Towards Grounded Enterprise Modelling[⋆]

Hend*erik* A. Proper[1,4,5], Marija Bjeković[1], Bas van Gils[2] and
Stijn J. B. A. Hoppenbrouwers[3,4]

[1] Luxembourg Institute of Science and Technology (LIST), Belval, Luxembourg
[2] Strategy Alliance, Lelystad, the Netherlands
[3] HAN University of Applied Sciences, Arnhem, the Netherlands
[4] Radboud University Nijmegen, Nijmegen, the Netherlands
[5] University of Luxembourg, Luxembourg
e.proper@acm.org, marija.bjekovic@list.lu,
bas.vangils@strategy-alliance.com, stijn.hoppenbrouwers@han.nl

Abstract. This paper is concerned with the concept of grounding enterprise models in terms of an underlying fact-based model, as a way to add more meaning to these enterprise models. We motivate the need for doing so in terms of a fundamental understanding of conceptual modelling, and enterprise modelling in particular. We also clarify why, next to e.g. adding more meaning by using formal semantics, or mapping the model to a foundational ontology, it remains important to ground enterprise models on fact-based models that capture the natural way in which people converse about / in their domain. The presented concepts are illustrated by means of a running example, while also reflecting on, and summarising, the results of earlier experiments in grounding different enterprise models.

1 Introduction

Conceptual models, in their many different purpose and / or domain specific variations, play an increasingly important role in society. From conceptual database designs, via ontologies, domain models, process models, and actor models, to enterprise models in general, models are increasingly first class citizens in the organisations using them. Such models are not created as mere "one off" artefacts. They rather have a life of their own, covering a broad range of uses (from analysis and understanding, via simulation and design, to execution and monitoring), while involving an even broader variety of stakeholders / audiences.

In this paper, we take the perspective that conceptual models should (unless they only serve a temporary "throw away" purpose) include a definition of their meaning[6] in a way that is understandible to the model's audience. We therefore posit that a conceptual model should be grounded on an (underlying) fact-based model involving verbalisations using the terminology as it is actually *used* (naturally) by the people involved

[⋆] This work has been partially sponsored by the *Fonds National de la Recherche Luxembourg* (`www.fnr.lu`), via the ValCoLa and CoBALab projects.

[6] In principle, we would prefer to use the word "semantics" here. However, since the word "semantics", in our computer science oriented community, tends to be equated to only mean "formal semantics", we will use the word *meaning*.

in / with the modelled domain. We see this as a key enabler for the transferability of models across time and among people, in particular in situations where the model needs to act as a *boundary object* [2].

At the same time, we can see how purpose / domain specific modelling language (e.g. process models, goal models, actor models, value models, architectural models, etc) create models using "boxes and lines" based constructs / abbreviations that only provide a limited linkage to the (natural) language as used by the model's audience. In general, the only link in this regard are the names used to label the "boxes". Relationships are replaced by generic graphical representations in terms of arrows and lines capturing relations such as "assigned to", "part of", "realises", "aggregates", "triggers", while leaving no room for situation specific nuance. Examples of such enterprise modelling languages / frameworks include a.o. ArchiMate [17], ARIS [29] and BPMN [23].

While the abstract, and more compact, notations of purpose / domain specific modelling languages enable a more compact representation of models, they offer no means to provide a "drill down" to an underlying grounding in terms of well verbalised fact types that capture, and honour, the original natural (language) nuances. The basic idea, as presented in this paper, is therefore to ground enterprise models on a fact-based model of the domain being modelled and, in line with the tradition of fact-based modelling, do so based on sample facts drawn from the domain being modelled. A fact model, grounding an enterprise model, might actually have a broader scope than the enterprise model, so as to capture even more of the relevant context.

The remainder of this paper is structured as follows. In Section 2, we discuss our fundamental view conceptual modelling, and enterprise modelling in particular. Section 3 then continues with a discussion on the need for a better grounding of enterprise models. In Section 4, we then illustrate this grounding in terms of a concrete example. In that Section, we will also briefly revisit some of the earlier experiments in grounding enterprise models. Section 5 then concludes the paper.

2 A fundamental view on conceptual modelling

This Section is concerned with our fundamental view on conceptual modelling. We consider a model to be: "*an artefact acknowledged by an observer as representing some domain for a particular purpose*" [5]. This definition of model is strongly based on the work reported in e.g. [30, 28, 10, 31], as well as our own earlier work [15, 16].

The *observer* in our definition refers to the group of people consisting of model creators and model audience. On one extreme, it can refer to the entire society, on the other extreme, it can refer to an the individual.

Similar to [10], we define *domain* as any "part" or "aspect" of the world *considered relevant by the observer*. The term *world* here refers to "reality", as well as to possible worlds [35]. In the context of conceptual database design, this notion of *domain* is also referred to as the *universe of discourse* [18].

The *purpose* of a model is often considered as the main discriminant of the added value of a model [30, 28, 31]. We understand *purpose* as aggregating two interrelated dimensions: (1) the *domain* that the model (should) pertains to, and (2) the intended *usage* of the model by its intended *audience*.

In terms of the above, we define a conceptual model as being *a model where its purpose involves a need to capture knowledge about the represented domain*. In other words, a model answering a need to understand and / or articulate the workings and / or structure of some domain. Such a model needs to reflect human cognition in that it concerns concepts, their relationships, and relevant properties. This is what makes it a *concept*ual model. In line with this, we consider an *enterprise model* to be a *conceptual model that represents some part and / or aspect of an organisation / enterprise* [20].

A specific class of conceptual models are *conceptual (database) schemas*, which are conceptual models of the (implementation free) structure of a *universe of discourse* as it is to be captured in a database [18]. Or, as [18] puts it: "*The description of the possible states of affairs of the universe of discourse including the classifications, rules, laws, etc., of the universe of discourse*".

This brings us to the role of modelling languages. As defined in [3], we regard a modelling language as having a *linguistic function* and a *representational function*.

The *linguistic function* refers to the ability of a modelling language to frame the discourse about a domain and shaping the observer's conception of a domain. In this regard, a modelling language should provide a *linguistic structure*, involving a specific classification of concepts to be used in the discourse about the world (the embodied *world view*, or *Weltanschauung*). This linguistic structure will differ between e.g. a modelling language for value modelling and one for process modelling.

For modelling languages, the so-called meta-model will largely define the linguistic structure. Additional (linguistic) structure may be added by combining this with e.g. formal semantics, providing a normative view on what models are semantically sound / correct, and which are not [13]. Another way to increase the linguistic structure, is by the (enforcement of the) use of e.g. a foundational ontology [11].

The *representational function* refers to the ability of the language to expres the conceived domain in a purposeful model. This generally involves a *representation system* involving an abstract and a concrete syntax of the modelling language.

It is important to acknowledge that the *linguistic structure*, being its essential world view (*Weltanschauung*), may not only limit the freedom of what can be expressed in a model. It may even limit, or at least influence, the way in which modellers observe the domain. This may lead to situations where a modelling language may "feel unnatural", in the sense that the linguistic structure puts to many restrictions on a modeller's "freedom of expression". This may, especially, become problematic when a model is used as a boundary object across communities [2].

At an anecdotical level, the influence of a specific (restricted) world view corresponds to the *hammer* and *nail* paradigm. At a more fundamental level, it corresponds to the notion of linguistic relativity [32][7], which states that the structure of a language determines, or greatly influences, the modes of thought and behaviour characteristic of the culture / context in which it is spoken. As we will see in the next Section, this point is also key in our observations that a non-normative means is needed to be able to add more (natural language based) meaning to enterprise models.

[7] More colloquially also known as the Sapir-Whorf hypothesis.

3 The need to ground enterprise models

Enterprise models, being conceptual models, involve *concepts* and their *relations*, as well as possibly a *typing* of these in terms of modelling constructs. Consider, as an example, the ArchiMate [17] model as shown in Figure 1. It contains, a.o., the concepts Patient, Doctor, Form, Examine and Diagnose. The icons in the boxes indicate wether a concept is a *role* (e.g. Patient), *activity* (e.g. Examine) or a passive *object* (e.g. Form). The line with the double dots is a so-called *assignment* relation. For example, Doctor and Patient are assigned to the Examine activity. The arrows correspond to triggering rules, so e.g. the Examine activity is triggered by the Register activity.

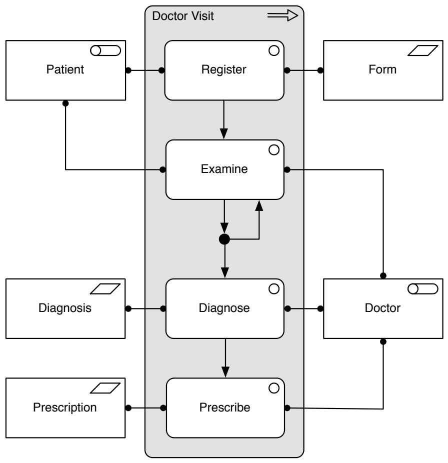

Fig. 1. Example ArchiMate model of a Doctor Visit

The example in Figure 1 also illustrates the point that such models only provide a limited linkage to the (natural) language as used by the model's audience. Consider the assignment of Doctor and Patient to the Examine activity. One can only infer that a Doctor examines a Patient, when using contextual knowledge about what Doctors and Patients are, and what usually happens in an Examination. This is, of course, a rather simple example. However, in real-world cases, it is likely to become more difficult to "re-construct" the more precise meaning, in particular when a broader audience with a higher variety of backgrounds are involved (e.g. when a model acts as a *boundary object* [2]), or when a longer period of time has passed since the model was created.

Different ways of adding more meaning, and precision, to models can be used. We already mentioned formal semantics and the use of foundational ontologies. At a more fundamental level, these lead to refinements of the *linguistic structure* of the modelling language used, and as such provide *normative* restrictions on the freedom to express models. These normative restrictions, which might be said to *freeze* the (modelling) language [14], are likely to hamper, and stress, the actual modelling process. At the same time, these normative restrictions certainly have a clear benefit, depending on the *purpose* of the model [4, 5]. When an enterprise model is to act as a sector / industry wide reference model, it is certainly good to use a shared foundational ontology. Also, when a model is to be used as a base for formal analysis, code generation, or even execution, then a formal semantics is indeed called for.

In this paper, however, we focus on another approach to capture more meaning in an enterprise model, targeting situations where it is necessary to capture more *organisation specific* meaning, and enable those who are involved in modelling processes, to stay close to the language / terms they are used to. In such situations, we suggest a more *descriptive* approach, rather than a *normative* approach, when adding more meaning. This, we think, resonates well with the objectives of fact-based modelling [12, 22], where the conceptual structures of a domain are expressed using fact(s) (types) in an explicit format, capturing the deeper conceptual meanings in a language that is understandable to the various stakeholders involved.

Grounding an enterprise model on a fact-based model has the potential added advantage that enterprise modelling can also benefit from the use of sample facts to validate the models (by e.g. a population check), in the sense that they can be more easily validated by domain experts. Here, we also see a strong analogy to grounded theory [21], which requires theories to be grounded in actual observed data; i.c. the sample facts that fact-based approaches start out from. This is also why we use the word *grounding*. An additional advantage is that the modelling procedure as suggested by most fact-based modelling approaches, ensures one starts out from *elementary* facts. Applying this in the context of enterprise modelling, could also lead to better models in terms of normalised relations in the model.

Each of the strategies to add more meanings to a model is bound to add a *specification burden* to the modelling process. The approach as suggested in the next Section, does so by requiring more elaborate verbalisations, and even the identification of elementary sample facts. Adding more formal semantics to models, or using foundational ontologies, adds more burden on the modelling process by putting normative restrictions on the linguistic function. Wether these extra "burdens" are worth the effort, depends on the purpose(s) of the model.

4 Grounded enterprise modelling

The aim of this Section, is to exemplify the grounding of enterprise models. Inspired by (1) earlier experiences with the need to better manage domain concepts during software and / or information system development [25], (2) work on explicitly identifying the need to introduce modelling concepts into a modelling language [16], as well as (3) the way in which the ArchiMat language was designed in terms of a series of layers with increasingly more specific modelling concepts [19], we developed the idea to use generic conceptual models to ground other, more specific, models on top of a semantically rich understanding of the domain in terms of a fact-based model [24]. In developing this approach, we also conducted some initial experiments in grounding enterprise models, involving (1) activity models [26, 9, 8], (2) system dynamics models [33, 34], and (3) architecture principles [6].

A concrete example of how the ArchiMate model from Figure 1 can be grounded on an ORM fact-based model has been depicted in Figure 2, 3 and 4 respectively. In

Figure 2, we see an ORM model[8] dealing with patients visiting a doctor. Patients fill out forms in order to register, they can be examined by a doctor, doctors produce diagnoses, as well as prescribe possibly prescriptions.

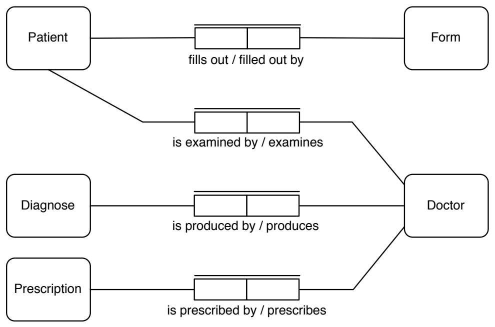
Patient
Form
fills out / filled out by
is examined by / examines
Diagnose
Doctor
is produced by / produces
Prescription
is prescribed by / prescribes

Fig. 2. Doctor Visit example; ORM grounding

When applying the ORM CSDP [12], one is also "invited" to carefully reflect on the question if Doctor and Patient should be treated as specialisation of a same super-type, such as Person. Especially, since a Doctor may also become a Patient. In the example, we assume that such a choice was not made. However, *making* such a conscious choice should be part of the grounding process.

What is missing in Figure 2 is the temporal order in which these facts occur, as well as the fact that these activities take place in the context of a Doctor Visit. This leads to the situation as shown in Figure 3, where we have also adorned the roles with icons corresponding to the modelling concepts of the ArchiMate language [17].

In adding a temporal semantics to ORM [27, 7] we assume that the regular ORM constraints (cardinality, etc.) need to apply at each individual moment in time. So, a mandatory role constraint, such as the one marked with a), should apply at each individual moment in time. In other words, if a Register occurence takes place during some period in time, then (also during that period in time), it must be taking place in the context of some Doctor Visit.

Normally, ORM uniqueness constraints are represented with a single bar over the involved roles. Now, consider the uniqueness constraint marked with b). If this one would have been marked with only a single bar, it would have signified that at each moment in time, a Register occurence can only be for one Doctor Visit. This would still make it possible for one Register occurence during some time period T to be assigned to two different Doctor Visits, but at non coinciding intervals in time T_1 and T_2, with $T_1, T_2 \subset T$. The double bar, therefore, signifies that the Registrer occurence can be part of a Doctor Visit once, ever. The patient can of course register for *an other* Doctor Visit by filling out *an other* form.

The required temporal order of events is depicted with an open arrow connecting the involved roles. See, for example, the one marked with c). This states that for

[8] To keep the diagram clean, we have omitted all of the so-called reference schemes, which identify how e.g. a Doctor or a Patient is referred to in this domain

Doctor Visit, we cannot see a Register occurrence after we have started to see (an) Examine occurence(s). We also see (indicated by the open arrow further below) that (the way it is modelled in the *example*) after a Diagnose occurence has taken place, for a given Doctor Visit, we can no longer see further Examine occurrences in the context of *this* Doctor Visit. Note also, that a Doctor Visit is only allowed to have one Diagnose occurence, but multiple Examine occurrences, as signified by the double bars.

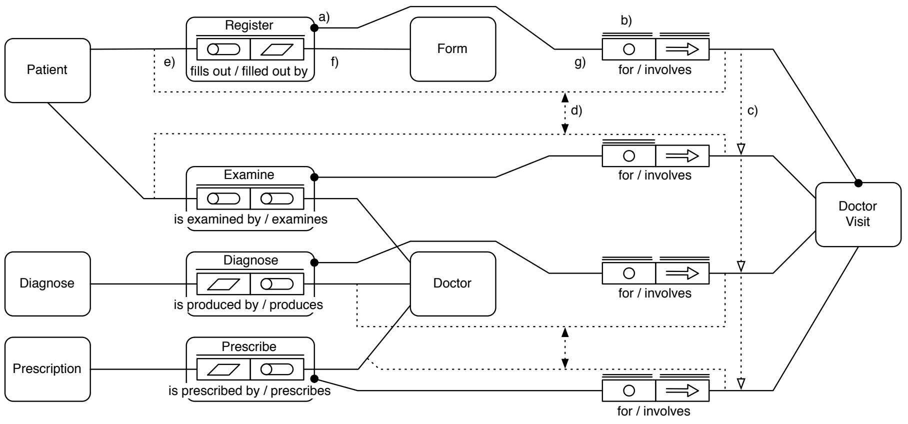

Fig. 3. Doctor Visit example with temporal ordering and ArchiMate mapping

The constraint pattern marked d) is also of interest. It insures that the Patient filling out the Form is also the Patient who is to be examined (in the context of one Doctor Visit). Similarly the Doctor doing the diagnosing is also required to be the Doctor writing the prescription.

The process flow as depicted in Figure 3 does not involve split / join junctions. Such structures could, however, also be modelled using similar temporal constraints. However, advanced workflow / temporal-ordering patterns, are probably best left to a dedicated modelling language [1]. In grounding enterprise models, we think it is wisest to focus on grounding the main conceptual structure of the domain.

Figure 3 also shows the a classification, by means of icons, of roles in terms of the modelling concepts from the ArchiMate language [17]. Consider, for instance, the role marked with e). When a Patient fills out a form, then they are, in terms of ArchiMate enacting a *business role*. The form, see f), then plays the passive role of a *business object*. The Register occurrence, see g), plays the role of a *business activity* in the context of a composed *business process* Doctor Visit.

In the case of larger examples, even when limited to educational settings, diagrams in the style of Figure 3 can easily become rather large. Therefore, we would suggest to use a graphical abbreviation in the ORM diagrams, in terms of a State Sequence (complex) object type, as used on the left hand side of Figure 4. The version represented on the right hand side, would actually result in a more ArchiMate-alike notation. Note the added fact verbalisations, as well as the addition of the more specific constraints on role participations of Doctors and Patients

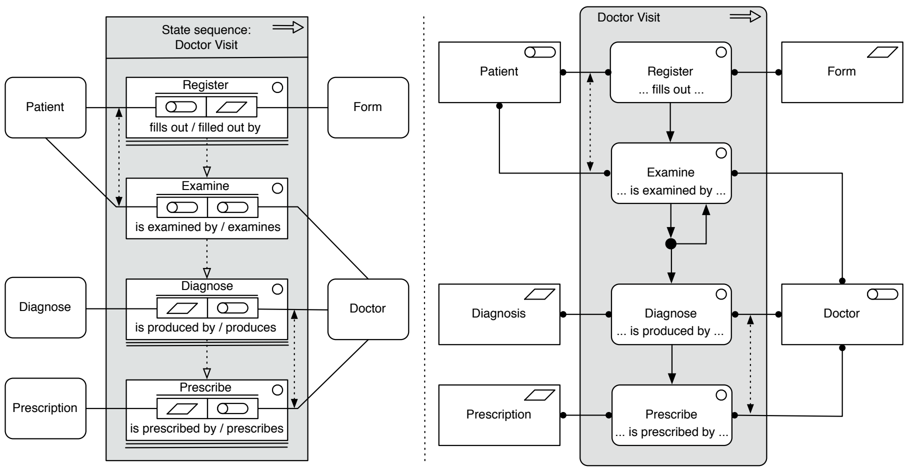

Fig. 4. Doctor Visit example, notational variations

5 Conclusion

In this paper, we presented the idea of grounding enterprise models in terms of fact-based models, in order to add more domain specific meaning to enterprise models. We discussed the need for doing so in terms of a fundamental understanding of conceptual modelling. We also argued why grounding enterprise models on fact-based models provides a complementary (descriptive) approach next to (more normative) alternatives. A grounding on fact-based models, allow us to leverage the traditional fact-based modelling advantages of capturing the deeper conceptual meanings in a language that is understandable to the various stakeholders involved. The presented concepts were illustrated by means of a running example, while also reflecting on, and summarising, the results of earlier experiments in grounding different enterprise models.

As a next step, we aim to further elaborate the grounding of ArchiMate models, as well as other enterprise modelling languages, while also formalising the used mapping mechanisms. In addition, to the theoretical underpinning as discussed in Section 2, and the early experiments as reported in [26, 9, 8, 6, 33, 34], we also plan to conduct more usage oriented experiments. Does adding a grounding to enterprise models lead to: *Models of higher quality? Models that can be more easily communicated among differen actors? Models that can more easily understood at a later point in time?*

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
2. Abraham, R., Niemietz, H., de Kinderen, S., Aier, S.: Can boundary objects mitigate communication defects in enterprise transformation? Findings from expert interviews. In: Jung, R., Reichert, M. (eds.) Proceedings of the 5th International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2013). LNI, vol. 222, pp. 27–40. Gesellschaft für Informatiek, Bonn, Germany (2013)

3. Bjeković, M., Proper, H.A., Sottet, J.S.: Embracing Pragmatics. In: Yu, E., Dobbie, G., Jarke, M., Purao, S. (eds.) Proceedings of the 33rd International Conference on Conceptual Modeling (ER 2014). LNCS, vol. 8824, pp. 431–444. Springer (2014)
4. 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. LNBIP, vol. 173, pp. 1–23. Springer (2014)
5. Bjeković, M., Sottet, J.S., Favre, J.M., Proper, H.A.: A Framework for Natural Enterprise Modelling. In: Proceedings of the 15th IEEE Conference on Business Informatics (CBI 2013). pp. 79–84. IEEE Computer Society Press, Los Alamitos, California (2013)
6. van Bommel, P., Buitenhuis, P.G., Hoppenbrouwers, S.J.B.A., Proper, H.A.: Architecture Principles – A Regulative Perspective on Enterprise Architecture. In: Reichert, M., Strecker, S., Turowski, K. (eds.) Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2007). pp. 47–60. No. 119 in LNI, Gesellschaft für Informatik, Bonn, Germany (2007)
7. van Bommel, P., Frederiks, P.J.M., van der Weide, T.P.: Object–Oriented Modeling based on Logbooks. The Computer Journal 39(9), 793–799 (1996)
8. van Bommel, P., Hoppenbrouwers, S.J.B.A., Proper, H.A., van der Weide, T.P.: On the Use of Object-Role Modeling For Modeling Active Domains, pp. 123–145. Research Issues in System Analysis and Design, Databases and Software Development, IGI Publishing, Hershey, Pennsylvania (2007)
9. van Bommel, P., Hoppenbrouwers, S.J.B.A., Proper, H.A., van der Weide, T.P.: On the use of Object-Role Modelling to Model Active Domains. In: Halpin, T.A., Krogstie, J., Proper, H.A. (eds.) Proceedings of the 13th Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2008), held in conjunction with the 20th Conference on Advanced Information Systems Engineering (CAiSE 2008), Montpellier, France. vol. 337, pp. 473–484. CEUR-WS.org (June 2008)
10. 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)
11. Guizzardi, G.: On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: Vasilecas, O., Eder, J., Caplinskas, A. (eds.) Databases and Information Systems IV - Selected Papers from the Seventh International Baltic Conference, DB&IS 2006, July 3-6, 2006, Vilnius, Lithuania. Frontiers in Artificial Intelligence and Applications, vol. 155, pp. 18–39. IOS Press (2006)
12. Halpin, T.A., Morgan, T.: Information Modeling and Relational Databases. Data Management Systems, Morgan Kaufman, 2nd edn. (2008)
13. ter Hofstede, A.H.M., Proper, H.A.: How to Formalize It? Formalization Principles for Information Systems Development Methods. Information and Software Technology 40(10), 519–540 (October 1998)
14. Hoppenbrouwers, S.J.B.A.: Freezing Language; Conceptualisation processes in ICT supported organisations. Ph.D. thesis, University of Nijmegen, Nijmegen, the Netherlands (2003)
15. Hoppenbrouwers, S.J.B.A., Proper, H.A., van der Weide, T.P.: A Fundamental View on the Process of Conceptual Modeling. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) 24 International Conference on Conceptual Modeling (ER 2005), Klagenfurt, Austria. LNCS, vol. 3716, pp. 128–143. Springer (2005)
16. Hoppenbrouwers, S.J.B.A., Proper, H.A., van der Weide, T.P.: Understanding the Requirements on Modelling Techniques. In: Pastor, O., Falcao e Cunha, J. (eds.) 17th International Conference on Advanced Information Systems Engineering, CAiSE 2005, Porto, Portugal. LNCS, vol. 3520, pp. 262–276. Springer (June 2005)

17. Iacob, M.E., Jonkers, H., Lankhorst, M.M., Proper, H.A., Quartel, D.A.C.: ArchiMate 2.0 Specification. The Open Group (2012)
18. ISO/IEC JTC 1/SC 32 Technical Committee on Data management and interchange: Information processing systems – Concepts and Terminology for the Conceptual Schema and the Information Base. Tech. Rep. ISO/TR 9007:1987, ISO (1987)
19. Lankhorst, M.M., Proper, H.A., Jonkers, H.: The Anatomy of the ArchiMate Language. Intern. Journal of Information System Modeling and Design (IJISMD) 1(1), 1–32 (2010)
20. Magalhães, R., Proper, H.A.: Model-enabled Design and Engineering of Organisations. Organisational Design and Enterprise Engineeering 1(1), 1–12 (2017)
21. Martin, P.Y., Turner, B.A.: Grounded Theory and Organizational Research. The Journal of Applied Behavioural Science 22(2), 141–157 (1986)
22. Semantics of Business Vocabulary and Rules (SBVR). Tech. Rep. dtc/06–03–02, Object Management Group, Needham, Massachusetts (March 2006)
23. OMG: Business Process Modeling Notation, V2.0. Tech. Rep. OMG Document Number: formal/2011-01-03, Object Management Group (January 2011)
24. Proper, H.A.: Grounded Enterprise Modelling. DaVinci Series, Nijmegen Institute for Information and Computing Sciences, Radboud University, Nijmegen, the Netherlands (2008)
25. Proper, H.A., Bleeker, A.I., Hoppenbrouwers, S.J.B.A.: Object–Role Modelling as a Domain Modelling Approach. In: Grundspenkis, J., Kirikova, M. (eds.) Proceedings of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD'04), held in conjunctiun with the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004),. vol. 3, pp. 317–328. Riga, Latvia (June 2004)
26. Proper, H.A., Hoppenbrouwers, S.J.B.A., van der Weide, T.P.: A Fact–Oriented Approach to Activity Modeling. In: Meersman, R., Tari, Z., Herrero, P. (eds.) On the Move to Meaningful Internet Systems 2005: OTM Workshops – OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005. LNCS, vol. 3762, pp. 666–675. Springer (2005)
27. Proper, H.A., van der Weide, T.P.: EVORM – A Conceptual Modelling Technique for Evolving Application Domains. Data & Knowledge Engineering 12, 313–359 (1994)
28. 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)
29. Scheer, A.W., Schneider, K.: ARIS – Architecture of Integrated Information Systems. In: Bernus, P., Mertins, K., Schmidt, G. (eds.) Handbook on Architectures of Information Systems, pp. 605–623. Springer (2006)
30. Stachowiak, H.: Allgemeine Modelltheorie. Springer (1973)
31. Thalheim, B.: The Theory of Conceptual Modelling and Foundations of Conceptual Modelling. In: Handbook of Conceptual Modeling, pp. 543–577. Springer (2011)
32. Tohidian, I.: Examining Linguistic Relativity Hypothesis as One of the Main Views on the Relationship Between Language and Thought. Journal of Psycholinguistic Research 38(1), 65–74 (2009)
33. Tulinayo, F., Hoppenbrouwers, S.J.B.A., Proper, H.A.: Integrating System Dynamics with Object-Role Modeling. In: Persson, A., Stirna, J. (eds.) First IFIP WG 8.1 Working Conference on The Practice of Enterprise Modeling: from Business Strategies to Enterprise Architectures Stockholm, Sweden 12-13 November, 2008. LNBIP, Springer (November 2008)
34. Tulinayo, F.P., van Bommel, P., Proper, H.A.: Enhancing the System Dynamics Modeling Proces with a Domain Modeling Method. International Journal of Cooperative Information Systems 22(02), 1350011 (2013)
35. Wyssusek, B.: On Ontological Foundations of Conceptual Modelling. Scandinavian Journal of Information Systems 18(1) (2006)