A Framework for Empirical Evaluation of Model Comprehensibility

Jorge Aranda, Neil Ernst, Jennifer Horkoff, and Steve Easterbrook
University of Toronto, Canada
{jaranda, nernst, jenhork, sme}@cs.toronto.edu

Abstract

If designers of modelling languages want their creations to be used in real software projects, the communication qualities of their languages need to be evaluated, and their proposals must evolve as a result of these evaluations. A key quality of communication artifacts is their comprehensibility. We present a flexible framework to evaluate the comprehensibility of model representations that is grounded on the underlying theory of the language to be evaluated, and on theoretical frameworks in cognitive science.

1. Introduction

Over the past decades, hundreds of conceptual modelling languages have been proposed as tools to understand and communicate software project information [14]. We have a wealth of notations at our disposal to represent almost any kind of information we wish, from machine states to stakeholder goals. Yet the use of these modelling languages in real software projects and their adoption rate by the software industry are still very low [7].

An important cause of this usage problem may be a lack of attention to the extent to which these languages enable effective communication among their users. Models have many uses, but one of the most prominent is serving as communication artifacts in software teams. In fact, if they have *one* purpose, for most languages, it is communicating ideas.

The effectiveness of software models depends on a number of communication *qualities* such as: Cost of production, comprehensibility, speed of 'decay' (loss of synchrony with the content it represents), and steepness of their learning curve. If a language is deficient in several of these qualities, then it does not matter whether it has a high expressive power or well-formalized semantics; it will not be used for communication purposes.

Considering models as communication artifacts raises an important issue. Even the simplest models of communication available [11] require a receiver to decode and assimilate the message for the communication instance to be successful. A communication event does not stop with the transmission of an encoded message. In practical terms, creating and sending a diagram to somebody may lead us to believe that we have communicated its information to that person; but if the diagram is not read, processed, and assimilated correctly by the receiver, the communication instance has failed.

For this reason, an essential quality of communication artifacts is their *comprehensibility*. Documents and diagrams that are cryptic, misleading, or vague will not serve their communication purpose. Therefore, it is important to bring comprehensibility, along with other communication qualities, to the forefront of the modelling language debate. Unfortunately, as we will discuss in Section 4, there have been very few careful empirical studies that evaluate the comprehensibility of software modeling languages. When it is considered at all, judgments about model comprehensibility are often very subjective and have little regard for empirical validity.

In this paper we present an empirical framework to evaluate model comprehensibility. The framework, presented as a sequence of steps and guidelines, is intended to guide evaluators to address the challenges of studying a construct as subtle and complex as comprehensibility. We assume that any researchers who apply it will have some empirical software engineering expertise, and access to expert modellers of the language of their choice.

2. The comprehensibility construct

2.1. Challenges to define the construct

The first challenge for evaluators of model comprehensibility is to define the meaning of the construct: it is an intuitive concept, but very difficult to define. The naive view ("Can I make sense of this document?") breaks down when we try to

operationalize it. To clarify it, we propose the use of the comprehensibility variables in Tables 1 and 2.

As can be seen in the tables, there are many variables to consider, and it may not be feasible to evaluate them all in a single empirical study. The choice of which of these should be addressed is up to the evaluator, though it is important to declare these decisions explicitly. We will return to these tables when discussing study hypotheses, in Section 3.

2.2. Challenges to evaluate comprehensibility empirically

Studying comprehensibility raises a number of challenges in addition to those inherent to all empirical work. We describe them in the following list:

Information equivalence: In practical terms, it is impossible to guarantee that two different representations transmit the same meaning to a human reader, even when their underlying conceptual content is the same (that is, when they have information equivalence [13]). This problem arises because our innate ability to handle qualitative information is notoriously difficult to operationalize. Figure 1 shows an example of this problem: A simple change in the layout of the nodes of a graph triggers different meanings in the reader. The problem is magnified when comparing documents in different notations. A class diagram is not designed to represent the same information as an entity-relationship diagram. Therefore, an evaluator needs to decide whether her comparisons will strive towards the ideal of information equivalence, at the risk or artificiality; or if she will use models with uneven information to achieve a comparison with greater realism.

Table 1 – Affected comprehensibility variables

Correctness of Understanding	The degree to which persons can answer questions about the representation correctly.
Time	Time required to understand the representation.
Confidence	Subjective confidence that people display regarding their own understanding of the representation.
Perceived difficulty	Subjective judgment that people display regarding the ease to obtain information through the representation.

Table 2 – Affecting comprehensibility variables

Type of task	The different tasks that readers perform with a representation are facilitated or hindered to varied degrees. Comprehensibility for information search, information retention, or functional tasks requiring the integration of information in the reader's mental model, lead to different evaluation results.
Language expertise	Previous expertise with the modelling language under study.
Domain expertise	Previous expertise with the domain being modelled.
Problem size	Size of the domain. Different modelling languages scale up with variable degrees of success.

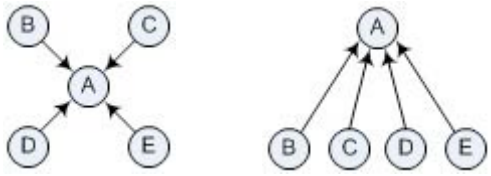

Figure 1 - The same graph may represent centrality or hierarchy

Accessibility of participants: If finding competent participants is a challenge in most software engineering studies, the problem is exacerbated when we require participants to be experts in one or several languages, given that this expertise is scarce both in industry and academia for most languages. Some workarounds for this problem are discussed in section 3.7.

Researcher bias and agenda: Since the evaluator of a notation is often its proponent, many evaluations suffer from researcher bias. This is evident in questionnaires that are trivially answered with the preferred model and unanswerable with another. Section 3.7 of this paper describes some safeguards we have developed against this sort of bias.

3. Evaluating comprehensibility

Our framework consists of a sequence of guidelines that may be followed in the order presented:

3.1. Select the modelling notation

Selecting the notation is an evident first step, but there are several detailed decisions to make in order to achieve precision. First, if the notation of choice has several versions, which of them is being studied? Does the study include language extensions? Will it be allowed to tweak the notation rules to better represent information (as often happens in practice), or will we adhere to a strict implementation of the rules, to the potential detriment of the readability of the model? It is essential to be clear about the version and conventions to be tested.

3.2. Articulate the underlying theory

Every modeling language is proposed under the assumption that it will be useful for particular reasons and situations. A notation might be proposed as a means of communication between analysts and developers, or between analysts and users. It may assume a logical skill set or familiarity with the domain on the part of its readers. It may require its models to have some complementary documentation, or may be designed to be meaningful on its own. Sometimes these assumptions are stated explicitly by the notation's designers; sometimes they arise through practice. We call the set of assumptions behind a language its *underlying theory*. Although this underlying theory is usually concerned with more than communication, here we focus on its communication aspect: the nature of the ideas the notation represents, and the context in which it is used.

Ideally, the underlying theory of a language will be obtained by studying its literature. Unfortunately, many language proposals are not explicit in this regard. They are frequently offered without considerations for the target users, domains, problem sizes, and required domain expertise of their readers. If this is the case, one should try to extract the underlying theory from the language as it is used in practice. If it is popular, one might detect patterns from its users and the domains in which it is applied. For example, we could observe that a language is used mostly as a communication mechanism among analysts, or as a validation tool between analysts and stakeholders. These observations would lead us to assert, under the lack of an explicit theory, that the language is intended to be used for those types of people, in those contexts.

If there is no community of users and no explicit underlying theory, then one has no choice but to fill in the blanks of the underlying theory and to state so directly. The study will run the risk of being countered with the argument that the underlying theory declared is incorrect (especially if the results of the study do not favour the notation). But this is not a flaw of the evaluation; it is a flaw –a major, though frequent flaw– of the language proposal that failed to offer this information explicitly to its audience.

3.3. Formulate the claims of the notation

Once we have identified the underlying theory of the language, we need to re-express the theory as a set of claims regarding comprehension. In extracting these from the underlying theory, we must be careful not to lose its spirit along the way. Later in the process, these claims will become the study hypotheses.

3.4. Choose a control

An evaluation should have a baseline for comparison – otherwise we cannot tell whether scores are caused purely by the characteristics of our notation, or by the inherent level of complexity of the model domains. We offer the three following guidelines:

First, the baseline should be a sensible alternative to perform the same tasks that the notation assists. It need not be diagrammatic; it is reasonable, for example, to compare a sequence diagram with a sentential description of the same scenario.

Second, if the notation under evaluation is an extension to a language, one might think that a sensible alternative for a baseline is a standard version of the same language. This decision should be taken carefully. Extensions are frequently designed to cover gaps in ideas that the original language cannot express; in that case the original language is not a sensible alternative.

Third, in cases when there is no clear baseline, the most suitable alternative is, simply, a natural language version of the same information, written in a style and tone similar to the one that potential readers would have access to. Some of the more esoteric notations only have a natural language counterpart.

3.5. Turn the claims into hypotheses

Once we have chosen a control we can turn the claims of the notation into testable hypotheses, with the following considerations:

First, although studying the overall comprehensibility of a model is important, from a language evolution perspective it is even more relevant to discover *which* elements of a notation work well and which do not. For instance, discovering that the meaning of a class diagram was only partially grasped is convenient; discovering that aggregations were obvious to readers but other associations were not is even better. We should aim to design hypotheses that cover both the abstract comprehension effect and the specific, concrete elements that the notation represents.

Second, for our purposes, syntax is not sugar. Syntactic refinements may yield far greater benefits than semantic modifications to the language. Evaluators should not shy away from evaluating the syntax of a notation and the elements and icons that communicate the semantics of the models. Similarly, if a language demands to be used with a specific tool, it should also be included in the evaluation.

Finally, it is desirable to generate hypotheses that cover most of the variables mentioned in Tables 1 and

2. In any case, we must define the expected domain and language expertise of participants and the size of the problem they will work with.

3.6. Inform the hypotheses

Software engineering is not the first discipline to study the effects of representations in human performance, and we should bring the insights of other research areas to our evaluations. Two theoretical perspectives are of particular relevance:

3.6.1. External Cognition. A branch of Cognitive Science, external cognition treats humans and the artifacts they use to solve problems as a single cognitive entity. Artifacts are part of our problem-solving resources, and they enhance and augment our capabilities. There are several ways in which representations can improve our reasoning [3]. The following are extracted from Scaife and Rogers [16]:

Offloading: Representations, such as models, can reduce a person's cognitive effort by putting knowledge in the world, rather than in the head. The less data we need to keep in our memory, and the fewer rules we need to process them, the better.

Re-representation: Some representations, by virtue of their cognitive fit to the problems they are used for, make problem solving easier. A classic example is performing a multiplication with Arabic (43 x 10) versus Roman numerals (XLIII x X).

Graphical constraining: If a diagram constrains the number of inferences we can make, it allows us to spend our cognitive power on them more effectively.

3.6.2. Cognitive Dimensions (CD) Framework. The Cognitive Dimensions framework [4], which enjoys some popularity in the HCI field, enumerates dimensions of tool use that are relevant from a cognitive perspective. CD proponents have designed a questionnaire [5] to assess the quality of tools and to guide design decisions. A sample of dimensions follows (the full framework has more than a dozen):

Visibility: Ability to view components easily.

Hidden dependencies: Are important links between entities visible? Does the tool user have to go through complicated processes to uncover these dependencies?

Role-expressiveness: Easily inferring the purpose of an entity.

3.6.3. Other relevant perspectives. We have chosen two theoretical perspectives, but these are not the only alternatives. A model evaluation proposal by Wand and Gemino [9] asks evaluators to perform ontological assessments of the model's expressive power, and to analyze them with a range of cognitive theories.

These perspectives add value to a comprehensibility study by offering systematic ways of understanding how models aid cognition. But for each additional perspective an evaluator considers, the practicality of performing evaluations decreases. We adopted external cognition and the cognitive dimensions framework because we believe they have a particularly high relevance to comprehensibility considerations.

3.6.4. How to inform the hypotheses? Once evaluators become familiarized with the theoretical perspectives they should inform the study hypotheses with insights from them. By this we mean to examine the hypotheses through the "lens" of the theory, in order to detect gaps in the hypotheses under evaluation.

For example, for each cognitive dimension we can ask whether the hypotheses we have previously generated test for the benefits of that particular dimension. If we find a relevant gap, we may decide to modify or add more hypotheses to our list. *Role-expressiveness*, for instance, may be evaluated by asking readers with little experience with a modelling language what each of its graphical elements means.

If there are important gaps between the hypotheses and relevant cognitive perspectives, the evaluator could be missing some comprehension-related benefits, and may wish to refine the hypotheses to include them.

3.7. Design and execute the study

We will not describe general empirical methods [19], but there are particularities concerning the design of comprehensibility studies that we should mention.

To evaluate a modelling language empirically, one *must* test particular instances of that modelling language. It is impossible to empirically assess, for example, the comprehensibility of sequence diagrams in general; one must assess the comprehensibility of *particular* sequence diagrams and, by induction, generalize to the language as a whole. However, evaluating particular models poses delicate problems. Here are some guidelines to consider:

Natural domains: The choice of domains to model should be natural for the language under study. For instance, a sequence diagram excels when displaying series of events and method calls, not decision-making algorithms (even though they may be used for the latter). Some studies evaluate notations out of their natural domains (for example, pseudocode of how to cook), and their results are questionable for this reason.

Familiar domains: A common challenge of empirical software engineering studies is getting enough qualified participants. Some hypotheses may

require high levels of background knowledge, making recruitment even harder than usual. A familiar domain should be chosen to improve the chances of getting enough participants. The exception to this guideline is if the evaluator wants to test the notation as a pedagogical tool: that is, to analyze whether novices understand the complexities of an unknown domain exclusively through the model under study.

Participant role: As mentioned before, the study should be explicit on the types of participants (stakeholders, analysts, developers, maintainers, or others) required, and on the levels of expertise they must possess. These criteria should be central to the recruitment process.

Expert modelers: The models should be prepared by notation experts, though this is a contentious issue. On one hand, real model readers routinely work with models produced by people at varying degrees of language expertise, and it is important to evaluate models at all of them. On the other hand, if models come from people with dubious expertise, poor results may be adjudicated to the flawed model instead of inherent notation problems. We prefer to avoid this potential bias. Additionally, modelers should not be part of the research team if possible, to avoid contaminating the models with the researchers' biases.

Number of domains: If resources allow, the study should test several models, from several domains, to avoid a mono-operation bias.

Questionnaire: Whenever feasible, the study should collect data for each question on several variables: c*orrectness* (did the participant give the right answer?), *confidence* (certainty of the participant in his answers), *perceived difficulty* (to respond the question), *source of answer* (did the answer come from the model, from previous knowledge, or from assumptions?), and *time to respond*.

3.8. Improve these guidelines

Since we assume all modelling languages, as all tools, are perfectible, we would be at fault if we did not assume the perfectibility of our framework as well. We have modified it, and we expect to continue modifying it, through its repeated application to multiple empirical studies. We hope the community will contribute in a similar fashion.

4. Related work

The topic of model comprehensibility has been previously addressed, with varying degrees of empirical rigour. Two early studies of the field were those of Ramsey et al. [15], who reported that pseudocode and flowcharts yield no difference in comprehension, and Scanlan [17], who countered that flowcharts actually outperform pseudocode. Neither study offers any theoretical foundation to ground the evaluation, and their methodological problems cast serious doubt on their validity.

More recently there have been significant steps to overcome the challenge of assessing comprehensibility. Progress has been twofold: the theoretical grounding of comprehensibility studies has been laid out more clearly, and the soundness of the empirical studies has increased notably. Agarwal et al. [1] based their comprehensibility assessment of models on the notion of *cognitive fit* (reflected in our insistence of basing evaluations on the underlying theory of the modelling language), and on the concept of information equivalence. Kim et al. [12] also drew from the information equivalence concept, while Gemino and Wand [10] advocated for ontological analyses and the use of the cognitive theory of multimedia learning to drive their evaluations. However, not every recent study bases its evaluation on theory. Finney et al. [8], and Zimmerman et al. [20], among others, did not address the issue of how to measure their constructs properly, if at all.

The idea that the evaluation of models should be grounded on their function is most notably present in a recent study [2] of UML documentation for maintenance tasks. Participant background, training, and type of tasks were also considered in a recent evaluation of formality in UML by Briand et al. [6].

We are aware of one other framework to evaluate conceptual models, by Gemino and Wand [9]. Their framework "is based on the notion that modeling techniques should be compared via their underlying grammars", although these "grammars" cannot truly be evaluated empirically. It defines two dimensions for evaluation: affecting and affected variables. They do not focus on comprehensibility, on the theoretical grounding of evaluations, or on the particular challenges of this type of empirical study. For these reasons we believe our framework complements and augments theirs.

5. Conclusions and future work

We have presented an empirical framework to evaluate the comprehensibility of model representations. The framework can be applied by the modelling community, provided they have empirical software engineering expertise in their teams.

This framework arose through the discussions of our team when designing a series of evaluations, with the goal of ensuring proper methodological and

theoretical foundations. We initially wanted to be able to apply it systematically –that is, to refine it into a benchmark [18], rather than a set of guidelines for *ad hoc* tests. We were soon convinced this was not possible. There are too many subtle distinctions between languages and too many differences of underlying theories for such a systematized solution to succeed. But we believe this framework provides a helpful guide to lead our theoretical analyses, our study designs, and our sense-making of the available modelling language literature.

There are far too many languages for us to evaluate, and each evaluation requires a considerable investment. We will perform evaluations on several languages based on their apparent promise and popularity. We expect our framework to be modified by each of our planned evaluations.

An appealing quality of this framework is that it is not restricted to diagrammatic representations. In the near future, we intend to evaluate sentential types of representations commonly used in software development, such as specifications and user stories.

The framework can also be modified to evaluate communication qualities other than comprehensibility. Analyzing production cost, for example, seems to be a promising area of research. This requires adapting Step 6 of the framework, described in Section 3.6, to suit the appropriate theoretical perspectives, but the rest of it should be as effective for evaluating other communication qualities as it is for comprehensibility.

6. References

[1] Agarwal, R., De, P., and Sinha, A.P. Comprehending Object and Process Models: An Empirical Study. *IEEE Transactions on Software Engineering*, 25, 4, 1999.

[2] Arisholm, E., Briand, L.C., Hove, S.E., and Labiche, Y. The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. *IEEE Transactions on Software Engineering*, 32, 6. June 2006.

[3] Bauer, M.I., and Johnson-Laird, P.N. How Diagrams can Improve Reasoning. *Psych. Science*, 4, 6. November 1993.

[4] Blackwell, A., and Green, T. Notational Systems—The Cognitive Dimensions of Notations Framework. In: *Carroll, J. (Ed) HCI: Models, Theories and Frameworks*, Morgan Kaufmann, 2003.

[5] Blackwell, A., and Green, T. A Cognitive Dimensions Questionnaire. 2000. Downloaded from the web at: http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDquestionnaire.pdf

[6] Briand, L.C., Labiche, Y., Di Penta, M., and Yan-Bondoc, H. An Experimental Investigation of Formality in UML-Based Development. *IEEE Transactions on Software Engineering*, 31, 10. October 2005.

[7] Dobing, B., and Parsons, J. How the UML is Used. *Communications of the ACM*, 49(5), May 2006.

[8] Finney, K., Fenton, N., and Fedorec, A. Effects of structure on the comprehensibility of formal specifications. *In IEE Proc-Softw.*, 146, 4, August 1999.

[9] Gemino, A., and Wand, Y. A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering*, 9, 248-260, 2004.

[10] Gemino, A., and Wand, Y. Evaluating Modeling Techniques Based on Models of Learning. *Communications of the ACM.* 2003.

[11] Hargie, O.D.W. (Ed.) *The Handbook of Communication Skills*. Routledge, 1997.

[12] Kim, J., Hahn, J., and Hahn, H. How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning. *Information Systems Research*, 11, 3, 284-303, 2000.

[13] Larkin, J.H., and Simon, H.A. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* 11, 65-99, 1987.

[14] Mylopoulos, J. Information Modeling in the Time of the Revolution. *Information Systems*, 23 (3-4), June 1998.

[15] Ramsey, H.R., Atwood, M.E., and Van Doren, J.R. Flowcharts Versus Program Design Languages: An Experimental Comparison. *Comm. of the ACM*, 26, 1983.

[16] Scaife, M., and Rogers, Y. External cognition: how do graphical representations work? *Int. J. Human-Computer Studies*, 45, 185-213, 1996.

[17] Scanlan, D.A. Structured Flowcharts Outperform Pseudocode: An Experimental Comparison. *IEEE Software,* September, 1989.

[18] Sim, S., Easterbrook, S., and Holt, R. Using Benchmarks to Advance Research: A Challenge to Software Engineering. *25th Intl. Conf. on Software Engineering (ICSE'03)*, 2003.

[19] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A. *Experimentation in Software Engineering: An Introduction.* Kluwer, 2000.

[20] Zimmerman, M.K., Lundqvist, K., and Leveson, N. Investigating the Readability of State-Based Formal Requirements Specification Languages. *In: Proceedings of the Intl. Conf. on Software Engineering (ICSE'02)*, 2002.