Springer.

[Mellor et al., 2004] Mellor, S. J., Scott, K., Uhl, A., and Weise, D. (2004). *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional.

[Merunka, 2003] Merunka, V. (2003). Critical assessment of the role of uml for information system development. *Systems Integration*, pages 445–452.

[Muller et al., 2012] Muller, P.-A., Fondement, F., Baudry, B., and Combemale, B. (2012). Modeling modeling modeling. *Software & Systems Modeling*, 11(3):347–359.

[Ober and Prinz, 2006] Ober, I. and Prinz, A. (2006). What do we need metamodels for. In *4th Nordic Workshop on UML and Software Modelling*, pages 8–28.

[Ogden and Richards, 1923] Ogden, C. K. and Richards, I. A. (1923). *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*, volume 29. K. Paul, Trench, Trubner & Company, Limited.

[Orilia and Swoyer, 2020] Orilia, F. and Swoyer, C. (2020). Properties. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2020 edition.

[Overbeek, 2006] Overbeek, J. (2006). Meta object facility (mof): investigation of the state of the art. Master's thesis, University of Twente.

[Rensink, 2005] Rensink, A. (2005). Subjects, models, languages, transformations. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[Rodriguez-Priego et al., 2010] Rodriguez-Priego, E., García-Izquierdo, F. J., and Rubio, Á. L. (2010). Modeling issues: a survival guide for a non-expert modeler. In *International Conference on Model Driven Engineering Languages and Systems*, pages 361–375. Springer.

[Rodriguez-Priego et al., 2013] Rodriguez-Priego, E., García-Izquierdo, F. J., and Rubio, Á. L. (2013). References-enriched concept map: a tool for collecting and comparing disparate definitions appearing in multiple references. *Journal of information science*, 39(6):789–804.

[Rodriguez-Priego et al., 2019] Rodriguez-Priego, E., García-Izquierdo, F. J., and Rubio, Á. L. (2019). Extended analysis of related work on modeling theory: Rcms of model, metamodel and modeling language. Technical report, The institution that published. Available online: https://doi.org/10.5281/zenodo.1321248.

[Rothenberg, 1991] Rothenberg, J. (1991). Prototyping as modeling: what is being modeled? In *Dynamic modelling of information systems*, pages 335–359. Elsevier.

[Sánchez et al., 2009] Sánchez, D. M., Cavero, J. M., and Marcos, E. (2009). The concepts of model in information systems engineering: a proposal for an ontology of models. *The Knowledge Engineering Review*, 24(1):5–21.

[Seidewitz, 2003] Seidewitz, E. (2003). What models mean. *IEEE software*, 20(5):26–32.

[Shan and Zhu, 2008] Shan, L. and Zhu, H. (2008). A formal descriptive semantics of uml. In *International Conference on Formal Engineering Methods*, pages 375–396. Springer.

[Stachowiak, 1973] Stachowiak, H. (1973). *Allgemeine modelltheorie*. Springer.

[Sykes, 2003] Sykes, J. (2003). Negotiating early reuse of components-a model-based analysis. In *UML and the unified process*, pages 66–79. IGI Global.

[Terrasse et al., 2006] Terrasse, M.-N., Savonnet, M., Leclercq, E., Grison, T., and Becker, G. (2006). Do we need metamodels and ontologies for engineering platforms? In *Proceedings of the 2006 international workshop on Global integrated model management*, pages 21–28.

[Vellido et al., 2012] Vellido, A., Martín-Guerrero, J. D., and Lisboa, P. J. (2012). Making machine learning models interpretable. In *20th European Symposium on Artificial Neural Networks*, volume 12, pages 163–172. Citeseer.

[Williams, 2013] Williams, H. P. (2013). *Model building in mathematical programming*. John Wiley & Sons.