Domain Modelling and the Law of Requisite Variety

Current state of an ongoing journey

Hend*erik* A. Proper[1,2][0000−0002−7318−2496] and Giancarlo Guizzardi[3]

[1] Luxembourg Institute of Science and Technology (LIST), Belval, Luxembourg
[2] University of Luxembourg, Luxembourg
[3] Free University of Bolzano-Bozen, Italy
e.proper@acm.org, giancarlo.guizzardi@unibz.it

Abstract. In the 1950's, W. Ross Ashby introduced (a.o.) the Law of Requisite variety in the context of General Systems Theory. The key tenet of this Law is that only "*variety can destroy variety*". Towards purposely created artefacts (such as domain models, including enterprise models), one of the operational consequences of this Law is the fact that the variety which a designed artefact needs to "meet", needs to be reflected in the variety of the artefact itself. In this paper we explore some of the consequences of the Law of Requisite Variety for models, modelling languages, and the act of modelling. To this end, we start with a review of our current understanding of domain modelling (including enterprise and conceptual modelling), and the role of modelling languages. We then briefly discuss the Law of Requisite variety, as introduced by Ashby. This is then applied to our understanding of models, the act of modelling and the role of modelling languages. This will a.o. lead to insights into important trade-offs in dealing with (domain) genericity / specificity of modelling languages.

1 Introduction

In the context of software engineering, information systems engineering, business process management, and enterprise engineering & architecting in general, many different kinds of models are used. This includes a.o.: enterprise (architecture) models, business process models, ontology models, organisational models, information models, software models, etc. In this paper, we consider each of these kinds of models as being valued members of a larger family of *domain models*.

Domain models have come to play an important role during all stages of the life-cycle of enterprises and their information and software systems. This includes their development, improvement, maintenance, operation, as well as regulation. The models involved, carry (potentially valuable) organisational knowledge; putting even more stress on the role of domain modelling. In line with this, it is interesting to observe that, for their own institutional information systems, the European Union also relies heavily on a model-based approach, even resulting in the creation of a dedicated *competence centre for modelling*.[4]

In our view, the key role of domain models across the life-cycle of enterprises, and their information and software systems, fuels the need for more fundamental reflection

[4] https://ec.europa.eu/jrc/sites/jrcsh/files/ccmod_leaflet.pdf

on domain modelling itself. This includes *the act of modelling*, *the essence of what a model is*, and *the role of (modelling) languages*.

Such fundamental topics have certainly been studied by different scholars (see e.g. [1, 52, 47, 33, 18, 29, 38, 36, 53, 48]), as well as by ourselves (see e.g. [31, 46, 21, 26, 22, 9, 23, 8, 24, 25, 15, 61, 7, 35, 44, 20]). At the same time many challenges remain. Some of these challenges have been discussed in e.g. [46, 23, 25, 44].

The amount of research effort that has been put into such fundamental topics, seems limited in comparison to the quantity of research conducted in specific "applied" fields of modelling, such as software modelling, information modelling, enterprise (architecture) modelling, (business) rules modelling, and business process modelling. Even though we do not argue against the importance of research conducted in these "applied" fields of domain modelling, we do argue that there is a need to find answers to some of the more fundamental challenges that will lead to generic insights, and results, that can be applied across the more specific areas of modelling.

In this paper, we focus on the challenge of how domain modelling needs to deal with different forms of variety (and complexity). We will, as the title of the paper suggests, do so from the perspective of the *Law of Requisite Variety*. This Law has been postulated by W. Ross Ashby [2], in the context of General Systems Theory and Cybernetics in particular, and also building on Shannon's Information Theory [50]. In this context, *variety* refers to the number of states of a system (system in the most general sense).

The key tenet of the *Law of Requisite Variety* is that only "*variety can destroy variety*". Towards purposely created artefacts (such as domain models, including enterprise models), one of the operational consequences of this Law is the fact that the variety which a designed artefact needs to "meet" should to be reflected in the variety of this artefact. In line with this, the aim of this paper is to explore the consequences of the Law of Requisite Variety for models, modelling languages, and the act of modelling.

We see this paper as part of an ongoing "journey" we undertake, with the aim of deepening our insights into the foundations of domain modelling, mixing our theoretical work and practical experiences in developing (foundational and core) ontologies and domain models, associated modelling languages, frameworks, and methods.

The remainder of this paper is structured as follows. In section 2, we start with a review of our current understanding of domain modelling, while also positioning enterprise models and conceptual models in relation to this. Section 3 then complements this with our view on the modelling languages. We then continue, in section 4, by reviewing the Law of Requisite variety as introduced by Ashby [2]. In section 5 we explore (some of) the consequences of this Law on domain modelling. The latter will a.o. lead to insights into important trade-offs in dealing with (domain) genericity / specificity of modelling languages.

2 Domain models

Based on foundational work by e.g. Apostel [1], and Stachowiak [52] on the notion of *model* and their semiotic roots [39], more recent work on the same by different authors [47, 29, 53, 48], as well as our own work [31, 46, 20, 22, 7], we consider a *domain model* to be:

> An *artefact* that is *acknowledged* by an *observer* to *represent* an *abstraction* of some *domain* for a particular *purpose*.

Each of the stressed words in this definition requires a further explanation, and as we will see in section 5 results in further nuances when considering the relevant *variety* involved.

A model is seen as an *artefact*. In other words, it is something that exists outside of our minds. In "our" fields of application, this artefact typically takes the *form* of some "boxes-and-lines" diagram. These diagrams, expressed (again, typically) in some form of concrete visual syntax, can have its grammar specified by a set of rules (e.g., a meta-model, a graph grammar [60]), and its semantics defined by a mapping to a mathematical structural (*formal semantics* [30]) or to an ontological theory (*ontological or real-world semantics* [15]). More generally, domain models can, depending on the *purpose* at hand, take other forms as well, including text, mathematical specifications, physical objects, etc.

With *domain*, we refer to "anything" that one can speak / reflect about explicitly. It could be "something" that already exists in the "real world", something desired towards the future, or something imagined. The *observer* observes the domain by way of their senses and / or by way of (self) reflection. What results in the mind of the observer is, what is termed a *conceptualisation* in [22], and a *conception* in [16].

When the *domain* to be modelled pertains to a part / perspective / aspect of an enterprise, then we can indeed refer to the resulting domain model as an *enterprise model*.

As, it is ultimately the observer who needs to *acknowledge* the fact that the *artefact* is indeed a model of the domain, it actually makes sense to treat *their* conceptualisation / conception of the domain as the de-facto "proxy" for the domain. As such, we should also realise that the *observer* observes the model (as artefact) as well, which therefore also creates a conceptualisation (in their mind) of the model. The observer, therefore, needs to validate the alignment between their model-conceptualisation and their domain-conceptualisation, where the *purpose* of the model determines the alignment criteria.

Models are produced for a *purpose*, also in relation to an expected *Return on Modelling Effort* (RoME) [42, chapter 4]. In the context of enterprise modelling, [43] suggest (at least) seven high-level purposes for the creation of enterprise models: *understand*, *assess*, *diagnose*, *design*, *realise*, *operate* and *regulate*. In specific situations, these high-level purposes will need to be made more specific in terms of, e.g., the need for different stakeholders to *understand*, *agree*, or *commit* to the content of the model [45], or for a computer to be able to intepret the model in order to e.g. automatically analyse the model, use the model as the base of a simulation / animation, or even execute the model, etc.

A model is the representation of an *abstraction* of the domain [31, 22, 24, 7]. This implies that, in line with the *purpose* of the model, some "details" of the domain are consciously filtered out. As a corollary to this definition, it implies that an observer (when acknowledging that that some artefact is indeed a model of the domain), must also be able to identify details in the domain that are *not* represented in the model.

In the context of domain modelling, four important flavours of abstraction are [4]: (1) *selection*, where we decide to only consider certain elements and / or aspects of the domain; (2) *classification*; (3) *generalisation*; and (4) *aggregation*. In our field of application, selection typically leads to frameworks of aspects / layers by which to model an enterprise, but also to mechanisms for view extraction (see discussion below), as well as clustering and model summarisation [17, 27]. Classification, typically leads to some class-instance and / or type-instance relationships, including type-instance relationships between types and higher-order types, i.e., multi-level structures [10]; generalisation leads to the formation of specialisation / generalisation *taxonomies*, in which sub-types specialise properties of super-types; aggregation leads to the formation of *partonomies* of various kinds in which entities, seen as integral wholes, can be decomposed into parts. Parts, on the other hand, hang together bound by some unity criterion that forms the whole [26].

As a consequence of the above, an observer actually needs to harbour (at least) four conceptualisations: (1) a "full" conceptualisation of the domain (as they "see" it), (2) a conceptualisation of the purpose for the model, (3) an abstracted conceptualisation of the domain, (4) a conceptualisation of the artefact that is (to be) the model representing the latter abstraction.

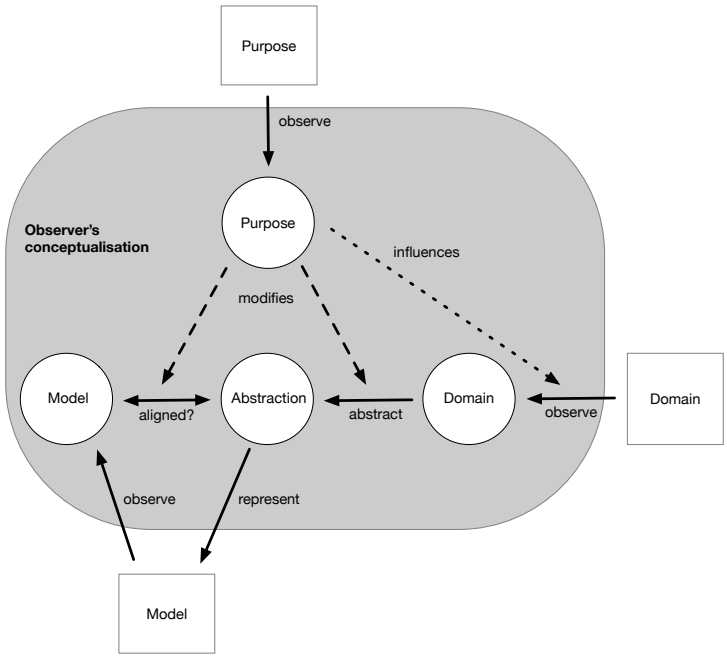

Fig. 1. Conceptualisations involved in domain modelling

The latter has been illustrated in figure 1, where we see the four conceptualisations in the middle, where the conceptualisation of the purpose modifies the abstraction and

the question of alignment between the conceptualisations of the model and the desired abstraction. The purpose may actually already influence the original observation of the domain.

When the model-conceptualisation corresponds to the abstraction-conceptualisation, then the observer would agree that the artefact is a model of the domain for the given purpose. As a consequence, different models (as artefacts) may indeed result in the same model-conception, in which case (for the observer) they are equivalent models of the same domain (for the same purpose).

If the observer is "the modeller", i.e. the person creating the model, they also need to "shape" the model in such a way that it best matches their *desired* model-conceptualisation.

So-far, we used the *observer* in the singular sense. In practice, modelling obviously involves multiple observers. Both during the process of creating a model, but in particular also during the usage of the model. Each of the observers involved in the life-cycle of a model will have their own domain-conceptualisation, purpose-conceptualisation, abstraction-conceptualisation, and model-conceptualisation, which also need mutual alignment among the observers. This is, indeed, a major challenge in collaborative modelling [44].

A notion related to model is the notion of a *view*, which is heavily used in the context of enterprise architecture [32], but actually dates back from well before enterprise architecture became popular [57]. We would argue that a view, by the nature of its name, provides a view on a larger / more detailed model, analogous to the way in which a database view provides a selection and / or aggregation on an underlying database. As such a view involves a further abstraction (in terms of e.g. selection, classification, generalisation, or aggregation) of the domain [17, 27].

In line with the above discussion, a domain model should be (the representation of) the abstraction of (the conceptualisation of) a domain. At the same time, for different purposes, such as the ability to use the model as a base for simulation, computer-based reasoning, animation, execution, or database design, it may be necessary to make "compromises" to the model. These compromises result in a model that does *not* correspond to (an abstraction of) the original domain. They are essentially models of a "close enough" approximation of (the conceptualisation of) the original domain.

For instance, OWL-based [37] domain models are likely to involve compromises to enable computational properties. Even more, conceptual database design [28] often introduces compromises such as the *closed world assumption* or *unique name assumption*. These *assumptions* are made for computational reasons, but are of course not assumptions that are generally made in real life. Other examples are the limitations found in many mainstream (but not all!) Object-Oriented programming languages in which an object instantiates a single static direct class and in which classes cannot not specialise multiple supertypes.[5]

This is where we can make a distinction between *conceptual domain models* and *non-conceptual domain models* in the sense that a *conceptual domain model* is:

[5] Technically speaking, because generalisation is transitive, if x instantiates A then x instantiates all supertypes of A. What this rule is meant to represent is that if x instantiates type A and B then these two types must be related by a generalisation path.

> A model of a domain, where the purpose of the model is dominated by the ambition to remain as-true-as-possible to the original domain.

Note the use of the word *ambition*. We are not suggesting there to be a crisp border between conceptual and non-conceptual domain models. However, the word *ambition* also suggest that a modeller / observer, as their insight in a domain increases should be driven to reflect on the conceptual purity of their conceptualisation and of the resulting model.

Non-conceptual domain models certainly have an important role to play. However, it is important to be aware of the compromises one has made to the original domain conceptualisation. As such, it is also possible that one conceptual model has different associated non-conceptual models, each having compromises to meet different purposes.

Since we consider models to be *artefacts*, it is relevant to briefly consider the notion of *identity* of such artefacts. In other words, when are two models the same, or not?

From an ontological point of view, artefacts are historically dependent on their creators [54]. If two people write content-wise identical novels (by total chance), these are still two different books (artifacts). In other words, "authorship" is part of the identity of an artefact. We argue that the same holds for models. In other words, if two models look the same, but have different creators, they are not the same.

We would also argue that the purpose with which an artefact, in particular a model, is created is part of the artefact's identity. So, even when a model or a book "looks" the same, if they are created for a different purpose, then they are different.

Finally, when one author writes both a Dutch and a Portuguese version of the same book, then even if they are the same content-wise, they are two different books. Therefore, the language used is also part of the identity of the artefact. Of course, anyone who is multilingual knows quite well that across languages it is not easy to say if a text is "content-wise" the same, due to different socio-cultural understandings of words and situations. The same applies in the context of domain models. An ER model [11], an ORM model [28], and a UML class diagram [40] might have the same real-world semantics. At the same time, their formal semantics is likely to differ, although with some assumed mappings between the modelling concepts (the underlying ontologies of these languages), one may be able to prove semantic equivalence.

3 The role of modelling languages

In its most general form a language identifies the conventions to expressions in the language should conform to.

In a domain modelling context, these conventions are often equated to a definition in terms of a concrete visual syntax, and a grammar in terms of a set of rules, while the semantics are defined in terms of some mathematical structure (*formal semantics* [30]) or an ontological theory (*ontological or real-world semantics* [15]). However, style guides, reference models, patterns, etc, can also be seen as part of the set of conventions that define a (modelling) language.

Sometimes, a modelling language comes in the form of a number of connected sub-languages. Typical examples are ARIS [49], UML [40] and ArchiMate [3, 34], each

featuring more specific languages focused on one aspect or layer, as well as (some more, some less) the integration / coherence between these aspects or layers.

By way of its grammar / meta-model / ontological underpinning, a modelling language does provide a normative frame [44] which the "user" of the language needs to commit to.

When modelling, a normative frame can be beneficial as it provides guidance for the conceptualisation of a domain, and the abstraction needed to arrive at a model that meets the purpose. However, a normative frame can also start to act as a straitjacket or even lead to tunnel vision [5]. It can also lead practitioners to use UML [40], BPMN [41] or ArchiMate [3, 34] "in name only", in the sense of creating diagrams in a free-format drawing tool, while using the symbols from one of these languages freely beyond the rules / definitions of the language.

The role of modelling languages as a normative frame, has certainly sparked a lot of debate in literature as well. For example, Wyssusek's [58] critique on the Bunge-Wand-Weber ontology [55] providing a normative frame on the linguistic structure of a modelling language, resulted in a lively debate (summarised in [59]).

As discussed in the previous section, we argue that the language used in creating a model is part of the identity of the model. Actually, we would prefer to take this argument a step further. If a model is represented in some (pre-)defined language, then the (relevant part of the) definition of that language should actually be seen as being a *part* of the model. This also allows us to illustrate the role of the modelling language in the sense of providing a trade off. If, given some purpose, there is a need to represent an abstraction A, and one has a choice between using a language L1 with an elaborate set of conventions (the light grey part), or using a language L2 with only a more limited set of conventions (the dark grey part), then this will lead to a difference in the size of the (situation specific parts of the) model one would still need have to specify. This has been illustrated in figure 2, where we show that the "size" of the two models as a whole remains the same. Of course, the notion of "size" is to be clarified. We will return to this in section 5 when discussing the consequences of the Law of Requisite Variety, as the "size" indeed connects directly to the variety of the model.

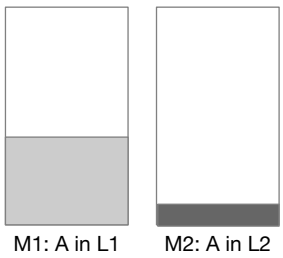

Fig. 2. Trade off between languages with different sizes of their definition

A final consideration is the fact that the conventions which define a modelling language need to reflect the intended set of valid models. Which also means that these

conventions need to accommodate all the possible purposes of these intended models. For standardised general purpose languages, such as UML [40], BPMN [41] or ArchiMate [3, 34], this does lead to tensions, as the set of purposes keeps growing, resulting in a continuous growth of the set of allowable models, thus also triggering a growth in the "size" of the body of conventions defining these languages [8, 5].

As such, the we should also realise that the grey parts in figure 2 amount only to that part of the respective languages that are relevant for the interpretation of the white parts. However, the other parts of the language will also need to be learned by the "users" of the language, or at least be visible as part of the standard.

4 The Law of Requisite Variety

The *Law of Requisite Variety*, as defined by W. Ross Ashby [2, page 2020], while also building on Shannon's Information Theory [50], states that only "*variety can destroy variety*". Ashby's work is one of the defining contributions to the field of General Systems Theory and Cybernetics in particular.

In this context, *variety* essentially refers to the number of states of a system, where we should consider *systems* in the most general sense. The idea of only "*variety can destroy variety*" is that for a system *C* to control / manage a system *R*, system *C* must (at least) match the variety of *R*. Here it is important to clearly understand the scope of the system that would need to be controlled. For example, controlling a car in the sense of getting it into motion and steering it in a certain direction on an empty car park, is quite different from driving a car through busy traffic. The latter *system* clearly needs to deal with a larger variety.

In the context of purposely created artefacts, such as cars, books, software, and domain models, different kinds of variety can be discerned, such as:

1. variety of the functional requirements on the artefact,
2. variety of the functional non-requirements on the artefact,
3. variety of the design of the artefact,
4. variety of the production of the artefact,
5. variety of the artefact itself,
6. variety of the (mis)usage scenarios of the artefact,
7. variety of a specific (mis)usage scenario of the artefact,
8. variety of pragmatic conventions (e.g., layout choices that signal expertise),
9. variety of complexity management mechanisms (e.g., patterns, modules),
10. variety required from a user of the artefact to use it for its intended purposes,
11. ...

The variety of the requirements are in principle linked to the varieties of (mis)usage. However, the former is more *by design* oriented, whereas the latter ones does not really emerge until the artefact is actually *in use*, which may indeed go beyond its intended use.

When considering the artefact *in use*, it is of course important to also identify the perspective of the user. Hence the need to distinguish between the variety of scenarios of use, and the variety of a specific scenario. For instance, a photo book might have multiple (types) of usage scenarios; each with their own variety:

1. From the perspective of someone who cannot read the language in which the book is written in, it is nice to look at the photos.
2. From the perspective of someone interested in art, it is interesting to look at the photos, and read the text explaining the story behind the photo.
3. From the perspective of an aspiring photographer, it is interesting to study the photos, the text, as well as the used lenses, exposure times, and focal length.

Needless to say, that for domain models there are clear analogies to be identified.

Before discussing the consequences of the Law of Requisite variety for domain modelling (in the next section), it is important to link variety to complexity and uncertainty. There seems to be no unified view of what complexity is. Nevertheless, whenever something is regarded as "complex" it implies that

1. there is some *observer*(s) who look(s) at a *domain* (the "something" that is regarded as complex),
2. this *observer* then harbours (in their mind) a *conceptualisation* of the domain,
3. mentally "navigating" (and understanding) this *conceptualisation* involves a large state space, i.e., a large space of possible distinctions,
4. in other words, understanding the *domain* requires a a high level of variety (*requisite variety*) on the side of the *observer*.

As such, we argue that to observe something that is (to be called) complex implies a *requisite variety* on the side of the observer. At a more abstract level, the system "*the observer who observes and tries to understand a domain*" needs a variety that matches the variety of the observed domain. Which of course needs to "fit" within the cognitive abilities of the observer. If the variety needed to understand the structures of the domain is experienced as "high" then the observer would call that domain "complex".

We can take this a step further in the sense that the understanding of *something* may also lead to the insight that there are several uncertainties / unknowns. Which, in terms of the observer's conceptualisation of the *something*, leads to a more nuanced / refined conceptualisation involving the known unknowns, and potential unknown unknowns. In other words, the observer harbours a conceptualisation with a number of identified question marks, i.e. more variety. This, then lead to a further increase in the requisite variety that is asked from the observer.

Here it is interesting to mention the Cynefin [51] framework, which makes a distinction between simple, complicated, complex, chaotic, and disorderly problems. Underlying these "levels", there is an increase in the complexity and uncertainties about the problem domain (about which one is to take decisions). We would argue that in terms of variety, the requisite variety expected from the decision maker(s) increases from the simple to the disorderly problems.

5 Consequences for domain modelling

In this section we discuss (some of) the consequences of the Law of Requisite variety on domain modelling. In doing so, we will visit the varieties of (1) the domain to be captured in the model, (2) the purpose for the model, (3) the desired abstraction, (4) the model itself, (5) the modelling language landscape, (6) the actors involved in modelling.

Variety of the domain – The domain that is to be modelled has an inherent variety. This variety may be due to the complexity of the domain, or unknown elements in / about the domain.

The first challenge with this variety is to ensure that this variety is acknowledged, and is then translated to a purpose for modelling, resulting in models capturing the variety. In this regards, it is important to refer back to our earlier discussion in section 4, regarding the Cynefin [51] framework, making a distinction between simple, complicated, complex, chaotic, and disorderly problems. This framework does underline the need to acknowledge the variety (in terms of complexities and uncertainties) of the domain being modelled.

Variety of the model purpose – The purpose of the model leads to three kinds of more specific kinds of variety.

The first kind of variety concerns the information that the model needs to be able to provide about the domain. In other words, the informational payload (as also mentioned by e.g. [38]) that the model is expected to have. This variety drives the requisite variety expected from the model in relation to the domain.

The second kind of variety pertains to the social complexity of the context in which the model is created. This is where, based on experiences from the IBIS project [12], Conklin coins the term social-complexity [13, 14]. This social-complexity involves both the people involved in the creation of a specific model (i.e. collaborative modelling), as well as the stakeholders involved in a software / information systems / enterprise engineering effort.

The third kind of purpose related variety is concerned with the potential of *multiple uses* of a model. In other words, a model, once created, might be used for multiple more specific purposes. On the one hand, this variety should lead to some measures to ensure that a model can indeed be used for these more specific purposes. At the same time, it also illustrates the need to clearly include the purpose of the model with the model itself, as a kind of disclaimer / instructions for use and interpretation.

Variety of the abstraction – Driven by the model purpose, there is a need to capture a relevant (but not trivialised) part of the variety of the domain to be modelled. This means that the purpose of the model, and the inherent variety of the domain, result in a requisite variety that *needs* to be covered by the abstraction (and indeed, the representation in terms of the model as artefact).

From a modelling-task point of view, this is actually the most challenging form of variety. In other words, the actual "art" of modelling lies here; finding the right abstraction needed for the purpose at hand. This also requires a balance between the simplification / abstraction needed for the model's purpose and target audience, and the inherent complexity / variety of the domain.

Variety of the model – In the context of models, we need to consider three kinds of more specific variety.

Firstly, the variety of the model links directly to the "informational payload" of the model as required by its purpose. This information payload has a direct impact on the variety required from the model, thus also resulting in a minimal "complexity" of the domain itself.

Here it it interesting to refer back to older discussions (in the field of information systenms engineering) regarding the complexity of modelling notations. As also mentioned by Moody [38], models need to provide some informational payload. A simpler notation might be easier to learn, and easier to look at when used in models, but when it cannot "carry" the needed informational payload, then the more "simpler" notation may actually turn out to be a liability.

Furthermore, the variety of the model takes us back to the earlier discussion about the fact that two different models may actually correspond to the same model-conceptualisation for an observer. As such, there is a variety of variations of the actual representation as an artefact (of the same model-conceptualisation / abstraction-conceptualisation).

Variety of the modelling language landscape – This variety involves the variety of the modelling language itself, as well as the entire landscape of interlinked modelling languages involved in a software / information systems / enterprise engineering context.

The discussion related to figure 2 already pointed at the need to essentially include the (relevant parts of the) definition of the modelling language in a model. The whole of the specified model (the white parts in figure 2) and the parts provided by the language (the gray parts in figure 2), needs to match the variety of the abstraction-conceptualisation. The border between these, however, can be determined.

Of course, when using a "thin" language with a small set of conventions, it may be easy to learn the language, and also easy to create modelling tools that support the language. At the same time, specifying the actual models (the white parts in figure 2) will require more effort than it would cost when using a language with more pre-defined concepts and conventions. Provided of course, that these (pre-defined concepts and conventions) indeed meet the modelling purpose, and domain, at hand.

Beyond a single language, a software / information systems / enterprise engineering effort will involve a mix of modelling languages, and supporting tools. This resulting landscape is a "domain modelling system" with its own variety. A variety that also needs to be acknowledged and managed [6].

Variety to be met by those involved in modelling – This refers to the requisite variety needed from the actors (human and / or IT-based) involved modelling.

We already pointed at the fact that the system: "*the observer who observes and tries to understand a domain*" needs to meet the variety of the domain, effectively requiting the observer having the appropriate cognitive abilities.

So, far, not much work has been done on the required cognitive abilities of the actors involved in modelling. Some work which we are aware of includes [19, 56].

6 Conclusion

In this paper we explored some of the consequences of the Law of Requisite Variety for domain models, tha associated modelling languages, and the act of modelling.

We first provided a review of our current understanding of domain modelling (including enterprise and conceptual modelling), and the role of modelling languages. In doing so, we already pointed at some of the considerations that have a Law of Requisite Variety related underpinning.

Using this as a base, we then explored some of the possible consequences / challenges of the Law of Requisite Variety in a domain modelling context.

As mentioned in the introduction, we see paper as part of an ongoing "journey", with the aim of deepening our insights into the foundations of domain modelling. In line with this, we certainly do not claim this paper to be a fully finished work. It provides a snapshot of our current understanding, and we hope that debates with our colleagues will aid us in continuing our journey.

References

1. Apostel, L.: Towards the Formal Study of Models in the Non-Formal Sciences. Synthese **12**, 125–161 (1960)
2. Ashby, W.R.: An Introduction to Cybernetics. Chapman & Hall, London, United Kingdom (1956)
3. Band, I., Ellefsen, T., Estrem, B., Iacob, M.E., Jonkers, H., Lankhorst, M.M., Nilsen, D., Proper, H.A., Quartel, D.A.C., Thorn, S.: ArchiMate 3.0 Specification. The Open Group (2016)
4. Batini, C., Mylopoulos, J.: Abstraction in conceptual models, maps and graphs. In: Tutorial presented at the 37th Intl. Conf. on Conceptual Modeling, ER 2018, Xi'an, China (2018)
5. Bjeković, M.: Pragmatics of Enterprise Modelling Languages: A Framework for Understanding and Explaining. Ph.D. thesis, Radboud University, Nijmegen, the Netherlands (2018)
6. Bjeković, M., Proper, H.A., Sottet, J.S.: Towards a coherent enterprise modelling landscape. In: Sandkuhl, K., Seigerroth, U., Stirna, J. (eds.) Short Paper Proceedings of the 5th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling, Rostock, Germany, November 7-8, 2012. CEUR Workshop Proceedings, vol. 933. CEUR-WS.org (2012)
7. Bjeković, M., Proper, H.A., Sottet, J.S.: Embracing pragmatics. In: Yu, E.S.K., 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, Heidelberg, Germany (2014)
8. Bjeković, M., Proper, H.A., Sottet, J.S.: Enterprise modelling languages - just enough standardisation? In: Shishkov, B. (ed.) Business Modeling and Software Design - Third International Symposium, BMSD 2013, Noordwijkerhout, The Netherlands, July 8-10, 2013, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 173, pp. 1–23. Springer, Heidelberg, Germany (2014)
9. Bommel, P.v., Hoppenbrouwers, S.J.B.A., Proper, H.A., Roelofs, J.: Concepts and Strategies for Quality of Modeling. In: Halpin, T.A., Krogstie, J., Proper, H.A. (eds.) Innovations in Information Systems Modeling, chap. 9. IGI Publishing, Hershey, Pennsylvania (2008)
10. Carvalho, V.A., Almeida, J.P.A., Fonseca, C.M., Guizzardi, G.: Multi-level ontology-based conceptual modeling. Data & Knowledge Engineering **109**, 3–24 (2017)
11. Chen, P.P.: The Entity–Relationship Model: Towards a Unified View of Data. ACM Transactions on Database Systems **1**(1), 9–36 (March 1976)
12. Conklin, J.: The IBIS Manual: a short course in IBIS methodology. Touchstone (2003)
13. Conklin, J.: Dialogue Mapping: Building Shared Understanding of Wicked Problems. John Wiley & Sons, New York, New York (2005)
14. Conklin, J.: Wicked Problems and Social Complexity. Tech. rep., CogNexus Institute, Edgewater, Maryland (2006)
15. De Carvalho, V., Almeida, J., Guizzardi, G.: Using reference domain ontologies to define the real-world semantics of domain-specific languages. In: International Conference on Advanced Information Systems Engineering. pp. 488–502. Springer (2014)

16. Falkenberg, E.D., Verrijn–Stuart, A.A., Voss, K., Hesse, W., Lindgreen, P., Nilsson, B.E., Oei, J.L.H., Rolland, C., Stamper, R.K. (eds.): A Framework of Information Systems Concepts. IFIP WG 8.1 Task Group FRISCO, IFIP, Laxenburg, Austria (1998)
17. Figueiredo, G., Duchardt, A., Hedblom, M.M., Guizzardi, G.: Breaking into pieces: An ontological approach to conceptual model complexity management. In: 2018 12th International Conference on Research Challenges in Information Science (RCIS). pp. 1–10. IEEE (2018)
18. Frank, U.: 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. p. 72. IEEE Computer Society Press, Los Alamitos, California, Washington, DC (2002)
19. Frederiks, P.J.M., Weide, T.P.v.d.: Information Modeling: the process and the required competencies of its participants. Data & Knowledge Engineering **58**(1), 4–20 (July 2006), best paper award in NLDB 2004 conference
20. Guarino, B., Guizzardi, G., Mylopoulos, J.: On the philosophical foundations of conceptual models. Information Modelling and Knowledge Bases XXXI **321**, 1 (2020)
21. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Ph.D. thesis, University of Twente, Enschede, the Netherlands (2005)
22. Guizzardi, G.: On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. Frontiers in artificial intelligence and applications **155**, 18 (2007)
23. Guizzardi, G.: Theoretical foundations and engineering tools for building ontologies as reference conceptual models. Semantic Web **1**(1, 2), 3–10 (2010)
24. Guizzardi, G.: Ontology-based evaluation and design of visual conceptual modeling languages. In: Domain engineering, pp. 317–347. Springer (2013)
25. Guizzardi, G.: Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. In: International Conference on Conceptual Modeling. pp. 13–27. Springer (2014)
26. Guizzardi, G., Pires, L.F., Van Sinderen, M.: 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. pp. 691–705. Springer (2005)
27. Guizzardi, G., Figueiredo, G., Hedblom, M.M., Poels, G.: Ontology-based model abstraction. In: 2019 13th International Conference on Research Challenges in Information Science (RCIS). pp. 1–13. IEEE (2019)
28. Halpin, T.A., Morgan, T.: Information Modeling and Relational Databases. Data Management Systems, Morgan Kaufman, 2nd edn. (2008)
29. Harel, D., Rumpe, B.: Meaningful Modeling: What's the Semantics of "Semantics"? IEEE Computer **37**(10), 64–72 (2004). https://doi.org/10.1109/MC.2004.172
30. Hofstede, A.H.M.t., Proper, H.A.: How to formalize it?: Formalization principles for information system development methods. Information and Software Technology **40**(10), 519–540 (October 1998)
31. Hoppenbrouwers, S.J.B.A., Proper, H.A., Weide, T.P.v.d.: A fundamental view on the process of conceptual modeling. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (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, pp. 128–143. Springer, Heidelberg, Germany (June 2005)
32. ISO/IEC/IEEE: Systems and software engineering – Architecture description is an international standard for architecture descriptions of systems and software. Tech. Rep. ISO/IEC 42010, ISO (July 2011)
33. Krogstie, J.: A Semiotic Approach to Quality in Requirements Specifications. In: Kecheng, L., Clarke, R.J., Andersen, P.B., Stamper, R.K., Abou–Zeid, E.S. (eds.) Proceedings of the

IFIP TC8 / WG8.1 Working Conference on Organizational Semiotics: Evolving a Science of Information Systems. pp. 231–250. Kluwer, Deventer, the Netherlands (2002)
34. Lankhorst, M.M., Hoppenbrouwers, S.J.B.A., Jonkers, H., Proper, H.A., Torre, L.v.d., Arbab, F., Boer, F.S.d., Bonsangue, M., Iacob, M.E., Stam, A.W., Groenewegen, L., Buuren, R.v., Slagter, R.J., Campschroer, J., Steen, M.W.A., Bekius, S.F., Bosma, H., Cuvelier, M.J., ter Doest, H.W.L., van Eck, P.A.T., Fennema, P., Jacob, J., Janssen, W.P.M., Jonkers, H., Krukkert, D., van Leeuwen, D., Penders, P.G.M., Veldhuijzen van Zanten, G.E., Wieringa, R.J.: Enterprise Architecture at Work – Modelling, Communication and Analysis. The Enterprise Engineering Series, Springer, Heidelberg, Germany, 4th edn. (2017)
35. van der Linden, D.J.T., Proper, H.A., Hoppenbrouwers, S.J.B.A.: Conceptual understanding of conceptual modeling concepts: A longitudinal study among students learning to model. In: Iliadis, L.S., Papazoglou, M.P., Pohl, K. (eds.) Advanced Information Systems Engineering Workshops - CAiSE 2014 International Workshops, Thessaloniki, Greece, June 16-20, 2014. Proceedings. Lecture Notes in Business Information Processing, vol. 178, pp. 213–218. Springer, Heidelberg, Germany (2014)
36. Mahr, B.: On the epistemology of models. In: Abel, G., Conant, J. (eds.) Rethinking Epistemology, pp. 1–301. De Gruyter (2011)
37. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language: Overview. Tech. rep., W3C (February 2004)
38. Moody, D.L.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering **35**(6), 756–779 (2009)
39. Ogden, C.K., Richards, I.A.: The Meaning of Meaning – A Study of the Influence of Language upon Thought and of the Science of Symbolism. Magdalene College, University of Cambridge, Oxford, United Kingdom (1923)
40. OMG: OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2. Tech. rep., The Object Management Group, Needham, Massachusetts (November 2007)
41. OMG: Business Process Modeling Notation, V2.0. Tech. Rep. OMG Document Number: formal/2011-01-03, Object Management Group, Needham, Massachusetts (January 2011)
42. Op 't Land, M., Proper, H.A., Waage, M., Cloo, J., Steghuis, C.: Enterprise Architecture - Creating Value by Informed Governance. The Enterprise Engineering Series, Springer, Heidelberg, Germany (2008)
43. Proper, H.A.: Digital Enterprise Modelling - Opportunities and Challenges. In: Roelens, B., Laurier, W., Poels, G., Weigand, H. (eds.) Proceedings of 14th International Workshop on Value Modelling and Business Ontologies, Brussels, Belgium, January 16-17, 2020. CEUR Workshop Proceedings, vol. 2574, pp. 33–40. CEUR-WS.org (2020), http://ceur-ws.org/Vol-2574/short3.pdf
44. Proper, H.A., Bjeković, M.: Fundamental challenges in systems modelling. In: Mayr, H.C., Rinderle-Ma, S., Strecker, S. (eds.) 40 Years EMISA 2019. pp. 13–28. Gesellschaft für Informatik e.V., Bonn (2020)
45. Proper, H.A., Hoppenbrouwers, S.J.B.A., Veldhuijzen van Zanten, G.E.: Communication of enterprise architectures. In: Enterprise Architecture at Work – Modelling, Communication and Analysis [34], pp. 59–72
46. Proper, H.A., Verrijn–Stuart, A.A., Hoppenbrouwers, S.J.B.A.: 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, pp. 25–34. Australian Computer Society, Sydney, New South Wales, Australia (2005)
47. Rothenberg, J.: The Nature of Modeling. In: Artificial intelligence, simulation & modeling, pp. 75–92. John Wiley & Sons, New York, New York, United States of America (1989)

48. Sandkuhl, K., Fill, H.G., Hoppenbrouwers, S.J.B.A., Krogstie, J., Matthes, F., Opdahl, A.L., Schwabe, G., Uludag, Ö., Winter, R.: From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling. Business & Information Systems Engineering **60**(1), 69—80 (2018)
49. Scheer, A.W.: Architecture of Integrated Information Systems: Foundations of Enterprise Modelling. Springer, Heidelberg, Germany, Secaucus, New Jersey (1992)
50. Shannon, C.E.: A Mathematical Theory of Communication. In: The Bell System Technical Journal. vol. 22, pp. 379–423,623–656 (1948)
51. Snowden, D.J., Boone, M.E.: A leader's framework for decision making. Harvard Business Review **85**(11), 68–76 (November 2007)
52. Stachowiak, H.: Allgemeine Modelltheorie. Springer, Heidelberg, Germany (1973)
53. Thalheim, B.: The Theory of Conceptual Models, the Theory of Conceptual Modelling and Foundations of Conceptual Modelling. In: Handbook of Conceptual Modeling, pp. 543–577. Springer, Heidelberg, Germany (2011)
54. Thomasson, A.L., et al.: Fiction and metaphysics. Cambridge University Press (1999)
55. Wand, Y., Weber, R.: An Ontological Model of an Information System. IEEE Transactions on Software Engineering **16**(11), 1282–1292 (November 1990)
56. Wilmont, I., Barendsen, E., Hoppenbrouwers, S.J.B.A., Hengeveld, S.: Abstract Reasoning in Collaborative Modeling. In: Hoppenbrouwers, S.J.B.A., Rouwette, E.A.J.A., Rittgen, P. (eds.) proceedings of the 45th Hawaiian International Conference on the System Sciences, HICSS-45; Collaborative Systems track, Collaborative Modeling minitrack. IEEE Explore, Los Alamitos, California (2012)
57. Wood–Harper, A.T., Antill, L., Avison, D.E.: Information Systems Definition: The Multiview Approach. Blackwell, Oxford, United Kingdom (1985)
58. Wyssusek, B.: On Ontological Foundations of Conceptual Modelling. Scandinavian Journal of Information Systems **18**(1) (2006)
59. Wyssusek, B.: Ontological Foundations of Conceptual Modelling Reconsidered: A Response. Scandinavian Journal of Information Systems **18**(1), Article 8 (2006)
60. Zambon, E., Guizzardi, G.: Formal definition of a general ontology pattern language using a graph grammar. In: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 1–10. IEEE (2017)
61. Zarwin, Z., Bjeković, M., Favre, J.M., Sottet, J.S., Proper, H.A.: Natural modelling. Journal Of Object Technology **13**(3), 4: 1–36 (July 2014)