SN Computer Science (2024) 5:861
https://doi.org/10.1007/s42979-024-03163-y

ORIGINAL RESEARCH

Understanding the Variety of Domain Models: Views, Programs, Animations, and Other Models

Henderik A. Proper[1] · **Giancarlo Guizzardi**[2]

Received: 1 July 2023 / Accepted: 22 July 2024 / Published online: 9 September 2024

Abstract

Humanity has long since used models, in different shapes and forms, to understand, redesign, communicate about, and shape, the world around us; including many different social, economic, biological, chemical, physical, and digital aspects. This has resulted in a wide range of *modeling practices*. When the models as used in such *modeling practices* have a key role to play in the activities in which these practices are 'embedded', the need emerges to consider the effectiveness and efficiency of such processes, and speak about *modeling capabilities*. In the latter situation, it also becomes relevant to develop a thorough understanding of the artifacts involved in modeling practices/capabilities. One context in which models play (an increasingly) important role is *model-driven systems development*, including software engineering, information systems engineering, business process engineering, enterprise engineering, and enterprise architecture management. In such a context, we come across a rich variety of modeling related artifacts, such as views, diagrams, programs, animations, specifications, etc. In this paper, which is actually part of an ongoing 'journey' in which we aim to gain deeper insights into the foundations of modeling, we take a fundamental look at the variety of modeling related artifacts as used in the context of model-driven (systems) development, while also presenting an associated *framework for understanding*, synthesizing the insights we obtained during the 'journey' so-far. In doing so, we will also argue that the aforementioned artifacts are actually specific kinds of models, albeit for fundamentally different purposes. The provided *framework for understanding* involves definitions of *domain model*, the *Return on Modeling Effort* (RoME), the *conceptual fidelity* of domain models, as well as *views* as a mechanism to manage the complexity of domain models.

Keywords Domain modeling · Return on modeling effort · Conceptual fidelity · Views

Introduction

Whenever we are confronted with complex phenomena, such as the processes we observe in nature, the construction of buildings, the design of information systems, etc., we tend to 'work with' an *abstraction* (in our mind) of the actual phenomenon; zooming in on those 'properties' of the phenomenon that matter to us, while filtering out all the properties that are not germane to the goals at hand. When we externalize this *abstraction* in terms of some artifact, then to us, as an individual, this artifact is a model of the observed phenomenon. For such a model to be 'recognized' as a model, the artifact needs to be a human understandable representation of said abstraction.

The latter view on models is rooted on (among others) the definitions of *model* as provided by e.g. Apostel [1] and Stachowiak [73]. Here, it is important to already acknowledge the fact that the notion of model as put forward by these scholars does not stipulate any a priori constraints on e.g. the level of completeness, precision, or even (mathematical) formality of the model. Depending on the situation, and the purpose of a model, different requirements can be put on a model [6, 39], including requirements regarding, e.g., its completeness, precision, or formality.

In line with the general notion of *domain* as e.g. provided in the Webster dictionary [79] "a sphere [...] of knowledge, influence, or activity", we prefer to refer to these models as *domain models*, where the term *domain* refers to the

✉ Henderik A. Proper
henderik.proper@tuwien.ac.at

✉ Giancarlo Guizzardi
g.guizzardi@utwente.nl

[1] TU Wien, Vienna, Austria

[2] University of Twente, Enschede, The Netherlands

represented 'something'[1]. The requirement for *domain models* to be human understandable, connecting the cognition of modelers, and model users, to a referent (i.c. the domain being modeled), also clearly distinguishes these models from, e.g., machine-learning models. For practical reasons, in the remainder of this paper, we will use the term *model* as an abbreviation for *domain model*.

More generally, one can observe how humanity has long since used (domain) models to understand, redesign, communicate about, and shape, the world around us, including many different social, economic, biological, chemical, physical, and digital aspects. These models may take different shapes and forms, such as sketches, precise drawings, textual specifications, or tangible forms mimicking key physical properties of some original. This widespread, and natural [6, 81], use of models has resulted in many different *modeling practices*.

When the models as created and/or used in such *modeling practices* have a key role to play in the activities in which these *modeling practices* are "embedded", a natural need emerges to consider the effectiveness and efficiency of such processes, and actually speak about *modeling capabilities*,[2]. In the latter situation, it becomes relevant to develop a thorough understanding of the artifacts involved in the modeling practices/capabilities.

One context in which models play (an increasingly) important role is *model-driven systems development*, including software engineering, information systems engineering, business process engineering, enterprise engineering, and enterprise architecture management. In this paper, we will focus on the variety of models as used in the context of *model-driven development*.

It should be noted that model-driven development approaches tend to limit the notion of *model* to being an artifact with (at least) an explicit structure, and often also require these models to have a formal semantics in mathematical terms. However, as mentioned before, the notion of model as put forward by e.g. Apostel [1] and Stachowiak [73] clearly suggest to consider the notion of model from a broader frame of mind; a perspective we wholeheartedly embrace. While models with a formal semantics are, indeed, important from an engineering point of view, it is important to also acknowledge that, for instance, drawings made on the 'back of a napkin', sketches of system architectures, use cases, etc, are essentially all used as models in their associated modeling practices.

In the context of model-driven development, we also come across a rich variety of modeling related artifacts, such as views, diagrams, programs, animations, specifications, etc. In this paper, which is actually part of an ongoing 'journey' [57–60] in which we aim to gain, and articulate, deeper insights into the foundations of modeling, we aim to take a deeper look at these artifacts. More specifically, in this paper, we will suggest a *framework for understanding* for the variety of modeling related artifacts. In doing so, we will argue that the aforementioned artifacts should actually be seen as specific kinds of models, albeit for fundamentally different purposes. A first version of said framework was sketched out in [60], which was inspired by the keynote of one of the authors at MODELSWARD 2023 [55]. The resulting framework, as presented in this paper, comprises:

1. a philosophically and linguistically founded definitions of *domain model*,
2. the identification of a model's *Return on Modeling Effort* (RoME),
3. the definition of the *conceptual fidelity* of domain models, and
4. *views* as a mechanism to manage the complexity of domain models,

and synthesizes much of our earlier work on the foundations of modeling [5, 20, 23, 35, 57–59, 61].

As mentioned above, we consider it to be of increasing importance to develop a thorough understanding of the artifacts involved in modeling practices/capabilities, and model-driven (system) engineering in particular. It is this conviction that also inspired us in developing the presented framework; which we certainly expect to evolve further as part of our ongoing 'journey' into the foundations of domain modeling.

In line with this, the remainder of this paper is structured as follows. In the section "Domain Models" we start by zooming in on the notion of (domain) model itself, resulting in a philosophically grounded definition of domain model. The section "Domain Models as Complex Speech Acts" takes a more fundamental perspective on a model as being a *complex speech act*. We then continue, in the section "Return on Modeling Effort", by addressing the importance of knowing a model's purpose and its potential *Return on Modeling Effort* in particular. In the section "Conceptual Fidelity", we turn our attention to the notion of conceptual model, which is a class of models that has initially grown

[1] Within some engineering disciplines, including software engineering in particular, the term *domain modeling* is used in a more restricted sense, namely, to refer to a (domain) model capturing the general 'problem area'. As such, using the term *domain model(ing)* as an abbreviation of *problem domain model(ling)*; see e.g. [50]. In the area of *domain engineering*, especially when explicitly connected to the notion of domain ontology as in [15], what is referred by the term domain model is an important exemplar of what we mean by the expression here. As such, we use the word domain here in the more general sense as put forward in the dictionary [79].

[2] Ontologically speaking, capabilities (or capacities) are *gradable dispositions* i.e., properties that are manifested in certain situations via the occurrences of events of a certain kind [3].

to play an important role in the field of information systems engineering, but has a much wider role to play. In that section, we will also introduce the notion of *conceptual fidelity*. This allow us to see the property of being *conceptual* itself as a gradable property of models. Building on this grounding, section "Complexity Management and Views" discusses the notion of *views* as a complexity management mechanism that supports making complex models *cognitively tractable*. Finally, before concluding, the section "Diagrams, Programs, Animations, and other Models" positions artifacts such as diagrams, programs, and animations, as being specific kinds of models, covering different purposes for different audiences.

Domain Models

Combining foundational work by Apostel [1], and Stachowiak [73], more recent work on the same topic by different authors [32, 66, 67, 74], as well as our own work [5, 6, 20, 23, 35, 57–59, 61], we currently understand a *domain model* to be:

> A *social artifact* that is *understood*, and *acknowledged*, by a *collective human agent* to *represent* an *abstraction* of some *domain* for a particular *cognitive purpose*.

In line with [79], with *domain* we refer to 'anything' that one can speak and/or reflect about; i.e. the domain of interest. As such, *domain* simply refers to 'that what is being modeled'. Below we will return in more detail regarding to the notion of the domain that is being modeled. Furthermore, the domain could be something that already exists in the 'real world', something that is desired to exist in the future, something imagined, or even something that is brought about by the existence of the model itself. We will return to this point in the section "Domain Models as Complex Speech Acts" when discussing models as *complex speech acts*. In the context of system development at large, examples of more specific classes of domain models include enterprise (architecture) models, business process models, ontologies, organizational models, information models, software models, problem (domain) models[1], etc. We consider all of these as valued members of the larger family of *domain models*.

A model must always be created for some *cognitive purpose*,[3]; i.e. to express, specify, learn about, or experience, knowledge regarding the modeled domain. This also implies that, in line with the *cognitive purpose* of the model, some, if not most, 'details' of the domain are consciously filtered out.

As we regard a model to be an *artifact*, this also implies that it is something that exists outside of our minds; i.e. as 'represented abstractions'. More specifically, a model is seen as a *social artifact*. in the sense that its role as a model should be recognizable by a *collective human agent*[4]. This, once more, eludes to the fact that models are, as we will discuss in the section "Domain Models as Complex Speech Acts", essentially *complex speech acts*.

The *understood, and acknowledged, by a collective human agent* phrase clearly differentiates *domain models* from, e.g., machine-learning models. Although a domain model can certainly involve complex mathematical formalisms, or computer readable specifications.

In the context of (model-driven) system development, models typically take the *form* of some 'boxes-and-lines' diagram. More generally, however, domain models can, depending on the *purpose* at hand, take other forms as well, including (controlled) natural language texts, mathematical specifications, games, sketches, animations, simulations, and physical objects. We will elaborate on the latter in the section "Diagrams, Programs, Animations, and other Models".

Domain Models as Complex Speech Acts

Requiring a model to be an artifact that needs to be understood by human agents, immediately puts models in the realm of language. Therefore, an important theoretical foundation of domain models is the semiotic triangle by Ogden and Richards [47], as depicted in Fig. 1. The semiotic triangle is traditionally used as a base to theorize about meaning in the context of language [13, 46, 69, 76], but has also been used widely to theorize about the meaning of domain models [33, 38, 39, 42].

The tenet of the semiotic triangle is that when we use *symbols*, including models, to speak about 'something', i.e. the *referent*, then these symbols represent, i.e. *symbolize*, our *thought or reference* about that something. The *thought or reference* is the meaning we have assigned to the *symbols*.

In the context of modeling, the notion of 'thought or reference' is generally replaced by the notion of *concept*. The *referent* can be anything, in an existing world, or in a desired/imagined world. It can involve physical phenomena (e.g., tree, car, bike, atom, planet, picture, etc), mental

[3] In earlier work, we did not include the explicit focus on *cognitive purpose* but rather spoke about some *purpose* in general. In retrospect, we think this was an omission. Adding *cognitive* clarifies the role of models as a way to express, specify, learn about, or experience, knowledge regarding the modeled domain. We would like to thank Jan Schoonderbeek for making us aware of this omission.

[4] The pre-noun *collective* does suggest that it would require the involvement of multiple people. We do, indeed, acknowledge the use of domain models by an individual person as well, but prefer to treat this as a special case concerning a 'self-shared' model.

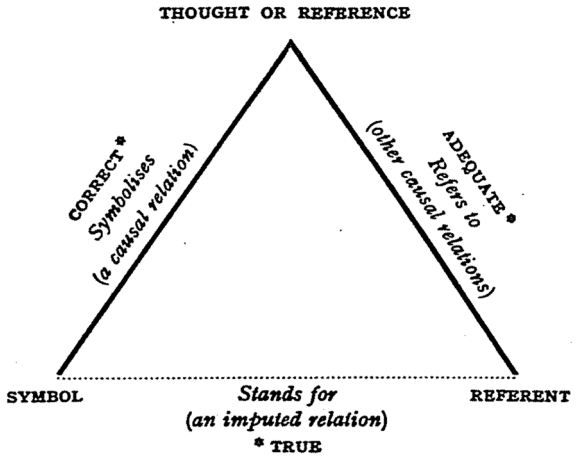

Fig. 1 Ogden and Richard's semiotic triangle [47]

phenomena (e.g., thoughts, feelings, etc), as well as social phenomena (e.g., marriage, mortgage, trust, value, etc).

A domain to be modeled generally involves a complex of (related) referents, which, following the semiotic triangle (Fig. 1), results in a complex of corresponding concepts/thoughts; the (domain) *abstractions* is in the mind of human agents. Depending on its purpose, such an abstraction can have differing levels of generality, mirroring the generality of the domain to be modeled. For instance, the abstraction can pertain to the *legal domain*, the world of *business processes*, or the world of *marital relationships* as a whole, but may also pertain to a specific *court case*, a specific *business process* or a specific *marriage*. For the world of *marital relationships* as a whole, the abstraction is likely to involve concepts that reflect (classes of) *people*, *marriages*, and *offspring*. In line with [20], we refer to such an abstraction of a general domain as a *conceptualization*. Such a *conceptualization* identifies the fundamental concepts in terms of which one creates their abstraction of the world. At the level of a specific marriage, the abstraction may involve *person John*, *person Mary*, *person Sarah*, the *marriage between John and Mary*, and *Sarah being born in the marriage between John and Mary*. In the latter case, we also see how the general domain *conceptualization* essentially defines a 'lens' by which we observe the more specific domain to be modeled.

In [31], while reflecting on the value of models and the underlying purpose for which models are ultimately created ('uttered') by (collective) human agents, we argued that *domain models* should essentially be regarded as *complex speech acts* [68]. Doing so, enabled us to apply the notion of *direction of fit* [70] from the areas of *philosophy of mind* and *philosophy of language* in defining a basic taxonomy of overall purpose for which models are created.

The notion of *direction of fit* [70] is meant to connect the *propositional content* of intentional aspects (i.e., mental states or speech acts) to the external state of affairs of which they are about. There are basically three possible directions of fit:

1. *World-to-Mind (or World-to-Word)* – the propositional content of a mental state (i.e., a desire or intention) or of a speech act is made true by making the world such that it conforms with that propositional content. In terms of the semiotic triangle, the referent, i.e. the part of the world that the thought or symbol refers to, needs to be made conformant to the thought or symbol. For example, if John intends to go to Barcelona next summer or if Mary plans to finish her paper by tomorrow, they have to intervene in the world to make the propositional content of their intention or speech act true.
2. *Mind-to-World (or Word-to-World)* – the propositional content of a mental state (i.e., a belief) or the speech act is made true if there is something in the world that makes it true. In terms of the semiotic triangle, the thought or symbol must be articulated as such to conform to their referent. For example, if John believes Rome is the capital of Italy or if Mary states "I am married to John", these things are true if there is something in the world that make them true (in this case, a particular city and country with a particular legal relation between the two, and a marriage).
3. *World-to-Word-to-World (or double direction of fit)* – by uttering something, an individual can bring about some change the world, which then becomes the *truthmaker* [21] of sentences with that corresponding propositional content. For example, if a judge utters "I hereby declare you (John) and you (Mary) husband and wife" this utterance creates a marriage binding John and Mary, which then becomes the truthmaker of the proposition "John and Mary are married". In terms of the semiotic triangle, we have the situation in which an actor expresses a symbol *s* and, by doing so, brings about in the world a referent *r* that is, hence, conformant to the semantic content of *s*. This then makes *s* a truthful description of *r* while then, as a consequence, other actors take/accept *s* as a (future) truth with regards to *r*.

If we take models to be complex speech acts of this form, we can come up with the analogous categories of (a) *World-to-Model*; (b) *Model-to-World*; (c) *World-to-Model-to-World* directions of fit. Models of type (a) and (c) are models *for changing the world*. In the case of models of type (c), the model itself brings about change in the world by its existence and recognition in a given community. We call these latter models *Creative Models*. Examples include a diagram in a patent file, which helps to create intellectual property rights, or a model included in a Will dividing a piece of real state among someone's heirs (in

both cases, by expressing the semantic content of those rights that are henceforth created).

In the case of models of type (a), the model is an instrument through which one can bring about changes in the world. These include *designs* (e.g., a blue print for a house) that will be implemented, and *plans* (e.g., a BPM model of a process TO-BE). We call these models *Prescriptive Models*. These models can be used by individuals or collective of individuals (i.e., coordination models).

Models of type (b) are called *Descriptive Models*. These are models that represent a relation between abstractions in the mind of human agents and some existing external reality (the referents of the model). Notice that these two relata correspond to *Thought or Reference* and *Referent* in Ogden and Richards' semiotic triangle [47], respectively.

Next to prescriptive and descriptive models, one can also identify predictive models which, by their very nature, have been created to predict the behavior of different aspects of the domain that is being modeled. This predictive 'capability' can be used for both descriptive and prescriptive models. One can make a predictive-descriptive model of an existing situation (*Model-to-World*) with the aim to predict the behavior of the modeled domain. In this case, we expect the domain to, without make changes to the domain, behave as predicted by this prescriptive-descriptive model. Conversely, one can make a predictive-prescriptive (*World-to-Model*) model to predict the behavior of a domain, when the domain is changed/adapted in line with the domain model.

Return on Modeling Effort

The creation, administration, and use, of domain models, as well as the development of modeling capabilities, require investments in terms of time, money, cognitive effort, etc. We contend that such investments should be met by a (potential) return. In other words, the resulting models and/or the processes involved in their creation, administration, and use, should add value that make these investments worth while. This has resulted, analogously to the notion of Return on Investment, in the notion of Return on Modeling Effort (RoME). We first coined this notion in a publication[5] in [40], while a more elaborate discussion of the concept is provided in [31, 59] as part of our joint endeavor to better understand the foundations of (domain) modeling and modeling practices.

[5] The notion of RoME actually made its first informal appearance in [52] as a leading principle in the research group of one of the authors, while a first informal elaboration of the concept was published as a blog post [53].

Examples of situations where explicit trade-offs regarding the (expected) RoME would be beneficial include the level of breath (scope-wise) of the model, the level of detail of a model, and the level of formality of a model.

When better underpinning the notion of RoME, it is important to consider both models and modeling, from a value-oriented perspective. In doing so, we take the *Value Proposition Ontology* as defined in [51] as a base. In particular, we rely on the notions of:

- *value object* – an object to which value is ascribed.
- *value experience* – an (envisaged, actual, or past) experience (an event) to which value is ascribed.
- *value bearer* – the generalization of *value object* and *value experience*,[6].
- *value beholder* – a role played by the actor who ascribes value.
- *value beneficiary* – a role played by the actor whose goals are (possibly, partially) satisfied by participating in value experiences.
- *value ascription* – a collection of *value ascription components*, each of which concerns a personal judgment by a *value beholder* of the *benefits* for, or *sacrifices* by, a *value beneficiary*.

For a given model, the *effort* part of RoME involves the *sacrifices* by the *value beneficiary*, while the *return* part of RoME involves the *benefits* accrued by them[7]. Based on this, one can determine the RoME ratio for a specific *value beneficiary*, or do so at an overall level by combining the *sacrifices* and *sacrifices* across all *value beneficiaries*. In line with this, it might have been better to speak about *Benefits from Modeling Sacrifices* instead of RoME as this would

[6] In [51], even when one considers *value objects* one are ultimately interested in the experiences afforded by (the capabilities and qualities of) these objects. In other words, the focus of *value ascription* is ultimately always a *value experience*.

[7] In [51], the *value beholder* ascribes value to *value bearers* (objects or experiences) considering benefits and sacrifices w.r.t. the goals of the *value beneficiary* – sacrifices are in a sense negative influences on goals of maintenance of resources (e.g., time, money, energy). The participation of a *value beneficiary* in a *value experiences* (enacted by *value objects*) always incur in sacrifices to them. In the cases in which the *value beholder* is not the same as the (main) *value beneficiary*, the former can also be the bearer of sacrifices. In this case, since the *value beholder* is willing to invest resources in enabling the *value experience* of the *value beneficiary*, we assume that the *value beholder* is also a (secondary) *value beneficiary* of the *value experience*. For example, when parents decides to invest resources in a high-quality education for their children, we have that: (i) these children have to invest resources (bear sacrifices) for the co-creation of the *value experience*; (ii) the goals to be satisfied are, besides those of the children – the main *value beneficiary*, also those of the parents. Here we simplify this analysis by simply taking benefits and sacrifices to be borne by the value beneficiary.

have more closely followed the terminology from the value proposition ontology. However, for RoME we prefer to stick to the analogy to RoI (Return on Investment).

In actual RoME related trade-offs, it will be necessary to distinguish between the ex-ante *expected* value ascriptions and the ex-post *realized* value ascriptions.

In the case of models and modeling, there seem to be three potential *value bearers*:

1. *Value in creation* – The *process* of (co-)creating a domain model. Such a process may, e.g., result in the added value that those who are involved in the modeling process[8] develop a deeper and/or more consistent (joint) understanding of the modeled domain, and also have the chance of building shared terminology based on that understanding. In this case, the *value beneficiaries* pertain to the actor(s) who are directly involved in the (co-) creation process, and/or those who stand to benefit from an increased (joint) understanding of the latter actor(s). The *value bearer* in this case is a *value experience* of (co-)creating the model.
2. *Value in use* – The operational *usage* of the model, in line with its purpose.[9] This may, e.g., involve the use of the model to support decision making, give prescriptive/descriptive guidance towards development processes and/or operational processes, etc. In this case, the *value beneficiary* is typically the user of the model – who benefits from the support of the model in decision-making, design, planning, coordinating, etc. However, they can also be an actor who has a more overall role/interest positively affected by the use of the model (such as the transfer of design knowledge from requirements engineering, via design, to implementation). The *value bearer* is the *value experience* afforded by the use of the model.
3. *Value in transaction* – The ownership of the model itself. This pertains to e.g. reference models, design models, etc, that capturing knowledge that can potentially be re-applied in different situations. In this case, the *value bearer* is a *value object*; i.e. the model.

At a more fundamental level, we would argue that ultimately *value in creation* and *value in use* are the root/direct value bearers of models. The *value in transaction* is derived from the potential of a model's future *value in use*. The combination of *value in creation* and *value in use* is what we refer to as the *Value in Action* (ViA) of models. This view allows us to think of the enactment of modeling practices as *value experiences*, and hence see RoME as the ratio between the benefits and sacrifices involved in these *value experiences*.

In the work we reported in [31], we provided a goal structure in terms of a taxonomy of modeling related goals. This taxonomy is based on the *direction of fit* [70] as also discussed above in the section "Domain Models as Complex Speech Acts". The resulting taxonomy of modeling goals distinguishes between models with a prescriptive purpose (interven*ing*, plann*ing*, coordinat*ing*), a creative purpose (bring*ing* about changes in reality), and a descriptive purpose (understand*ing*, problem-solv*ing*, communicat*ing*, and document*ing*), each time involving models that receive their *Value in Action*.

For example, we refer to descriptive models (*model-to-world* direction of fit) that create value by enhancing domain understanding, conceptual clarification, meaning negotiation via the *creation* of models that truthfully describe (a proper abstraction/conceptualization of) that domain. In addition, cases of *value in use* of such models (i.e. *Value in Action*) include: (i) *communication* (i.e., descriptive models that bring value to the value beneficiary by informing truthful information about the domain and via the experience of model interpretation); (ii) *problem-solving* (i.e., descriptive models that bring value to the model user and other indirect beneficiaries via the experience of model manipulation); (iii) *intervening* (i.e., prescriptive models with a *world-to-model* direction of fit that bring value by supporting an experience of intervening in reality to make it satisfy the propositional content of that model).

Finally, the complexity management tools (views) as discussed in the section "Complexity Management and Views", as well as the other types of modeling artifacts we discuss in the section "Diagrams, Programs, Animations, and other Models", result in different types of artifacts that afford different modeling experiences leading to different *Value in Action*(s).

Conceptual Fidelity

In the context of information systems engineering, an important role is played by *conceptual models*, which we see as a specific class of domain models. According to the traditional information systems engineering view [37], a conceptual model captures the essential structures of some *universe of discourse*. In this context, conceptual models are used to express the concepts, and their (allowed) relations and constraints, of the *universe of discourse* (while avoiding the inclusion of design/implementation/storage details).

The field of information systems engineering, indeed, provides a fruitful application area for conceptual modeling. At the same time, however, we suggest to avoid a 'framing'

[8] Which could be a group of actors, but can also be a single actor expressing their thoughts about an existing/future domain.

[9] One could indeed also gain value from a model by (ab)using it beyond its intended purpose.

of what a conceptual model is to this application area only. As such, we suggest a more generalized understanding of the notion of *conceptual model*. More specifically, based on [20, 54, 57], in our current understanding a *conceptual model* is:

A domain model, where:

1. the purpose of the model is dominated by the ambition to remain as-true-as-possible to the *conceptualization* of the domain by the collective agent, while
2. there is an explicit *mapping* from the elements in the model to the latter *conceptualization*.

As discussed in section "Domain Models as Complex Speech Acts", the *conceptualization* of a general domain (e.g. the *legal domain*, the world of *business processes*, or the world of *marital relationships* as a whole) identifies the fundamental concepts in terms of which the *collective agent* create(s) their abstraction(s) of the world. This mapping characterizes the *ontological commitment* of the model (and the *collective agent*) as well as its real-world semantics [23, 28].

Returning to the above point regarding the need to consider the role of conceptual models beyond the field of information systems engineering, the ambition to remain *as-true-as-possible to the conceptualization of the domain by the collective agent* is not only of value in the context of information systems engineering, but other contexts as well. For instance, [20] already stated that the history of conceptual modeling can be traced back to at least the 60 s [62]. Furthermore, ontology engineering [23] also involves the construction of conceptual models representing an ontology.

At a more general level, we also observe that in many different endeavors in which we (as humans) aim to understand the workings of some domain and/or aim to express, or study, design alternatives, we actually do so in terms of (purpose and situation specific) domain models. This includes many examples across science and engineering at large. We also argue that in these cases, a deepening of our understanding of the essential mechanisms leads to a natural drive to create domain models that remain as-true-as-possible to the original domain (and our conceptualization thereof); i.e. conceptual models.

Since a conceptual model is meant to be used by human agents in tasks such as domain understanding and learning, communication (including explanation [28]), problem-solving, and meaning negotiation [20], another fundamental quality attribute of a conceptual model is its *pragmatic efficiency*, i.e., how easy it is for those human agents to perform these aforementioned tasks with these models [25, 29].

As a result, a conceptual model provides an explicit – human understandable, ideally, pragmatically efficient – representation of a theory about the entities and their ties that are assumed to exist in a given *domain of interest* (according to a given ontological commitment); as such explicitly capturing descriptive and/or prescriptive selected aspects of the modeled domain. Conceptual models, therefore, enable us to explicitly clarify the things we talk and reason about; at a chosen level of abstraction and from a desired perspective.

Identifying conceptual models as a specific class of domain models, does raise the question regarding the role of 'other' domain models that are 'not conceptual'. In [57] it is suggested to, next to conceptual models, also identify *computational-design models*. These latter models may involve 'conceptual compromises', with regard to the ambition *to remain as-true-as-possible to the original domain conceptualization*, to cater for highly desirable computational considerations to, e.g., support simulation, animation, or even execution of the model. In [54] it is suggested to generalize this towards *utility-design models*, to cater for the fact that 'conceptual compromises' may not only be introduced for *computational* purposes, but also for e.g. *experiential* purposes, such as the ability to touch, feel, or even 'enact' a model.

An interesting analogy, which certainly needs further investigation, is the notion of *surrogate modeling* in the context of simulation [63] of real-world systems. The level at which a simulation model reflects all (relevant) properties of a (planned/existing) real-world system is referred to as the *fidelity* of the simulation model: "Fidelity in the modeling context refers to the degree of the realism of a simulation model" [63]. Likewise, one can speak of being as-true-as-possible-to a given domain as a sort of *conceptual fidelity*. Conceptual fidelity represents the level of homomorphism between a given representation and the underlying domain conceptualization it commits to. An alternative name for this concept, that is also often used, is *domain appropriateness*.

In the ideal case, this representation artifact is not only isomorphic to the structure of that conceptualization (i.e., it represents in a univocal and non-redundant way all its constituting concepts and only them)[10] but it also only allows for

[10] We emphasize here that a model is always the result of an intentional act, i.e., it is the result of a deliberate intention of connecting representation to a conceptualized reality. As discussed in [20], in line with [19], having an isomorphism between a certain symbolic structure and a represented entity is not sufficient for that structure to be a model of that entity: an explicit intentional act is also necessary for that to be the case. Moreover, although we have (in an ideal case) an isomorphic mapping between the representation (the artifact) and the conceptualization, the relation of *being a model of* is non-reflexive, asymmetric and non-transitive. This contrasts with (only) *being an isomorphism*, which is an equivalence relation. These meta-properties of the former relation are explained by the lack of an intentional act making the model a model of itself, making the domain conceptualization a model of the representation structure, etc. Finally, in conformity with the view defended in [10], we consider that conceptual

interpretations that represent state of affairs deemed acceptable by that conceptualization [25, 26]. As, in the case of simulation models of real-world systems, the involved high fidelity models may be too computationally intensive to simulate as a whole, one uses so-called *surrogate models* [63] that are computationally more efficient, while approximating the high fidelity model good enough to meet the (optimization) purpose at hand.

In an information systems engineering context, it is interesting to note that the ambition for a conceptual model to *remain as-true-as-possible to the conceptualization of the domain by the collective agent*, has a direct correspondence to the *conceptualization principle* as put forward in the well known ISO report on the design of information systems [37].

It is important to note that we do not argue that non-conceptual models are a bad thing; far from it. In fact, utility-design models are necessary for mapping conceptual models to possibly multiple enactment solutions (e.g., multiple computational implementations). However, it needs to be clear what the 'conceptual deviations' are of a non-conceptual model in relation to the conceptual model of the same domain, and what the benefit are of these deviations in terms of e.g. computational efficiency or experiential properties. As such, it might quite well be the case that one conceptual model has different associated non-conceptual models catering for different needs [24]. Conversely, we would expect that each non-conceptual model has been based on (at least one) corresponding conceptual model.

Returning briefly to the notion or RoME, we postulate that the RoME of a conceptual model is necessarily higher than the sum of the RoME of each of the non-conceptual models that have been derived from it. If only because – besides the benefits accrued via the actions involving these non-conceptual models – this *common* conceptual model provides the additional benefit of relating (ideally, semantically interoperating) these multiple non-conceptual models. After all, the latter are derivations of the *very same model*. In other words, since non-conceptual models are *historically dependent* on a conceptual model, whatever value they bring to *value beneficiaries* are (indirect) values brought also by the *use* of the original conceptual model. The latter, however, has the additional benefits of binding, interconnecting and providing a space of design exploration for utility-design models derived from it.

Complexity Management and Views

As discussed in e.g. [2, 17, 43], views are positioned as providing a powerful mechanism to create domain models that are more suitable in the communication with different stakeholders (and for different purposes) than the 'full scale' model would provide.

At a fundamental level, views are a complexity management mechanism that supports making complex models *cognitively tractable*, while also tuning this to the audience and their concerns/interests at hands. In this vein, [36] defines the notion of *view*, in the context of the architecture of software systems, as: "A representation of a whole system from the perspective of a related set of concerns". Based on this definition, [43] speaks about a view as having an "underlying model", making it explicit that a view is indeed based on an *underlying model*. At the same time, the fact that a view provides a *representation of a whole system from the perspective of a related set of concerns* implies that a view is a model as well. In line with this, we currently understand a *view* on another *domain model* (and the modeled domain) as being:

> A *domain model* of the modeled domain, which differs from the original domain model, while:
>
> - being at least at the same level of conceptual fidelity,
> - and provide a coherent subset of the information as (potentially) provided by the original domain model.

which we hold as being a generalization of the way(s) the notion of view is used in [17, 36, 43]. For instance, the "from the perspective of a related set of concerns" [36] corresponds to the need of a view to provide a *coherent* subset of the information as provided by the original domain model. Note that, since a view is a domain model as well, one can recursively create views on views.

We do realize that the "information as provided by the original domain model" may be hard to formalize, as the "information as in potential provided" does depend on the observer of the model[11]. This is also the reason why we have added the "*in potential*" qualification. For instance, a large

Footnote 10 (continued)

models can be instrumental both in their own evolution by supporting a progressive process of modeling, as well as in the construction of the very underlying domain conceptualization to which they will (eventually) be isomorphic to.

[11] This is actually analogous to the challenge of defining what information can potentially be provided by an 'information carrier' in general, in the context of information retrieval systems. A theoretical framework to explicitly about this has been reported in [9, 56]. It remains a possible avenue for further research to apply this in the context of models and views. At least for some models with certain explanatory functions, the information content of a model can be associated with its ability to answer *why-questions* [65].

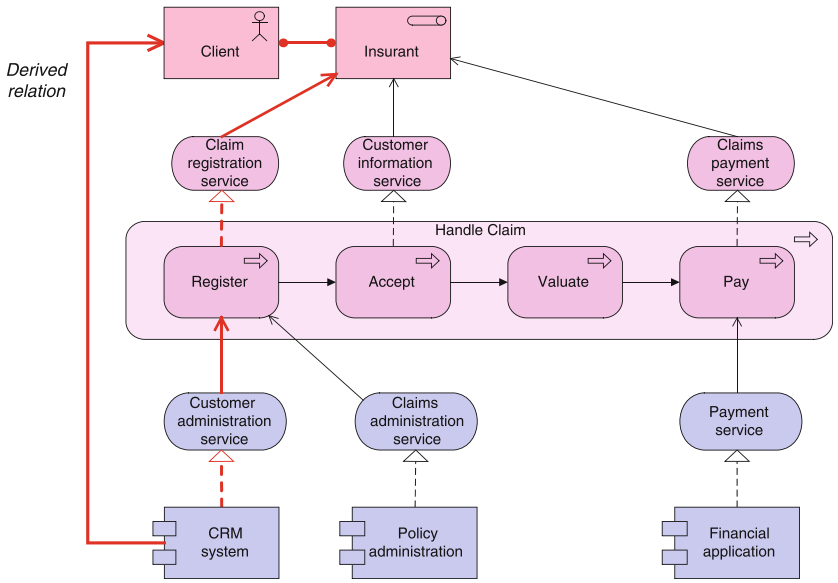

Fig. 2 Derived relations in an ArchiMate model; adopted from [41]

domain model might be so (cognitively) 'overwhelming' to an observer that they might actually glean more information from the view than from the original model.

When the involved domain models (i.e., the view and the original domain model) are represented in terms of explicit modeling languages, one can mathematically think of these models as being typed-graphs that are typed in terms of the modeling concepts as provided by the modeling language(s). In that case, the notion of "proper subset of the *potential* information as provided by the original domain model" can be formalized by requiring there to be a function that maps a sub-graph of the (implied) graph representing the original domain model to the graph that represents the view, where this function respects ontological commitment(s) of the used modeling language(s). Given such a sub-graph and mapping function, the requirement that the (potential) information provided by a view should be *coherent*, then corresponds to the requirement that the sub-graph (representing the view should) be a connected graph.

It is important to note that models (and views) are not required to be 'minimal'. It is allowed for models to contain elements that can be derived from other parts of the model. This is also why, above, we added 'implied' when writing "(implied) graph representing the original domain model".

Consider, for instance, the derived relationships [43] as featured in the ArchiMate [4][12] modeling language for enterprise architecture. Such derived relationships may, depending on the purpose at hand, may be included in the model/view or not. An example, taken from [41], is provided in Fig. 2. A client *uses* (the red arrow marked Derived relation) the CRM system. This is derived from the fact that the Client is *assigned* to the role of Insurant, which *uses* the Claim registration service, which is *realized by* the Register business process, which *uses* the Customer administration service, and which is finally *realized* by the CRM system. This view (on a larger ArchiMate model) contains the derived relation as well as the underlying (more basic) relations. Even though including such a derived relation in a view (such as the one depicted in Fig. 2) does not carry new information in an objective sense, adding it might make an observer more explicitly aware of the derived relationship. Even more, depending on the modeling language at hand, such derived relationships enable the creation of more 'compacted' views. Consider, for instance, Fig. 3 (based on an example from from [41]). On the left hand side, we see there is a data *flow* from Function 1 to Function 2. As these functions are assigned to Component 1 and Component 2 respectively, the ArchiMate model allows us to conclude that there is a data *flow* from Component 1 to Component 2. This also allows us to create a view where the functions are not shown, as illustrated on the right hand side of Fig. 3.

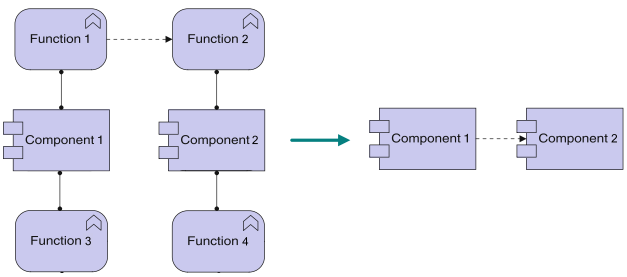

Fig. 3 Compacting views; based on an example used [41]

[12] ArchiMate is an industry standard managed by The Open Group, similar to e.g. BPMN [49] and UML [48], which are managed by the Open Management Group (OMG).

In terms of the above discussed possible formalization of views, we currently posit there to be four 'stackable' operations to construct views:

1. *Selection* – involving the focusing of the view on a specific part of the original model. In this case, the mapping function should involve a bijective function between the (selected!) part(s) of the graph of the original domain model, and the view's graph. The central question in using this operation is *what is the (sub)domain to focus on in the view?* When one would be taking a photo of a subject, this would correspond to the question of where to point the camera at. In this case, as the metaphor goes, the angle (perspective) from which the picture (i.e., model) is taken reveals or occludes certain elements from the scene (i.e., domain). In a system development context this operation pertains to both the scoping of what is to be included in the model/view in terms of e.g. enterprise-wide, business unit specific, etc, as well as the perspective in terms of the high level structures the well-known 'engineering frameworks' (e.g. [4, 7, 14, 18, 71, 72, 75, 80]). These frameworks usually take the form of a two-dimensional structure involving different 'cells' organized in columns and rows. For each of the 'cells' of the latter engineering frameworks, one can create a view on the model of the system (of systems) as a whole.
2. *Distillation* – involving a further abstracting away from the original domain, by distilling specific aspects of the domain. The central question in using this operation is *what phenomena to include in the view?* In terms of the analogy of taking photos, this would correspond to the question if one would make a color photo, a gray-scale photo, or possibly even an infra-red photo. In a system development context, this is where we find the need to hone in on specific aspects (e.g. process flows, resource use, information flows, etc), and/or cross-cutting concerns (e.g. security, privacy, sustainability, etc.). The earlier example shown in Fig. 3 is an example of distilling in the sense of blending out functions, while still maintaining an interest in the data flows. In 'distilling', one *may* need to combine certain elements/properties from the original domain model. Therefore, in this case, the mapping from the (selected part of the) original domain model to the view's graph would involve a surjective (but possibly bijective) function. As an example of this surjectivity, building on Fig. 3, Fig. 4 shows (left and right) two original models that could both be 'distilled' to the compacted view on the right hand side of Fig. 3.
3. *Summarization* – also involves a further abstracting away from the original domain, but now by clustering of different elements in the original domain model into more coarse grained elements. As a result, the mapping from the (selected part of the) original domain model to the view's graph would again involve a surjective function, but in this case, this is not allowed to be a bijective function. A typical example would be a process model, where a summarized view is created in which the decomposition of the process is not included. The leading question here is *what level of detail is needed?* In terms of taking a photo, this is the question of the level at which one zooms in/out on the subject in relation to the resolution of the optical sensors as used in the camera.
4. *Translation* – involving a translation between one modeling language to another modeling language (and medium). This implies that the mapping needs to provide a 'translation' between the modeling languages used. The main question for this operation is *what is the best language and medium to represent the view?* In a system development context, this is where one may find a variety of representations that are tuned to different stakeholders in terms of e.g. heat-maps, matrix-like representations, textual descriptions combined with infographics, animations, etc. In section 7, we will return to this point in terms of the representation of a model actually involving a connection between its informational payload and a concrete 'medium system'.

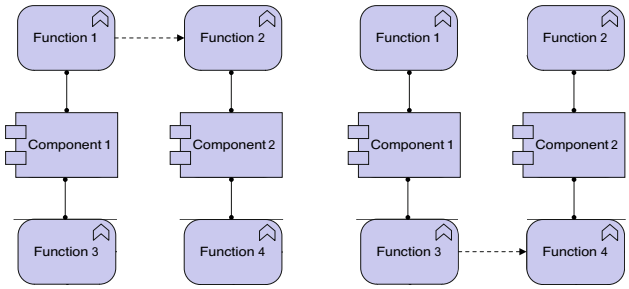

Fig. 4 Surjectivity in distilling views

The needed mix of operations used in creating a specific view, depends on the specific purpose for the view at hand. Even more, given one domain model, one can even construct an entire hierarchy of views, using the different operations.

As mentioned above, in the context of system development, the *selection* operation has a natural link to the high level perspectives of engineering frameworks. Collectively, the views that correspond to the different cells in such frameworks provide different 'chunks' that make up the model of the entire (as-is/to-be) system. In addition, strategies exist that enable a *recoding* or *modularization* of models using an underlying foundational ontology. In *model recoding* [16], models are re-organized by grouping elements in terms of higher-granularity modeling primitives. In *model*

modularization [30], models are reorganized in cognitively tractable chunks that can be understood as a whole.

The *distillation*, *summarization*, and *translation* operations are directly linked to the question of the concepts and relations to be used in modeling the domain. In other words, the ontological commitment by which we will look at the domain. This is where we find the meta-models.[13] underlying actual modeling languages and methods (e.g. [4, 48]), as well as explicit 'content frameworks' (e.g. [75, 80]) that (albeit without a 'concrete syntax') do define the concepts and relation in terms of which the system is to be observed and modeled.

The question of *what level of detail is needed?* behind the *summarization* operation is also directly linked to the role of views as a complexity management mechanism to make complex models *cognitively tractable*. What is needed for summarization is some kind of '(un)folding' mechanism. Having a (recursive) (un)folding mechanism also enables the construction of a dynamic hierarchy of views, that allows for a navigation in terms of zooming in/out akin to the way we use Google Maps.

The needed (un)folding mechanism can be based on *mereology* (i.e., part-whole relations), as well as different forms of 'attribution'. For instance, in the case of processes, it is quite commonplace to decompose these in terms of their mereological structure, i.e., decomposing a process into smaller temporal parts (sub-processes/tasks). In the case of organizational structures, the organizational hierarchy (which is structure of delegation power) coincides with a mereological structure in which organizations, their branches and units are decomposed into other smaller (functional) parts. 'Attribution' refers to the fact that one concept might be considered as an attribute of another concept. For instance, the height of a person, the name of a person, etc. are attributes of a person. This 'attribution' can be applied recursively in the sense that a person in the role of a manager of a department, might be seen as an attribute of that department [12].

General strategies for the (un)folding mechanisms needed for model summarization have been the subject of study in the past in the context of dealing with large conceptual models [11, 12, 34], as well as more recently utilizing foundational ontologies to generate ontologically founded (un) foldings [27, 64] of large conceptual models. The general idea of the latter approaches is to let the models undergo an (automatic) lossy transformation based on the underlying foundational ontology, to yield another model (a summarizing view) that captures the gist of what the original model was about. By applying this recursively, a hierarchy of (ontology based) summarizations results.

Since a view involves a mapping from the original model to the view that is generally a surjective function (and not a bijective one), updating/editing a view can lead to a variation of the 'view update' problem as known from the field of databases. Operationally this means that if a change is made in a view based on some original domain model, it may not be clear how to then make a corresponding change in the original domain model.

In practice, this 'view update' problem becomes even more pressing as in the context of system development one is likely to actually start modeling a system from different angles; not unlike taking photos of the same object. Of course, knowing that these would be models of the same system, would imply that these models are essentially views of a larger model. Indeed, during a modeling process, one may use views to gradually (in a bottom up fashion) construct the 'larger picture'.

A challenge is, of course, to ensure linkages between these views to maintain consistency and maintain/obtain the 'larger picture' (see the discussion in e.g. [8]). As the different 'cells' of the aforementioned 'engineering frameworks' involve a different perspective of the system (and its development) under consideration, the modeling concepts used for each of the cells will be 'cell-specific'. As a result, there will also be cell-specific meta-models. When, across an engineering framework, these cell-specific meta-models are aligned well (as is explicitly the case for, e.g., ArchiMate [4], IAF [80] and MEMO [18]), views corresponding to the different 'cells' of the framework can be connected to maintain/obtain the 'larger picture'.

Diagrams, Programs, Animations, and other Models

Before we review (some of) the 'other' model kinds, such as *tables*, *diagrams* and *animations*, it is important to have a closer look at the actual representation of models. Before doing so, however, it is important to revisit the earlier made observations that the notion of model we subscribe to does not stipulate any a priori constraints on e.g. the level of completeness, precision, or even (mathematical) formality of the model. At the same time, we would argue that a model is always expressed in terms of some modeling language. This can be a precisely (a priori) defined modeling language, but

[13] The term meta-model is often overloaded in the area. Often, it refers to the description of a language's *abstract syntax* In contrast, we are using it here in the sense of what is termed the *ontological meta-model of the language*, or simply, the *ontology of the language*. This corresponds to a model that captures the worldview (or rather *conceptualization*) that is embedded in the language's modeling primitives. For example, Peter Chen's Entity Relationship model commits to a worldview/conceptualization that accounts for the existence of four types of things: entity, relationship, attribute and attribute value spaces [23].

can equally well involve a highly informal ad-hoc (emerging) language when e.g. sketching on the back of a napkin.

When zooming in on modeling languages, one can make a distinction between the conventions that govern what constructions are allowed in the language, and conventions that govern the way these are concretely presented in terms of representational mechanisms associated to a medium system. The former corresponds to the *grammar* and the *abstract syntax* of the language, while the latter pertains to the *concrete syntax*. The *concrete syntax* also ties a model's *abstract syntax* to an actual medium system. Obvious examples of medium systems that can be used to 'render' the concrete syntax models include paper (including the back of a napkin), a 2D vector graphics (and textual fonts) rendering engine, or a simulation engine. Less obvious examples, include game engines or even tangible objects, in order to create models (and views) that may provide a more tangible experience.

Building on this, it is also important to note that a model (qua representation on a medium system) may be of a static nature or of a dynamic (and even an interactive) nature. This distinction is actually orthogonal to the question if the modeled domain is static or dynamic. If a domain is static, one could, indeed imagine creating a model with a static representation, such as a simple paper-based 'org chart' of the structure of an organization. However, one could also opt to e.g. create a 'navigable' model where a 'viewer' can interactively navigate over/through the model, as we e.g. already hinted at when discussing the (un)folding of models in the section "Complexity Management and Views". Conversely, a dynamic domain such as a business process can be represented in terms of a simulation or animation, but also as a static representation in terms of e.g. a BPMN [49] model.[14]

In the remainder of this section, we will argue how *specifications*, *programs*, *diagrams*, *tables*, *spreadsheets*, *simulations*, and *animations* can all essentially be seen as models; albeit with fundamentally different purposes and represented on different media systems. The chosen set (*specifications*, *programs*, *diagrams*, *tables*, *spreadsheets*, *simulations*, and *animations*) is not intended as a complete coverage of all 'things model' that may be used in a system development context. Together, however, they do illustrate the variety of the kinds of models we may come across in the *modeling practices* as embedded in system development:

- *Specification* – A specification is a model that normatively prescribes the properties of a (to be designed, to be elaborated, to happen, to be brought about, ...) phenomenon. In terms of our earlier work [31] towards a taxonomy of modeling-related goals, a specification has a world-to-model direction of fit (in a world-to-word vein [70]) with the aim to *change the world* (the world in the sense of the phenomenon which' properties are described normatively). Specifications tend to be represented using precise/formal languages on 2D text/graphics based medium systems. For instance, a specification of business rules in as a text file in controlled natural language format, a mathematical specification on (digital) paper, or a graphical Petri-net based specification.
- *Program* – A specification which models the *required behavior of a computer* in an actionable way, such that a computer can directly exhibit this required behavior (via interpretation or compilation). Traditionally, programs are specified in some controlled textual form. In the past, programs were specified in terms of other medium systems, such as punched cards. Meanwhile, there has also been an increase in the use of visual ways to represent programs. Some of the modern editors for programs allow for some forms of (un)folding, de-facto resulting in a more dynamic (navigable) representation. Although we consider programs as models (again, of computation), we do not consider programming languages as appropriate conceptual modeling languages. Programming languages are designed with computational concerns in mind (e.g., computational complexity, performance, to facilitate compiler construction) and as a result, for the purpose of conceptual modeling, they: compromise expressivity and conceptual fidelity; hinder separation of concerns by forcing the modeler to consider at the same time conceptual, design and implementation issues.[15]
- *Diagrams* – Diagrams, in particular involving boxes-and-lines, are a common way to represent models. In general, diagrams involve some (static) graphical structure, possibly adorned with icons and/or text. In principle, diagrams provide a static representation. However, as hinted at before, such models can be made dynamic in a Google Maps like style by (un)folding and/or blending in/out specific (types of) elements. See the earlier discussions in section "Complexity Management and Views" regarding the *distillation* and *summarization* operation for the creations of views. Diagrammatic notations are often part of the concrete syntax of general-purpose and domain-specific modeling languages alike. When designed in a proper way, diagrammatic notations can increase the pragmatic efficiency of the models it pro-

[14] Where this model can, of course, be complemented with a simulation of actual process instances.

[15] For this discussion in the context of ontology engineering, see, e.g., [23]; for the trade-off between expressivity and tractability in knowledge representation languages, see, e.g., [44]. Another manifestation of this problem is the well-known impedance mismatch problem in mapping ontologically-rich conceptual models to relational databases, see, e.g., [22].

duces [25, 29]. However, in order to attain these properties, these notations have to be properly designed [45] lest denting problem-solving and producing unintended cognitive inferences (unintended *implicatures*) [25, 29].

- *Table* – A table essentially provides a two-dimensional grid representation on a 2D medium (such as paper or a computer screen) that can capture a (possibly derived) ternary relation (type) concerning the modeled domain. In principle, a table is a static representation. However, by 'allowing' one to blend in/out specific rows/columns, thus changing the 'informational payload' which the table (qua model) provides to us at that moment, the representation becomes interactive.
- *Spreadsheet* – A spreadsheet is a specific way to represent/render one or more connected tables on a computational medium system. Using formulas, derived parts can be included as well. A spreadsheet with 'intentionally left open' cells to enable 'what if analysis' is an example of an *interactive* model as it allows one to 'play' with the model. Spreadsheets, with their traditional numbered columns and rows, are, of course, not the most suitable to capture the structures of a domain in a clear way.
- *Animation* – A model that is represented on a video-based medium system (i.e. a 'movie') that illustrates the dynamic behavior in the modeled domain in terms of the involved agents, subjects, etc
- *Simulation* – A model that is represented on a simulation engine (as the medium system) and that provides a simulation of the dynamic behavior of the modeled domain. If simulation-runs can be generated 'on the fly' based on different scenario's, the simulation (qua model) becomes an interactive model Note: a 'screenshot' of an animation or a simulation, can be seen as a model as well. In that case, it would be a view based on a 'temporal distillation'.

Conclusion

We started this paper with the observation that humanity has, quite naturally, developed a wide range of *modeling practices*. We also observed that when the models as used in such *modeling practices* have a key role to play in the activities in which these *modeling practices* are 'embedded', the need emerges to consider the effectiveness and efficiency of such processes, and speak about *modeling capabilities*. Based on this, we argued for a need to develop a thorough understanding of the artifacts involved in the modeling practices/capabilities.

From this general backdrop, we then zoomed in on *model-driven (systems) development* in a general sense (including software engineering, information systems engineering, business process engineering, enterprise engineering, and enterprise architecture management) as an area where models play an (increasingly) important role.

In this context, we zoomed in on a variety of model(ing) related artifacts, such as views, diagrams, programs, animations, specifications, etc, that play an important role in the *modeling practices* that take place in the context of system development, While doing so, we also introduced a *framework for understanding* to position the variety of modeling related artifacts, where we also took the view that these artifacts are to be seen as specific kinds of models, albeit for fundamentally different purposes. The foundation of the presented *framework for understanding* is formed by a philosophically and linguistically grounded definition of domain model, where we also positioned domain models as complex speech acts. Based on this, foundation, the presented framework also includes definitions of the *Return on Modeling Effort* (RoME), and the *conceptual fidelity* of domain models, as *views* as a mechanism to manage the complexity of domain models.

In future work, we expect to further evolve the presented framework. More specifically, we also intend to develop a more completer ontology of models dealing with aspects such as the mereology of models, models as artifacts (i.e., property connecting modeling acts to intentions), identity as aspects of models and how they relate to other artifacts in the ecosystem of modeling (in the spirit of the ontology of software as proposed in [77]). Finally, we intend to investigate the role of different types of *assumptions* [78] to modeling practices and to models as artifacts.

Funding Open access funding provided by TU Wien (TUW).

Declarations

Conflict of interest The authors declare that they have no competing interests.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Apostel L. Towards the formal study of models in the non-formal sciences. Synth Int J Epistemol, Methodol Philos Sci. 1960;12:125–61.
2. Arbab F, de Boer FS, Bonsangue M, Lankhorst MM, Proper HA, van der Torre LWN. Integrating architectural models: symbolic, semantic and subjective models in enterprise architecture. Enterp Model Inf Syst Archit. 2007;2(1):40–57. https://doi.org/10.18417/emisa.2.1.4.
3. Azevedo CLB, Iacob ME, Almeida JPA, van Sinderen MJ, Ferreira Pires L, Guizzardi G. Modeling resources and capabilities in enterprise architecture: a well-founded ontology-based proposal for ArchiMate. Inf Syst. 2015;54:235–62. https://doi.org/10.1016/j.is.2015.04.008.
4. Band I, Ellefsen T, Estrem B, Iacob ME, Jonkers H, Lankhorst MM, Nilsen D, Proper HA, Quartel DAC, Thorn S. ArchiMate 3.0 Specification. The Open Group, Reading, Berkshire, United Kingdom (2016)
5. Bjeković M, Proper HA, Sottet JS. Embracing pragmatics. In: Yu ESK, Dobbie G, Jarke M, Purao S (eds.) Conceptual Modeling – 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27–29, 2014. Proceedings, Lecture Notes in Computer Science, vol. 8824, pp. 431–444. Springer, Berlin, Germany (2014). https://doi.org/10.1007/978-3-319-12206-9_37
6. Bjeković M, Sottet JS, Favre JM, Proper HA. A framework for natural enterprise modelling. In: IEEE 15th Conference on Business Informatics, CBI 2013, Vienna, Austria, July 15–18, 2013. IEEE Computer Society, Washington, DC, United States of America (2013).https://doi.org/10.1109/CBI.2013.20. https://ieeexplore.ieee.org/xpl/conhome/6642227/proceeding.
7. Boar BH. Constructing blueprints for enterprise IT Architectures. New York City: Wiley; 1999.
8. Boiten E, Bowman H, Derrick J, Linington P, Steen M. Viewpoint consistency in ODP. Comput Netw. 2000;34(3):503–37.
9. van Bommel P, Proper HA, van der Weide TP. Information coverage in advisory brokers. Int J Intell Syst. 2007;22(11):1155–88. https://doi.org/10.1002/int.20240.
10. Boon M, Knuuttila T. Models as epistemic tools in engineering sciences. In: Philosophy of technology and engineering sciences. NY: Elsevier; 2009. p. 693–726.
11. Campbell LJ, Halpin TA, Proper HA. Conceptual schemas with abstractions: making flat conceptual schemas more comprehensible. Data Knowl Eng. 1996;20(1):39–85. https://doi.org/10.1016/0169-023X(96)00005-5.
12. Creasy PN, Proper HA. A generic model for 3-dimensional conceptual modelling. Data Knowl Eng. 1996;20(2):119–61. https://doi.org/10.1016/0169-023X(95)00043-R.
13. Cruse A. Meaning in language, an introduction to semantics and pragmatics. Oxford, United Kingdom: Oxford University Press; 2000.
14. DoD Deputy Chief Information Officer: The DoDAF architecture framework version 2.02 (2011). http://tinyurl.com/zhw3kaf.
15. Falbo RA, Guizzardi G, Duarte KC. An ontological approach to domain engineering. In: Proceedings of the 14th international conference on Software engineering and knowledge engineering; 2002. pp. 351–358.
16. Figueiredo G, Duchardt A, Hedblom MM, Guizzardi G. Breaking into pieces: An ontological approach to conceptual model complexity management. In: 2018 12th International Conference on Research Challenges in Information Science. IEEE Computer Society, Washington, DC, United States of America. RCIS; 2018. pp. 1–10.
17. Frank U. Multi-perspective enterprise modeling (MEMO) – Conceptual framework and modeling languages. In: 35th Hawaii International Conference on System Sciences (HICSS-35 2002), CD-ROM / Abstracts Proceedings, 7–10 January 2002, Big Island, HI, USA. IEEE Computer Society, Washington, DC, United States of America; 2002. pp. 1258–1267. https://doi.org/10.1109/HICSS.2002.993989.
18. Frank U. Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges. Softw Syst Model. 2014;13:941–62. https://doi.org/10.1007/s10270-012-0273-9.
19. Frigg R. Scientific representation and the semantic view of theories. Theor Rev Teor, Hist Fundam Cienc. 2006;21(1):49–65.
20. Guarino N, Guizzardi G, Mylopoulos J. On the philosophical foundations of conceptual models. Inf Model Knowl Bases XXXI. 2020;321:1–15. https://doi.org/10.3233/FAIA200002.
21. Guarino N, Porello D, Guizzardi G. On weak truthmaking. In: International Workshop on Foundational Ontology (FOUST 2019); 2019.
22. Guidoni GL, Almeida JPA, Guizzardi G. Preserving conceptual model semantics in the forward engineering of relational schemas. Front Comput Sci. 2022;4:1–20. https://doi.org/10.3389/fcomp.2022.1020168.
23. Guizzardi G. On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In: Vasilecas O, Eder J, Caplinskas A (eds.) Proceedings of the 2007 Conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB &IS'2006. IOS Press, Amsterdam, The Netherlands; 2007. pp. 18–39. https://doi.org/10.5555/1565421.1565425.
24. Guizzardi G. Theoretical foundations and engineering tools for building ontologies as reference conceptual models. Semant Web. 2010;1(1–2):3–10.
25. Guizzardi G. Ontology-based evaluation and design of visual conceptual modeling languages. In: Reinhartz-Berger I, Sturm A, Clark T, Cohen S, Bettin J, editors. Domain engineering. Berlin, Germany: Springer; 2013. p. 317–47. https://doi.org/10.1007/978-3-642-36654-3_13.
26. Guizzardi G, Ferreira Pires L, van Sinderen MJ. An ontology-based approach for evaluating the domain appropriateness and comprehensibility appropriateness of modeling languages. In: Briand L, Williams C, editors. MODELS 2005: Model Driven Engineering Languages and Systems, vol. 3713. Lecture notes in computer science. Berlin, Germany: Springer; 2005. p. 691–705. https://doi.org/10.1007/11557432_51.
27. Guizzardi G, Figueiredo G, Hedblom MM, Poels G. Ontology-based model abstraction. In: 2019 13th International Conference on Research Challenges in Information Science. RCIS; 2019. pp. 1–13.https://doi.org/10.1109/RCIS.2019.8876971.
28. Guizzardi G, Guarino N. Explanation, semantics, and ontology. Data Knowl Eng. 2024;153:102325. https://doi.org/10.1016/j.datak.2024.102325.
29. Guizzardi G, Pires LF, van Sinderen MJ. On the role of domain ontologies in the design of domain-specific visual modeling languages. In: Proceedings of the 2nd Workshop on Domain-Specific Visual Languages. ACM Press, New York City, United States of America; 2002.
30. Guizzardi G, Prince Sales T, Almeida JPA, Poels G. Automated conceptual model clustering: a relator-centric approach. Softw Syst Model. 2022;21(4):1363–87. https://doi.org/10.1007/s10270-021-00919-5.
31. Guizzardi G, Proper HA. On understanding the value of domain modeling. In: Guizzardi G, Prince Sales T, Griffo C, Furnagalli M (eds.) Proceedings of 15th International Workshop on Value Modelling and Business Ontologies (VMBO 2021), Bolzano, Italy, 2021, CEUR Workshop Proceedings, vol. 2835. CEUR-WS.org, Aachen, Germany; 2021.

32. Harel D, Rumpe B. Meaningful modeling: what's the semantics of "semantics"? IEEE Comput. 2004;37(10):64–72. https://doi.org/10.1109/MC.2004.172.
33. Henderson-Sellers B, Gonzalez-Perez C, Walkerden G. An application of philosophy in software modelling and future information systems development. In: Franch X, Soffer P (eds.) Advanced Information Systems Engineering Workshops. Springer, Berlin, Germany; 2013. pp. 329–340. https://doi.org/10.1007/978-3-642-38490-5_31.
34. ter Hofstede AHM, Proper HA, van der Weide TP. Data modelling in complex application domains. In: Loucopoulos P (ed.) Advanced Information Systems Engineering, CAiSE'92, Manchester, UK, May 12–15, 1992, Proceedings, Lecture Notes in Computer Science, vol. 593. Springer, Berlin, Germany; 1992. pp 364–377. https://doi.org/10.1007/BFb0035142.
35. Hoppenbrouwers SJBA, Proper HA, van der Weide TP. A fundamental view on the process of conceptual modeling. In: Delcambre LML, Kop C, Mayr HC, Mylopoulos J, Pastor López Ó (eds.) Conceptual Modeling – ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24–28, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3716. Springer, Berlin, Germany; 2005. pp. 128–143. https://doi.org/10.1007/11568322_9.
36. IEEE Computer Society: Recommended practice for architectural description of software intensive systems. Tech. Rep. IEEE P1471:2000, ISO/IEC 42010:2007, Piscataway, New Jersey; 2000.
37. ISO/IEC JTC 1/SC 32 Technical Committee on Data management and interchange: Information processing systems – Concepts and terminology for the conceptual schema and the information base. Tech. Rep. ISO/TR 9007:1987, ISO; 1987.
38. Kecheng L, Clarke RJ, Andersen PB, Stamper RK, Abou-Zeid ES (eds.) IFIP TC8-WG8.1 Working Conference on Organizational Semiotics – Evolving a Science of Information Systems. Kluwer, Deventer, The Netherlands; 2002.
39. Krogstie J, Lindland OI, Sindre G. Defining quality aspects for conceptual models. In: Falkenberg ED, Hesse W, Olivé A (eds.) Information System Concepts: Towards a Consolidation of Views, Proceedings of the IFIP International Working Conference on Information System Concepts (ISCO 1995), Marburg, Germany, 28–30 March 1995, *IFIP Conference Proceedings*, vol. 26. Chapman & Hall, London, United Kingdom; 1995. pp. 216–231.
40. Op 't Land M, Proper HA, Waage M, Cloo J, Steghuis C. The results of enterprise architecting. In: The enterprise engineering series. Berlin, Germany: Springer; 2008. https://doi.org/10.1007/978-3-540-85232-2_4.
41. Lankhorst MM, Arbab F, Bekius SF, Bonsangue M, Bosma H, Campschroer J, Cuvelier MJ, Fennema P, Groenewegen L, Hoppenbrouwers SJBA, Iacob ME, Janssen WPM, Jonkers H, Krukkert D, Penders PGM, Proper HA, Slagter RJ, Stam AW, Steen MWA, Wieringa RJ, de Boer FS, ter Doest HWL, van Buuren R, van Eck PAT, van Leeuwen D, van der Torre LWN, Veldhuijzen van Zanten GE. Enterprise architecture at work - modelling, communication and analysis. In: The enterprise engineering series. 4th ed. Berlin, Germany: Springer; 2017. https://doi.org/10.1007/978-3-662-53933-0.
42. Lankhorst MM, van der Torre LWN, Proper HA, Arbab F, de Boer FS, Bonsangue M. Foundations: enterprise architecture at work - modelling, communication and analysis. In: The enterprise engineering series. 4th ed. Berlin, Germany: Springer; 2017. https://doi.org/10.1007/978-3-662-53933-0_3.
43. Lankhorst MM, van der Torre LWN, Proper HA, Arbab F, Steen MWA. Viewpoints and visualisation: enterprise architecture at work - modelling, communication and analysis. In: The enterprise engineering series. 4th ed. Berlin, Germany: Springer; 2017. p. 171–214. https://doi.org/10.1007/978-3-662-53933-0_8.
44. Levesque HJ, Brachman RJ. Expressiveness and tractability in knowledge representation and reasoning 1. Comput Intell. 1987;3(1):78–93.
45. Moody DL. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans Softw Eng. 2009;35(6):756–79.
46. Morris CW. Signs, Language and Behaviour. Hoboken, New Jersey, United States of America: Prentice Hall/Braziller; 1946.
47. Ogden CK, Richards IA. The meaning of meaning: a study of the influence of language upon thought and of the science of symbolism. Oxford: Magdalene College, University of Cambridge; 1923.
48. OMG: UML 2.0 superstructure specification – Final adopted specification. Tech. Rep. ptc/03-08-02, Object Management Group, Needham, Massachusetts, United States of America; 2003.
49. OMG: Business process model and notation (BPMN), version 2.0. Tech. rep.; 2011. http://www.omg.org/spec/BPMN/2.0/.
50. Osis J, Donins U. Topological UML modeling. In: Topological UML modeling - an improved approach for domain modeling and software development, computer science reviews and trends. Boston: Elsevier; 2017. p. 133–51. https://doi.org/10.1016/B978-0-12-805476-5.00005-8.
51. Prince Sales T, Guarino N, Guizzardi G, Mylopoulos J. An ontological analysis of value propositions. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference. IEEE Computer Society, Washington, DC, United States of America. EDOC; 2017. pp. 184–193 https://doi.org/10.1109/EDOC.2017.32.
52. Proper HA. TEE group: Focus & drives—Return on modelling effort; 2005. http://www.cs.ru.nl/tee/focus-drives.htm.
53. Proper HA. Models that matter; Return on modelling effort. Blog; 2009. http://erikproper.blogspot.com/2009/02/models-that-matter-return-on-modelling.html.
54. Proper HA. On model-based coordination of change in organizations. In: Aier S, Rohner P, Schelp J, editors. Engineering the transformation of the enterprise: a design science research perspective. Berlin, Germany: Springer; 2021. https://doi.org/10.1007/978-3-030-84655-8_6.
55. Proper HA. Keynote: On views, diagrams, programs, animations, and other models. In: Mayo FJD, Ferreira Pires L, Seidewitz E (eds.) Proceedings of the 11th International Conference on Model-Based Software and Systems Engineering, MODELSWARD 2023, Lisbon, Portugal, February 19–21, 2023. SCITEPRESS; 2023. pp. 13–14.
56. Proper HA, Bruza PD. What is information discovery about? J Am Soc Inf Sci. 1999;50(9):737–50. https://doi.org/10.1002/(SICI)1097-4571(1999)50:9<737::AID-ASI2>3.0.CO;2-C.
57. Proper HA, Guizzardi G. On domain modelling and requisite variety: Current state of an ongoing journey. In: Grabis J, Bork D (eds.) The Practice of Enterprise Modeling: 13th IFIP Working Conference, PoEM 2020, Riga, Latvia, November 25–27, 2020, Proceedings, Lecture Notes in Business Information Processing, vol. 400. Springer, Berlin, Germany; 2020. pp. 186–196. https://doi.org/10.1007/978-3-030-63479-7_13.
58. Proper HA, Guizzardi G. On domain conceptualization. In: Aveiro D, Guizzardi G, Pergl R, Proper HA (eds.) Advances in Enterprise Engineering XIV – 10th Enterprise Engineering Working Conference, EEWC 2020, Bozen-Bolzano, Italy, September 28, October 19, and November 9–10, 2020, Revised Selected Papers, *Lecture Notes in Business Information Processing*, vol. 411. Springer, Berlin, Germany; 2021. pp. 49–69.https://doi.org/10.1007/978-3-030-74196-9_4.
59. Proper HA, Guizzardi G. Modeling for enterprises; Let's go to RoME ViA RiME. In: Clark T, Zschaler S, Barn B, Sandkuhl K (eds.) Proceedings of the Forum at Practice of Enterprise Modeling 2022 (PoEM-Forum 2022) Co-located with PoEM 2022, London, United Kingdom, November 23–25, 2022, vol. 3327.

CEUR-WS.org, Aachen, Germany; 2022. pp. 4–15. https://ceur-ws.org/Vol-3327/paper02.pdf.
60. Proper HA, Guizzardi G. On views, diagrams, programs, animations, and other models. In: Strecker S, Jung J, editors. Informing possible future worlds: essays in honour of Ulrich Frank. Berlin, Germany: Logos Verlag; 2024. p. 123–38. https://doi.org/10.30819/5768.
61. Proper HA, Verrijn-Stuart AA, Hoppenbrouwers SJBA. On utility-based selection of architecture-modelling concepts. In: Hartmann S, Stumptner M (eds.) Conceptual Modelling 2005, Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, NSW, Australia, January–February 2005, Conferences in Research and Practice in Information Technology Series, vol. 43. Australian Computer Society, Sydney, New South Wales, Australia; 2005. pp. 25–34. https://crpit.scem.westernsydney.edu.au/abstracts/CRPITV43Proper.html.
62. Quillian MR. Semantic memory, semantic information processing. Ph.D. thesis, MIT, Boston, Massachusetts, United States of America; 1968.
63. Razavi S, Tolson BA, Burn DH. Review of surrogate modeling in water resources. Water Resour Res. 2012;48:7. https://doi.org/10.1029/2011WR011527.
64. Romanenko E, Calvanese D, Guizzardi G. Abstracting ontology-driven conceptual models: Objects, aspects, events, and their parts. In: Research Challenges in Information Science: 16th International Conference. RCIS 2022, Barcelona, Spain, May 17–20, 2022, Proceedings. Berlin, Germany: Springer; 2022. pp. 372–88.
65. Romanenko E, Calvanese D, Guizzardi G. Towards pragmatic explanations for domain ontologies. In: Knowledge Engineering and Knowledge Management: 23rd International Conference. EKAW 2022, Bolzano, Italy, September 26–29, 2022, Proceedings. Berlin, Germany: Springer; 2022. pp. 201–8.
66. Rothenberg J. The nature of modeling. In: Widman LE, Loparo KA, Nielsen NR, editors. Artificial intelligence, simulation & modeling. New York City, United States of America: Wiley; 1989. p. 75–92.
67. Sandkuhl K, Fill HG, Hoppenbrouwers SJBA, Krogstie J, Matthes F, Opdahl AL, Schwabe G, Uludag Ö, Winter R. From expert discipline to common practice: a vision and research agenda for extending the reach of enterprise modeling. Bus Inf Syst Eng. 2018;60(1):69–80. https://doi.org/10.1007/s12599-017-0516-y.
68. Searle JR. Speech acts: an essay in the philosophy of language. Cambridge, United Kingdom: Cambridge University Press; 1969.
69. Searle JR. A taxonomy of illocutionary acts. Cambridge, United Kingdom: Cambridge University Press; 1979. https://doi.org/10.1017/CBO9780511609213.003.
70. Searle JR, Willis S, et al. Intentionality: an essay in the philosophy of mind. Cambridge, United Kingdom: Cambridge University Press; 1983.
71. Sowa JF, Zachman JA. Extending and formalizing the framework for information systems architecture. IBM Syst J. 1992;31(3):590–616. https://doi.org/10.1147/sj.313.0590.
72. Spewak SH. Enterprise architecture planning: developing a blueprint for data, applications, and technology. New York City, United States of America: Wiley; 1993.
73. Stachowiak H. Allgemeine modelltheorie. Berlin, Germany: Springer; 1973. https://doi.org/10.1007/978-3-7091-8327-4.
74. Thalheim B. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In: Handbook of conceptual modeling: theory, practice, and research challenges. Berlin, Germany: Springer; 2011. p. 543–77. https://doi.org/10.1007/978-3-642-15865-0_17.
75. The Open Group. TOGAF version 9.1: TOGAF series. Zaltbommel, The Netherlands: Van Haren Publishing; 2021.
76. Ullmann S. Semantics: an introduction to the science of meaning. Oxford, United Kingdom: Basil Blackwell; 1967.
77. Wang X, Guarino N, Guizzardi G, Mylopoulos J. Towards an ontology of software: a requirements engineering perspective. In: Formal ontology in information systems. Amsterdam, The Netherlands: IOS Press; 2014. p. 317–29.
78. Wang X, Mylopoulos J, Guizzardi G, Guarino N. How software changes the world: The role of assumptions. In: 2016 IEEE Tenth International Conference on Research Challenges in Information Science IEEE Computer Society, Washington, DC, United States of America. RCIS; 2016. pp. 1–12.
79. Webster M. Domain. https://www.merriam-webster.com/dictionary/domain.
80. van't Wout J, Waage M, Hartman H, Stahlecker M, Hofman A,. The integrated architecture framework explained. In: The integrated architecture framework explained: why, what, how. Berlin, Germany: Springer; 2010. p. 1–7. https://doi.org/10.1007/978-3-642-11518-9.
81. Zarwin Z, Bjeković M, Favre JM, Sottet JS, Proper HA. Natural modelling. J Object Technol. 2014;13(3):1–36. https://doi.org/10.5381/jot.2014.13.3.a4.