Finding the Right Level of Abstraction

Cédric Jeanneret
Department of Informatics, University of Zurich, Switzerland
jeanneret@ifi.uzh.ch

Abstract

Today, modelers must rely on their instinct and experience to decide how much detail of a system is worth being modeled. This ad-hoc modeling may result in models that are either too abstract or too detailed for their intended use. We propose to investigate objective measurement of a model's abstractness and systematic guidance to attain the right level of abstraction. Such a contribution has the potential to improve the quality of models and reduce the amount of time needed for modeling activities.

1. Motivation and Research Problem

Abstraction plays a central role in Software Engineering. It makes possible the separation of the specification of a software artefact from its realization. The specification enables both the reuse of a realization [1] and its testing [2]. In requirements engineering, abstraction permits the isolation of relevant properties in a problem domain from the rest, making the problem more tractable for engineers.

Finding the right level of abstraction for a model is essential. A model may loose its value if it lacks some important details. Conversely, the time spent on the elaboration of irrelevant details is lost purposelessly. While this issue is general to all kinds of models, it becomes very challenging for requirements specifications at the boundary of the problem and solution domains. Such models serve for a wide range of purposes and are interpreted by various project stakeholders.

Today, deciding how much and which details to mention in a model is based on the modeler's skill and experience. As Kramer observes in [3], not every one is equally endowed for abstract thinking. Improving the education of abstraction skills can only partially solve the problem. We think that building models at the right level of abstraction is too important to depend solely on the modeler's skills. Thus, our research project aims to define objective measurement of a model's abstractness and systematic guidance to attain the right level of abstraction. Albeit much research effort has been devoted to modeling and abstraction, the state of the art cannot support modelers in objectively assessing and systematically improving the level of abstraction of their models in a satisfactory manner.

Abstraction is a fundamental property of models [4]. Thus, a large extent of the modeling literature (including [5], [6], [7]) is devoted to it. Still, our current understandings of the phenoma is not deep enough for the definition and validation of a measure.

Several model quality frameworks identify abstractness as a quality criterion for models, often under different names such as *minimalism* in [8] or the *right level of detail* in [9]. However, none of these frameworks propose formal measurements for it. Davis et al. explain in [9] that it is difficult to measure this criterion, because it is highly dependent of the usage scenario of the model. Our research is precisely aimed at overcoming this difficulty.

There exist some guidelines for modeling such as those presented in [8]. But they are either language specific, independent of the modeling purpose or not operationalized. We search for guidelines that can easily be tailored for any modeling language and any modeling purpose.

Levels of abstraction are not only useful to assess and improve model quality. In software product management, Gorschek et al. [10] propose to sort and work-up incoming requirements based on their level of abstraction. To this end, they identified four abstraction levels (from higher to lower): *product*, *feature*, *function* and *component*. These levels form an ordinal scale for a measure of abstraction. In this approach, a requirement's abstractness is subjectively evaluated for predefined purposes.

The *model type* system, presented by Steel et al. in [11], makes model transformations resilient to (minor) metamodel changes that should not prevent their execution. We consider this approach as a nominal scale measurement of a model's abstractness: either

its metamodel contains the elements required by the transformation or not.

2. Research Ideas

We draw our inspiration from the seminal work of Lindland et al. [12] on the quality of conceptual models. Consider the set $\mathcal{U}$, the infinite set of statements correctly rooted to the orginial but not necessarily relevant that could be made about a system or a problem domain under study. Modeling consists of picking up some of these statements and making them explicit in a model m.

Every model is created with a purpose in mind. Typically, one writes a model to either document an existing system or specify a system to be built. It is then possible to infer, maybe mechanically, new statements about the system under study from the statements made in m. Possible inferences include mental understanding, execution simulations, performance analysis or any other model transformations. We call a particular inference procedure an *interpretation operation*. When performed, such an operation reads statements in the model m and outputs a new set of statements. If a statement is actually read during the operation's execution, we qualify it as *relevant*.

The pertinence of a model can be assessed in a similar manner as the result of an information retrieval task. Typical measures include *precision* and *recall*:

$$\begin{aligned} precision(m, op) &= \frac{|m \cap \mathcal{A}_{op}|}{|m|} \\ recall(m, op) &= \frac{|m \cap \mathcal{A}_{op}|}{|\mathcal{A}_{op}|} \end{aligned}$$

with $\mathcal{A}_{op}$ being the set of all relevant statements in $\mathcal{U}$ for the interpretation operation op.

We believe that these measures are related to the level of abstraction of a model or its *abstractness*. Informally, a model with a precision smaller than 100% contains irrelevant details. In other words, the model is too detailed for the considered purpose. On the other hand, a model with a recall smaller than 100% lacks some relevant details and is therefore too abstract for its intended use.

Thus, the abstractness of a model reflects its capability to be efficiently used, instead of $\mathcal{U}$, for a given purpose, represented by an interpretation operation. In this sense, abstractness is an essential quality criterion for models. If a model's abstractness is too high, the result of the interpretation operation may be inaccurate or even wrong. If it is too low, excessive time has been spent on the modeling activity and the processing time of the interpretation operation may be longer. Also, the risk of early commitment to incorrect albeit irrelevant details is not to be neglected.

In the literature, abstractness has already been identified as an essential property of models [6], [4] or as a quality criterion for models [9], [8]. But, to the best of our knowledge, no formal measurement method has yet been developed to measure the abstractness of models and, thus, there is no guidelines for its improvement. As Moody points out in [13], these two elements are extremely important to objectively evaluate the quality of a model and successfully improve it.

3. Research Questions

Our primary research goal is to support modelers with measures and guidelines related to the level of abstraction of their models, particularly when modeling requirements. We need first to gain further knowledge about abstraction. Thus, the first research questions seek to build a theory about modeling and abstraction.

RQ 1.1: Does abstractness exist?

Admittedly, models are abstract representations of an original [6]. But any measure relies on a consensus about the attribute to be measured. With this question, we are interested in the existence and the strength of a consensus about the abstractness of a model.

RQ 1.2: What is it like?

We need to identify the main constructs of a theory about abstraction and their relationships. In the literature, the abstractness of a model is often associated with its genericity [14] or the simplicity of its representation [15]. Incompleteness may also explain the absence of relevant statements. The theory that we intend to build shall draw a clear distinction among these attributes.

RQ 1.3: What are its properties?

Similar to the approach in [16], we intend to build a set of axioms that encodes intuitions and observations related to abstraction. These axioms will define unambiguously the concepts identified by the research question 1.2. They will form a firm foundation for the measurement of abstractness.

We are especially interested in axioms that relate *construction operations* on models to the change of abstractness they introduce. Such construction operations include the refinement of a specification element (e.g., a black-box component) towards its realization or the adjunction of a behavioral description to a structural model.

RQ 2: How can we measure the abstractness of a model?

As DeMarco put it in [17, p. 3], one cannot control what one cannot measure. Without proper feedback, it is difficult to assess and control the adequacy of a model for a particular purpose. Today, this is achieved in an ad-hoc manner, based on modelers' instincts and experience. Consequently, we are interested in formal and objective measurement of abstractness.

The research questions 1.x will characterize abstractness and, to some extent, define a measure for it. This research question addresses challenges related to the practical measurement of abstractness. One of the main difficulties resides in determining the set $\mathcal{A}_{op}$ for the measures we proposed in the previous section.

Furthermore, these measures can only be computed if they are used on formal models and automated interpretation operations. However, none of these assumptions hold for common requirements engineering models and their usages. Although the state of the art in RE includes some automatic analyses of formal models [18], specifications are usually interpreted mentally by stakeholders. Also, few specifications are formal. Still, techniques exist to build a semi-formal model from an informal specification [18]. Thus, we seek for measurements at least applicable to semi-formal specifications and to mental interpretation operations.

To achieve this, we envisage defining a set of *usage profiles* for every major category of modeling purposes such as specification, documentation, communication or testing. The result of the measurement process will be a vector, whose components measure the abstractness of a model for the corresponding usage.

RQ 3: What is an effective strategy to build a model at the right level of abstraction for a given purpose?

The evaluation of the abstractness of a model is only a first step. Modelers need to know how they should elaborate or simplify their models so that they eventually reach an adequate level of abstraction. Based on the indications given by our measure, we can provide them with guidelines or suggestions.

We expect our contribution to be especially valuable for modeling languages that define several views on a system such as UML or ADORA [19]. The richness of these modeling languages exacerbates the problem of deciding which details need to be modeled for a given purpose. Thus, we will extend an editor supporting one of these languages with this novel feature.

4. Research Methodology and Validation

The validation and development of software measurement is still subject to discussion. For this project, we will follow the methodology presented in [20].

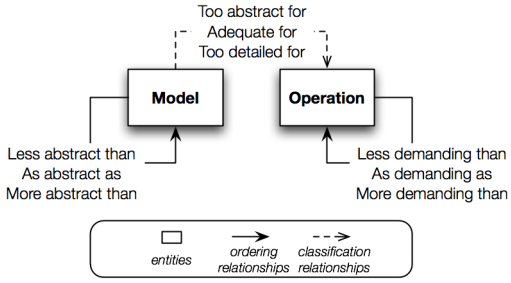

Figure 1. Empirical system for the abstraction phenomenon

To build our theory (RQ 1.x) and validate our measure, we will establish an *empirical system* as described in [21]. An empirical system is a set of entities and a set of relationships among them empirically established. In our case, entities are models and interpretation operations. The relationships we are interested in are depicted in Figure 1. Experts will classify pairs of model/operation according to an *adequacy* criterion. Furthermore, models and operations will be ranked according to an *abstractness* resp. a *demand for details* criterion.

We plan to build the empirical system incrementally. We will add models or operations into the empirical system as we progress with our theory and identify representative entities. Also, the group of surveyed experts will grow along the research project from our research group to an international selection of researchers and practitioners. Because software is intangible and thus difficult to grasp, we will begin our investigations with some geographical maps. Such models are more illustrative than models involved in the development of software systems. Therefore, they have often been used as illustrative examples in the literature (e.g. [22], [5]). We will shift to software-related models once sufficient insights have been gained.

If ranking and classification emerge consensually, we will have strong evidence for the existence of abstractness (RQ 1.1). Technically, the strength of the consensus can be evaluated with non-parametrical statistics such as *Kendall's coefficient of concordance* and *Fleiss' kappa* [21].

The measure (RQ 2) must satisfy the axioms developed for RQ 1.3. Our measure will also be validated against the empirical system. A *valid* measure is a measure that respects the *representation condition*, i.e., the measure reflects empirical relations. The degree to which a measure respects the representation condition can be measured with the *Kendall correlation* [21].

Validity, despite its necessity, is not a sufficient qual-

ity for assessment or control purposes. Schneidewind requires such measurements to present good *association*, *consistency*, *discriminative power*, *tracking* and *repeatability* [23]. We will evaluate our measurement against these criteria on the empirical system.

We intend to validate our guidelines (RQ 3) with a controlled experiment. Students from our modeling class will be asked to model a system for a given purpose. We will consider two treatments: (1) modeling with an extended editor (providing feedback and guidance) and (2) modeling with the basic editor. The dependent variables will be both the time elapsed before the modeler claims to have completed the modeling task and the adequacy of the resulting model for the purpose. To mitigate the threat to external validity posed by the use of students, we plan to survey experienced modelers with a questionnaire to assess the pertinence and usefulness of our guidelines.

5. Conclusion

In this proposal, we propose to investigate the abstractness of a model as an aspect of its quality. This quality does not concern the system being modeled directly; it rather concerns the ability of the model to play its role as a description of an original for a given usage.

Our contribution will allow requirements engineers to evaluate the adequacy of their models for a given purpose. In addition, we will provide them with support to choose which details should be added or removed so that their models present an appropriate level of abstraction. Our research will also extend scientific knowledge about modeling and abstraction, especially by providing a valid measure of the abstractness of a model.

References

[1] C. W. Krueger, "Software reuse," *ACM Computing Surveys*, vol. 24, no. 2, pp. 131–183, 1992.

[2] W. Prenninger and A. Pretschner, "Abstractions for model-based testing," *Electronic Notes in Theoretical Computer Science*, vol. 116, pp. 59–71, 2005.

[3] J. Kramer, "Is abstraction the key to computing?" *Communications of the ACM*, vol. 50, no. 4, pp. 36–42, 2007.

[4] B. Selic, "The pragmatics of model-driven development," *IEEE Software*, vol. 20, no. 5, pp. 19–25, 2003.

[5] T. Kühne, "Matters of (meta-) modeling," *Software and Systems Modeling*, vol. 5, no. 4, pp. 369–385, December 2006.

[6] H. Stachowiak, *Allgemeine Modelltheorie*. Wien: Springer, 1973.

[7] J. P. van Gigch, *System Design Modeling and Metamodeling*. New York, NY, USA: Plenum Press, 1991.

[8] R. Schuette and T. Rotthowe, "The guidelines of modeling: An approach to enhance the quality in information models," in *17th International Conference on Conceptual Modeling (ER'98)*, ser. Lecture Notes in Computer Science, vol. 1507. Springer, November16–19 1998, pp. 240–254.

[9] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos, "Identifying and measuring quality in a software requirements specification," in *First International Software Metrics Symposium*, May 1993, pp. 141–152.

[10] T. Gorschek and C. Wohlin, "Requirements abstraction model," *Requirements Engineering*, vol. 11, no. 1, pp. 79–101, 2005.

[11] J. Steel and J.-M. Jézéquel, "On model typing," *Software and Systems Modeling*, vol. 6, no. 4, pp. 401–413, 2007.

[12] O. I. Lindland, G. Sindre, and A. Sølvberg, "Understanding quality in conceptual modeling," *IEEE Software*, vol. 11, no. 2, pp. 42–49, 1994.

[13] D. L. Moody, "Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions," *Data & Knowledge Engineering*, vol. 55, no. 3, pp. 243–276, 2005.

[14] J. Verelst, "The influence of the level of abstraction on the evolvability of conceptual models of information systems," *Empirical Software Engineering*, vol. 10, no. 4, pp. 467–494, October 2005.

[15] L. Saitta and J.-D. Zucker, "Abstraction and complexity measures," in *Abstraction, Reformulation, and Approximation*, ser. Lecture Notes in Computer Science, vol. 4612. Springer, 2007, pp. 375–390.

[16] L. C. Briand, S. Morasca, and V. R. Basili, "Property-based software engineering measurement," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 68–86, 1996.

[17] T. DeMarco, *Controlling Software Projects: Management, Measurement, and Estimates*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1986.

[18] B. H. C. Cheng and J. M. Atlee, "Research directions in requirements engineering," in *FOSE '07: 2007 Future of Software Engineering*. IEEE Computer Society, 2007, pp. 285–303.

[19] M. Glinz, S. Berner, and S. Joos, "Object-oriented modeling with ADORA," *Information Systems*, vol. 27, no. 6, pp. 425–444, 2002.

[20] N. Habra, A. Abran, M. Lopez, and A. Sellami, "A framework for the design and verification of software measurement methods," *Journal of Systems and Software*, vol. 81, no. 5, pp. 633–648, 2008.

[21] B. Kitchenham, S. L. Pfleeger, and N. Fenton, "Towards a framework for software measurement validation," *IEEE Transactions on Software Engineering*, vol. 21, no. 12, pp. 929–944, 1995.

[22] J. Bézivin, "On the unification power of models," *Software and Systems Modeling*, vol. 4, no. 2, pp. 171–188, 2005.

[23] N. F. Schneidewind, "Methodology for validating software metrics," *IEEE Transactions on Software Engineering*, vol. 18, no. 5, pp. 410–422, 1992.