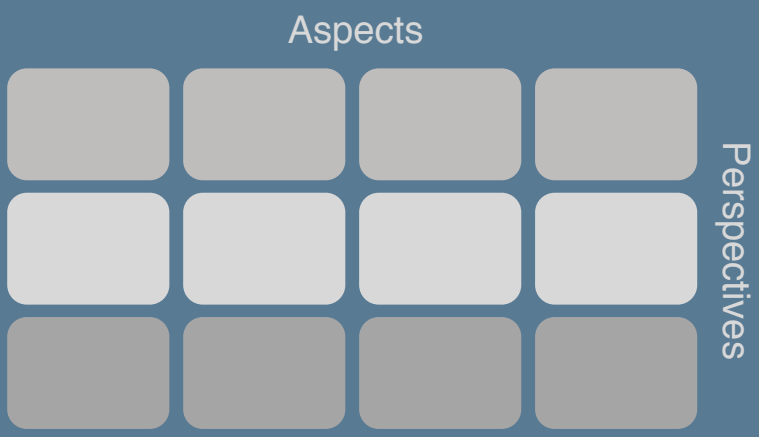
Stefan Strecker, Jürgen Jung (Eds.)

Informing Possible Future Worlds

Essays in Honour of Ulrich Frank



λογος

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.d-nb.de.

Financial support for publishing and printing by the FernUniversität in Hagen and for open access publishing by the Frankfurt University of Applied Sciences is acknowledged.

Logos Verlag Berlin GmbH 2024

ISBN 978-3-8325-5768-3
DOI 10.30819/5768

Logos Verlag Berlin GmbH
Georg-Knorr-Str. 4, Geb. 10,
12681 Berlin, Germany

Tel.: +49 (0)30 / 42 85 10 90
Fax: +49 (0)30 / 42 85 10 92
https://www.logos-verlag.com

Chapter 5

On Views, Diagrams, Programs, Animations, and Other Models

Hend*erik* A. Proper and Giancarlo Guizzardi

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 *modeling 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 becomes relevant to develop a thorough understanding of the artifacts involved in the modeling practices/capabilities. One field in which models play (an increasingly) important role is the field of system development (including software engineering, information systems engineering, and enterprise design management). In this context, we come across notions, such as views, diagrams, programs, animations, specifications, etc. The aim of this paper is to take a fundamental look at these notions. In doing so, we will argue that these notions should actually be seen as specific kinds of models, albeit for fundamentally different purposes.

5.1 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, and 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 this artifact is a model (to us, as an individual) of the observed phenomenon.

More generally, one can observe how humanity has long since used 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 wide spread, and natural (Zarwin et al. 2014) 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 speak about

modeling capabilities.[1] In the latter situation, it becomes relevant to develop a thorough understanding of the artifacts involved in the modeling practices/capabilities.

One field in which models play (an increasingly) important role is the field of system development (including software engineering, information systems engineering, and enterprise design management[2]). In this context, we come across notions, such as views, diagrams, programs, animations, specifications, etc. In this chapter, we aim to take a deeper look at these notions, where we will also argue that these notions should be seen as specific kinds of models; albeit for fundamentally different purposes.

In line with this, the remainder of this chapter is structured as follows. In Section 5.2, we start by zooming in on the notion of (domain) model. In doing so, we will discuss the importance of knowing a model's purpose and its potential Return on Modeling Effort (RoME) in particular. This will then also take us, in Section 5.3 to the notion of conceptual model, which is a class of models that has initially grown to play an important role in the field of information systems engineering, but has a much wider role to play. Identifying conceptual models as a distinct class of models, does suggests there to be a class of domain models that are not conceptual. This also results in a need to speak about *conceptual fidelity*.

Building on this grounding, Section 5.4 discusses the notion of *view* as complexity management mechanism that supports making complex models *cognitively tractable*. Finally, before concluding, Section 5.5 position notions such as diagrams, programs, and animations, as being specific kinds of models, covering different purposes, different audiences.

5.2 Domain Models and Their RoME

Based on foundational work by, e. g., Apostel (1960), and Stachowiak (1973), more recent work on the same by different authors (Rothenberg 1989; Harel and Rumpe 2004; Thalheim 2011; Sandkuhl et al. 2018), as well as our own work (Hoppenbrouwers et al. 2005; Proper et al. 2005; Guarino et al. 2020; Guizzardi 2007; Bjeković et al. 2014; Proper and Guizzardi 2020; Proper and Guizzardi 2021; Proper and Guizzardi 2022), 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*.

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'*.[3] 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.[4] In the context of system development at large, more

1 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 (Azevedo et al. 2015).

2 Used here as a generalization of (among others) Business Process Management, Enterprise Architecture Management, Enterprise Engineering, Enterprise Transformation, and Organization(al) Design.

3 In Guizzardi (2013), a more precise characterization of the notion of the domain being modeled is provided in terms of usage: *abstraction* – when referring to a *mental model* (Guarino et al. 2020) of a *particular* state of affairs (e. g., a configuration of the existing subway lines of Milan); *conceptualization* – when we are referring to general notions that are *repeatable* across multiple state of affairs (e. g., the concept of a subway line, being an ancestor, or an offspring in genealogy).

4 In Guizzardi and Proper (2021), we discussed how models can socially be seen as complex speech acts and, as such, they can be characterized as having different relations to world in terms of *directions of fit*. In particular,

specific classes of domain models include enterprise (architecture) models, business process models, ontology models, organizational models, information models, software models, 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*, 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*.[5] The *understood, and acknowledged, by a collective human agent* phrase also differentiates *domain models* from, e. g., machine-learned models. A domain model can certainly involve complex mathematical formalisms, or computer readable specifications. However, it must be transparent expressions of a human agent's mental model (or conceptualization), in line with the *cognitive purpose*. In the context of 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 this in Section 5.5).

Finally, the creation, administration, and use, of such 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 in the notion of Return on Modeling Effort, which we first coined in a publication,[6] in Op 't Land et al. (2008) while a more elaborate discussion of the concept in Guizzardi and Proper (2021) and Proper and Guizzardi (2022) is provided as part of our joint endeavor to better understand the foundations of (domain) modeling and modeling practices.

The enactment of a modeling practice and the exercise of a modeling capability is an event, an intentional event (or an action). From an ontological point of view, events are always manifestations of dispositions. Ordinary events are manifestations of interacting dispositions (Molnar 2003; Mumford and Anjum 2011). In particular, modeling events are the manifestation of the modeler's beliefs, intentions and modeling capabilities interacting with capabilities/affordances that are present in the modeling ecosystem at hand, including modeling languages (in terms of primitives, semantics and pragmatics), other models, tools, methodologies, etc. In line with the ontology of value propositions proposed in Sales et al. (2017), we speak of value experiences with its *benefits* (afforded by a system of dispositions) and *sacrifices*. This view allows us to think of the enactment of modeling practices as value

we recognized the case of the *constructive models*, i. e., models with a *double direction of fit*, namely, they represent the propositional content of a creative speech act and end up describing the entities that they help to create. Examples include patent diagrams or diagrams that, e. g., create new real estate properties by dividing a piece of land.

5 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.

6 The notion of RoME actually made its first informal appearance in Proper (2005) 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 posting (Proper 2009).

experiences and, hence, to think of RoME as *Return of Model Experiences*,[7] also underlining the fact that the value of models lies mainly in their *Value in Action* (ViA) (Proper and Guizzardi 2022) during their creation and/or use.

The complexity management tools (views) as discussed in Section 5.4, as well as the other types of modeling artifacts we discuss in Section 5.5, result in different types of artifacts that afford different modeling experiences leading to *Value in Action*.

5.3 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 (ISO/IEC JTC 1/SC 32 Technical Committee on Data management and interchange 1987), 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, of the *universe of discourse* (while avoiding the inclusion of 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'* 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 Proper and Guizzardi (2020), Proper (2021) and Guarino et al. (2020), 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 *domain conceptualization*.

The *domain conceptualization* identifies the fundamental concepts in terms of which the *collective agent* create(s) their conception of the world. This mapping specifies the real-world semantics of that model and characterizes the *ontological commitment* (Guizzardi 2007) of the model (and the *collective agent*) as well as its real-world semantics.

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 *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, Guarino et al. (2020) already stated that the history of conceptual modeling can be traced back to at least the 60s (Quillian 1968). Furthermore, ontology engineering (Guizzardi 2007) also involves the construction of conceptual models representing a (domain) 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

7 We thank Tiago Sales and Isadora Valle Sousa for calling this to our attention. We shall explore these notions together in a future paper.

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, problem-solving, and meaning negotiation (Guarino et al. 2020), 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 (Guizzardi et al. 2002; Guizzardi 2013).

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* (the 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 Proper and Guizzardi (2020), 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 Proper (2021), 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 (Razavi et al. 2012) 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' (Razavi et al. 2012). Likewise, one can speak of being as-true-as-possible-to a given domain as a sort of *conceptual fidelity*. Conceptual fidelity, also frequently called *domain appropriateness*, represents the level of homomorphism between a given representation and the underlying domain conceptualization it commits to. 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) but it also only allows for interpretations that represent state of affairs deemed acceptable by that conceptualization (Guizzardi et al. 2005; Guizzardi 2013). 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* (Razavi et al. 2012) 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, 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 (ISO/IEC JTC 1/SC 32 Technical Committee on Data management and interchange 1987).

It is important to note that we do not argue that non-conceptual models are a bad thing; far from it. 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 (Guizzardi 2010). 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 at least as high as the sum of the RoME of each of the non-conceptual models that have been derived from it.

5.4 Complexity Management and Views

As discussed in e. g., Arbab et al. (2007), Lankhorst et al. (2017) and Frank (2002), 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 hand). In this vein, IEEE (2000) 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, Lankhorst et al. (2017) 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 of the same level of conceptual fidelity,
> - and providing an integrated subset of the *potential* information as provided by the original domain model.

which we hold as being a generalization of the way(s) the notion of view is used in IEEE (2000), Lankhorst et al. (2017) and Frank (2002) where, for instance, the *from the perspective of a related set of concerns* (IEEE 2000) corresponds to the need of a view to be an *integrated* subset of the information as provided by the original domain model. Note that, as 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 provided* does depend on the observer of the model.[8] This is also the reason why we have added the *'in potential'* qualification. For instance, a

8 This issue 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 reason about this has been reported in Proper and Bruza (1999) and Bommel et al. (2007). 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* (Romanenko et al. 2022b).

large 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 the syntax and (formal) semantics of the modeling language(s) used. The requirement that the (potential) information provided by a view should be *integrated*, then corresponds to the requirement of the graph representing the view being a connected graph.

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 in ArchiMate (Lankhorst et al. 2017) that, depending on the purpose at hand, may or may not be included in the model/view. As an example, if a business role *r* is assigned to (the execution of) business process *p*, while business process *p* is (part of) the realization of business service *s*, then this implies that business role *r* is also (part of) the realization of business service *s*.

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 from the (selected!) part 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 in 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., Sowa and Zachman 1992; Spewak 1993; Boar 1999; The Open Group 2011; van't Wout et al. 2010; DoD Deputy Chief Information Officer 2011; Frank 2014; Band et al. 2016). 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.

   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.

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.).

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.

   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 also find the variety of representations that are tuned to different stakeholders in terms of e. g., heat-maps, matrix-like representations, textual descriptions combined with info-graphics, animations, etc. In Section 5.5, 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'*.

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* (Figueiredo et al. 2018), models are re-organized by grouping elements in terms of higher-granularity modeling primitives. In *model modularization* (Guizzardi et al. 2021), 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[9] underlying actual modeling languages and methods (e. g., OMG 2003; Band

9 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*

et al. 2016), as well as explicit *'content frameworks'* (e. g., van't Wout et al. 2010; The Open Group 2011) 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 an attribute 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.

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 (Hofstede et al. 1992; Campbell et al. 1996; Creasy and Proper 1996), as well as more recently utilizing foundational ontologies to generate ontologically founded (un)foldings (Guizzardi et al. 2019; Romanenko et al. 2022a) 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 but capturing 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 discussion, e. g., in Boiten et al. 2000). When, across an engineering framework, the *'cell-specific'* meta-models are aligned well – as is

of the language, or simply, the *ontology of the language*, i. e., a description of the worldview embedded in the language's modeling primitives. For example, Peter Chen's Entity Relationship model commits to a worldview that accounts for the existence of four types of things: entity, relationship, attribute and attribute value spaces (Guizzardi 2007).

explicitly the case for, e. g., ArchiMate (Band et al. 2016), IAF (van't Wout et al. 2010) and MEMO (Frank 2014), views corresponding to the different *'cells'* of the framework can be connected to maintain/obtain the *'larger picture'*.

5.5 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.

A model will be expressed in terms of some modeling language. This can be a precisely (a-priori) defined modeling language, but can also be a highly informal ad-hoc (emerging) language. In a (modeling) language, 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 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, for example, 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 Section 5.4.

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, for example, a BPMN model.[10]

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 richness 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 (Guizzardi and Proper 2021) towards a taxonomy of modeling-related goals, a specification has a world-to-model direction of fit (in a

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

world-to-word vein, cf. Searle, Willis et al. 1983) 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 program 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 purpuse 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.[11]

- *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 5.4 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 produces (Guizzardi 2013; Guizzardi et al. 2002). However, in order to attain these properties, these notations have to be properly designed (Moody 2009) lest denting problem-solving and producing unintended cognitive inferences (*implicatures*) (Guizzardi et al. 2002; Guizzardi 2013).

- *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.

11 For this discussion in the context of ontology engineering, see, e. g., Guizzardi (2007); for the trade-off between expressibity and tractability in knowledge representation languages, see, e. g., Levesque and Brachman (1987). Another manifestation of this problem is the well-known impedance mismatch problem in mapping ontologically-rich conceptual models to relational databases, see, e. g., Guidoni et al. (2022).

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'.

5.6 Conclusion

In this paper, we zoomed in on notions such as views, diagrams, programs, animations, specifications, that play an important role in the *modeling practices* that take place in the context of system development (including software engineering, information systems engineering, and enterprise design management). In doing so, we took the view that these notions are be seen as specific kinds of models, albeit for fundamentally different purposes.

In future work, we 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 Wang et al. (2014). Finally, we intend to investigate the role of different types of *assumptions* (Wang et al. 2016) to modeling practices and to models as artifacts.

References

Apostel, L. (1960). 'Towards the Formal Study of Models in the Non-Formal Sciences'. In: *Synthese* 12, pp. 125–161.

Arbab, F., Boer, F. S. de, Bonsangue, M., Lankhorst, M. M., Proper, H. A. and Torre, L. van der (2007). 'Integrating Architectural Models – Symbolic, Semantic and Subjective Models in Enterprise Architecture'. In: *Enterprise Modelling and Information Systems Architectures* 2.1, pp. 40–57. DOI: 10.18417/emisa.2.1.4.

Azevedo, C. L. B., Iacob, M.-E., Almeida, J. P. A., Sinderen, M. v., Pires, L. F. and Guizzardi, G. (2015). 'Modeling resources and capabilities in enterprise architecture: A well-founded ontology-based proposal for ArchiMate'. In: *Information Systems* 54, pp. 235–262.

Band, I., Ellefsen, T., Estrem, B., Iacob, M.-E., Jonkers, H., Lankhorst, M. M., Nilsen, D., Proper, H. A., Quartel, D. A. C. and Thorn, S. (2016). *ArchiMate 3.0 Specification*. The Open Group.

Bjeković, M., Proper, H. A. and Sottet, J.-S. (2014). 'Embracing Pragmatics'. In: *Conceptual Modeling – 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings*. Ed. by E. S. K. Yu, G. Dobbie, M. Jarke and S. Purao. Vol. 8824. Lecture Notes in Computer Science. Springer, pp. 431–444. DOI: 10.1007/978-3-319-12206-9_37.

Boar, B. H. (1999). *Constructing Blueprints for Enterprise IT architectures*. New York, NY: John Wiley & Sons.

Boiten, E., Bowman, H., Derrick, J., Linington, P. and Steen, M. (2000). 'Viewpoint consistency in ODP'. In: *Computer Networks* 34.3, pp. 503–537.

Bommel, P. van, Proper, H. A. and Weide, T. P. van der (Nov. 2007). 'Information coverage in advisory brokers'. In: *International journal of intelligent systems* 22.11, pp. 1155–1188. DOI: 10.1002/int.20240.

Campbell, L. J., Halpin, T. A. and Proper, H. A. (1996). 'Conceptual Schemas with Abstractions: Making Flat Conceptual Schemas More Comprehensible'. In: *Data & Knowledge Engineering* 20.1, pp. 39–85. DOI: 10.1016/0169-023X(96)00005-5.

Creasy, P. N. and Proper, H. A. (1996). 'A Generic Model for 3-Dimensional Conceptual Modelling'. In: *Data & Knowledge Engineering* 20.2, pp. 119–161. DOI: 10.1016/0169-023X(95)00043-R.

DoD Deputy Chief Information Officer (2011). *The DoDAF Architecture Framework Version 2.02*. URL: https://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf (visited on 15/07/2016).

Figueiredo, G., Duchardt, A., Hedblom, M. M. and Guizzardi, G. (2018). 'Breaking into pieces: An ontological approach to conceptual model complexity management'. In: *2018 12th International Conference on Research Challenges in Information Science (RCIS)*. IEEE, pp. 1–10.

Frank, U. (2014). 'Multi-perspective Enterprise Modeling: Foundational Concepts, Prospects and Future Research Challenges'. In: *Software & Systems Modeling* 13.3, pp. 941–962.

Frank, U. (2002). 'Multi-perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages'. In: *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 3*. Los Alamitos, CA: IEEE Computer Society Press.

Guarino, N., Guizzardi, G. and Mylopoulos, J. (2020). 'On the Philosophical Foundations of Conceptual Models'. In: *Information Modelling and Knowledge Bases XXXI* 321, p. 1.

Guidoni, G. L., Almeida, J. P. A. and Guizzardi, G. (2022). 'Preserving conceptual model semantics in the forward engineering of relational schemas'. In: *Frontiers in Computer Science* 4. DOI: 10.3389/FCOMP.2022.1020168.

Guizzardi, G. (2010). 'Theoretical foundations and engineering tools for building ontologies as reference conceptual models'. In: *Semantic Web* 1.1–2, pp. 3–10.

Guizzardi, G. (2013). 'Ontology-based evaluation and design of visual conceptual modeling languages'. In: *Domain engineering, Product Lines, Languages, and Conceptual Models*. Ed. by I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen and J. Bettin. Springer, pp. 317–347.

Guizzardi, G., Ferreira Pires, L. and Sinderen, M. v. (2005). 'An ontology-based approach for evaluating the domain appropriateness and comprehensibility appropriateness of modeling languages'. In: *International Conference on Model Driven Engineering Languages and Systems*. Springer, pp. 691–705.

Guizzardi, G. and Proper, H. A. (2021). 'On Understanding the Value of Domain Modeling'. In: *Proceedings of 15th International Workshop on Value Modelling and Business Ontologies (VMBO 2021), Bolzano, Italy, 2021*. Ed. by G. Guizzardi, T. P. Sales, C. Griffo and M. Furnagalli. Vol. 2835. CEUR Workshop Proceedings. CEUR-WS.org.

Guizzardi, G. (2007). 'On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta) Models'. In: *Frontiers of Artificial Intelligence and Applications*. Vol. 155. IOS Press, p. 18.

Guizzardi, G., Figueiredo, G., Hedblom, M. M. and Poels, G. (2019). 'Ontology-Based Model Abstraction'. In: *13th International Conference on Research Challenges in Information Science, RCIS 2019, Brussels, Belgium, May 29–31, 2019*. Ed. by M. Kolp, J. Vanderdonckt, M. Snoeck and Y. Wautelet. IEEE, pp. 1–13. DOI: 10.1109/RCIS.2019.8876971.

Guizzardi, G., Pires, L. F. and Van Sinderen, M. J. (2002). 'On the role of domain ontologies in the design of domain-specific visual modeling languages'. In: *Proc. 2nd Workshop on Domain-Specific Visual Languages*. ACM Press New York.

Guizzardi, G., Sales, T. P., Almeida, J. P. A. and Poels, G. (2021). 'Automated conceptual model clustering: a relator-centric approach'. In: *Software and Systems Modeling*, pp. 1–25.

Harel, D. and Rumpe, B. (2004). 'Meaningful Modeling: What's the Semantics of "Semantics"?' In: *IEEE Computer* 37.10, pp. 64–72. DOI: 10.1109/MC.2004.172.

Hofstede, A. H. M. ter, Proper, H. A. and Weide, T. P. van der (May 1992). 'Data Modelling in Complex Application Domains'. In: *Advanced Information Systems Engineering, CAiSE'92, Manchester, UK, May 12–15, 1992, Proceedings*. Ed. by P. Loucopoulos. Vol. 593. Lecture Notes in Computer Science. Springer, pp. 364–377. DOI: 10.1007/BFb0035142.

Hoppenbrouwers, S. J. B. A., Proper, H. A. and Weide, T. P. van der (June 2005). 'A Fundamental View on the Process of Conceptual Modeling'. In: *Conceptual Modeling – ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings*. Ed. by L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos and O. Pastor. Vol. 3716. Lecture Notes in Computer Science. Springer, pp. 128–143. DOI: 10.1007/11568322_9.

IEEE (Sept. 2000). *Recommended Practice for Architectural Description of Software Intensive Systems*. Tech. rep. IEEE P1471:2000, ISO/IEC 42010:2007. The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE. Piscataway, New Jersey: IEEE Explore, Los Alamitos, California.

ISO/IEC JTC 1/SC 32 Technical Committee on Data management and interchange (1987). *Information processing systems – Concepts and Terminology for the Conceptual Schema and the Information Base*. Tech. rep. ISO/TR 9007:1987. ISO.

Lankhorst, M. M., Torre, L. van der, Proper, H. A., Arbab, F. and Steen, M. W. A. (2017). 'Viewpoints and Visualisation'. In: *Enterprise Architecture at Work – Modelling, Communication and Analysis*. 4th. The Enterprise Engineering Series. Springer, Heidelberg, Germany, pp. 171–214. ISBN: 978-3-662-53932-3. DOI: 10.1007/978-3-662-53933-0_8.

Levesque, H. J. and Brachman, R. J. (1987). 'Expressiveness and tractability in knowledge representation and reasoning 1'. In: *Computational intelligence* 3.1, pp. 78–93.

Molnar, G. (2003). *Powers: A study in metaphysics*. Clarendon Press.

Moody, D. (2009). 'The "Physics" of notations: toward a scientific basis for constructing visual notations in software engineering'. In: *IEEE Transactions on software engineering* 35.6, pp. 756–779.

Mumford, S. and Anjum, R. L. (2011). *Getting causes from powers*. Oxford University Press.

OMG (2003). *UML 2.0 Superstructure Specification – Final Adopted Specification*. Tech. rep. ptc/03–08–02. Object Management Group, Needham, Massachusetts.

Op 't Land, M., Proper, H. A., Waage, M., Cloo, J. and Steghuis, C. (2008). 'The Results of Enterprise Architecting'. In: *Enterprise Architecture – Creating Value by Informed Governance*. The Enterprise Engineering Series. Springer, Heidelberg, Germany. Chap. 4. ISBN: 978-3-540-85231-5. DOI: 10.1007/978-3-540-85232-2.

Proper, H. A. (2005). *TEE Group – Focus & Drives – Return on modelling effort*. URL: http://www.cs.ru.nl/tee/focus-drives.htm (visited on 20/01/2005).

Proper, H. A. (2009). *Models that matter; Return on Modelling Effort*. Blog. URL: http://erikproper.blogspot.com/2009/02/models-that-matter-return-on-modelling.html (visited on 04/01/2021).

Proper, H. A. (2021). 'On Model-Based Coordination of Change in Organizations'. In: *Engineering the Transformation of the Enterprise: A Design Science Research Perspective*. Ed. by S. Aier, P. Rohner and J. Schelp. Springer, Heidelberg, Germany, pp. 79–98. ISBN: 978-3-030-84655-8. DOI: 10.1007/978-3-030-84655-8_6.

Proper, H. A. and Bruza, P. D. (1999). 'What Is Information Discovery About?' In: *Journal of the American Society for Information Science* 50.9, pp. 737–750.

Proper, H. A. and Guizzardi, G. (2020). 'On Domain Modelling and Requisite Variety – Current state of an ongoing journey'. In: *The Practice of Enterprise Modeling, PoEM 2020*. Ed. by J. Grabis and D. Bork. Vol. 400. Lecture Notes in Business Information Processing. Springer, pp. 186–196. DOI: 10.1007/978-3-030-63479-7_13.

Proper, H. A. and Guizzardi, G. (2021). 'On Domain Conceptualization'. In: *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*. Ed. by D. Aveiro, G. Guizzardi, R. Pergl and H. A. Proper. Vol. 411. Lecture Notes in Business Information Processing. Heidelberg: Springer, pp. 49–69. DOI: 10.1007/978-3-030-74196-9_4.

Proper, H. A. and Guizzardi, G. (2022). 'Modeling for Enterprises; Let's go to RoME ViA RiME'. In: *PoEM 2022 Forum Proceedings*. CEUR-WS.org.

Proper, H. A., Verrijn–Stuart, A. A. and Hoppenbrouwers, S. J. B. A. (2005). 'On Utility-based Selection of Architecture-Modelling Concepts'. In: *Conceptual Modelling 2005, Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, NSW, Australia, January/February 2005*. Ed. by S. Hartmann and M. Stumptner. Vol. 43. Conferences in Research and Practice in Information Technology Series. Sydney, New South Wales, Australia: Australian Computer Society, pp. 25–34.

Quillian, M. R. (1968). 'Semantic memory, Semantic Information Processing'. PhD thesis. Massachusetts: MIT.

Razavi, S., Tolson, B. A. and Burn, D. H. (2012). 'Review of surrogate modeling in water resources'. In: *Water Resources Research* 48.7. DOI: 10.1029/2011WR011527.

Romanenko, E., Calvanese, D. and Guizzardi, G. (2022a). '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*. Springer, pp. 372–388.

Romanenko, E., Calvanese, D. and Guizzardi, G. (2022b). 'Towards Pragmatic Explanations for Domain Ontologies'. In: *Knowledge Engineering and Knowledge Management: 23rd International Conference, EKAW 2022, Bolzano, Italy, September 26–29, 2022, Proceedings*. Springer, pp. 201–208.

Rothenberg, J. (1989). 'The Nature of Modeling'. In: *Artificial Intelligence, Simulation & Modeling*. Ed. by L. E. Widman, K. A. Loparo and N. Nielson. New York, NY: John Wiley & Sons, pp. 75–92.

Sales, T. P., Guarino, N., Guizzardi, G. and Mylopoulos, J. (2017). 'An Ontological Analysis of Value Propositions'. In: *2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*, pp. 184–193. DOI: 10.1109/EDOC.2017.32.

Sandkuhl, K., Fill, H.-G., Hoppenbrouwers, S. J. B. A., Krogstie, J., Matthes, F., Opdahl, A. L., Schwabe, G., Uludag, Ö. and Winter, R. (2018). 'From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling'. In: *Business & Information Systems Engineering* 60.1, pp. 69–80.

Searle, J. R., Willis, S. et al. (1983). *Intentionality: An essay in the philosophy of mind*. Cambridge University Press.

Sowa, J. and Zachman, J. A. (1992). 'Extending and formalizing the framework for information systems architecture'. In: *IBM Systems Journal* 31.3, pp. 590–616.

Spewak, S. H. (1993). *Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology*. New York, NY: John Wiley & Sons.

Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Heidelberg: Springer.

Thalheim, B. (2011). 'The Theory of Conceptual Models, the Theory of Conceptual Modelling and Foundations of Conceptual Modelling'. In: *Handbook of Conceptual Modeling*. Springer, Heidelberg, Germany, pp. 543–577.

The Open Group (2011). *TOGAF Version 9.1*. 10th ed. Zaltbommel, NL: Van Haren Publishing.

van't Wout, J., Waage, M., Hartman, H., Stahlecker, M. and Hofman, A. (2010). *The Integrated Architecture Framework Explained*. Heidelberg, Germany: Springer.

Wang, X., Guarino, N., Guizzardi, G. and Mylopoulos, J. (2014). 'Towards an ontology of software: a requirements engineering perspective'. In: *Formal Ontology in Information Systems*. IOS Press, pp. 317–329.

Wang, X., Mylopoulos, J., Guizzardi, G. and Guarino, N. (2016). 'How software changes the world: The role of assumptions'. In: *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, pp. 1–12.

Zarwin, Z., Bjeković, M., Favre, J.-M., Sottet, J.-S. and Proper, H. A. (July 2014). 'Natural Modelling'. In: *Journal Of Object Technology* 13.3, 4: 1–36. DOI: 10.5381/jot.2014.13.3.a4.