Assessing the Power of A Visual Notation - Preliminary Contemplations on Designing a Test -

Dominik Stein and Stefan Hanenberg

Universität Duisburg-Essen
{ dominik.stein, stefan.hanenberg }@icb.uni-due.de

Abstract. This paper reports on preliminary thoughts which have been conducted in designing an empirical experiment to assess the comprehensibility of a visual notation in comparison to a textual notation. The paper sketches shortly how a corresponding hypothesis could be developed. Furthermore, it presents several recommendations that aim at the reduction of confounding effects. It is believed that these recommendations are applicable to other experiments in the domain of MDE, too. Finally, the paper reports on initial experiences that have been made while formulating test questions.

1 Introduction

Although modeling does not imply visualization, people often consider the visual representation of models to be a key characteristic of modeling. One reason to this could be that modeling techniques such as State Machines or Petri-Nets are often taught and explained with help of circles and arrows rather than in terms of mathematical sets and functions. Apart from that, other kinds of modeling, e.g. data modeling with help of Entity-Relationship-Diagrams, make heavy use of visual representations, although the same concepts could be specified in a purely textual manner, too.

However, let alone the impression that visual representations are considered very appealing by a broad range of developers, customers, maintainers, students, etc., a scientific question would be if visual representations actual yield any extra benefit to software development, maintenance, or teaching, etc.

Driven by a personal belief of the authors that this extra benefit exists, this paper reports on preliminary thoughts which have been conducted in designing an empirical experiment. The goal of this empirical experiment is to assess (such a "soft" property as) the "comprehensibility" of a visual notation in comparison to a textual notation.

This paper does not formulate a concrete hypothesis. Instead, it conducts general contemplation about hypotheses that are concerned with the evaluation of "comprehensibility". In particular, the paper presents several recommendations that aim at the reduction of confounding effects while running the test. It is suggested that these recommendations should be obeyed in other experiments in the domain of MDE, too. Furthermore, the paper reports on experiences that have been made while formulating the test questions for a test on comprehensibility.

The paper is structured as follows: In section 2, the process to define a hypothesis is outlined. In sections 3 and 4, a couple of considerations are presented in order to reduce confounding effects. In section 5, problems are presented which have been encountered while formulating test questions. Section 6 presents some related work. And section 7 concludes the paper.

2 Defining the Goal of the Experiment, and What to Measure?

When designing an controlled experiment, everything is subordinate to the overall assumption, or hypothesis, that is going to be tested. Usually, the development of the test will require to repeatedly reformulate (refine) the hypothesis since the initial hypothesis turned out not to be (as easily) testable (as presumed). (A possible reason for this could be, for example, that it is overly hard, or impossible, to reduce the impact of confounding effects or to find suitable questions; cf. sections 3, 4 and 5.)

2.1 Experiment Definition

A fist step could be to define the experiment in general. When comparing visual vs. textual notations, this could be done as follows (using the experiment definition template suggested by [10]):

The goal of the study is to analyze visual and textual program specifications (i.e. diagrams versus code), with the purpose of evaluating their effect on the "comprehensibility" of the information shown. The quality focus is the perception speed and completeness and correctness with which all relevant information is apprehended. The perspective is that of teachers and program managers, who would like to know the benefit that visual notations can bring to their work (i.e. teaching students in computer science or developing software). The context of the experiment is made up of artificial/sample code snippets and their corresponding diagrams (= objects) as well as undergraduate and graduate students (= subjects).

2.2 Hypothesis Formulation

According to [4], a scientific hypothesis meets the following three criteria:

- A hypothesis must be a "for-all" (or rather a "for-all-meeting-certain-criteria") statement. This means in particular that the hypothesis must be true for more than a singular entity or situation.
- A hypothesis must be (able to be reformulated as) a conditional clause (of the form "whenever A is true/false, this means that B is (also) true/false").
- A hypothesis must be falsifiable. That means that, in principle, it must be able to find an entity or situation in which the hypothesis is *not* true.

Furthermore, for practical reasons, [1, 5] suggest to base the hypothesis on observable data. That is, in the (possibly reformulated) conditional clause, the value of one observable data (called "the dependent variable") must be specified to depend on the value of one other observable data (called "the independent variable") in a consistent way. The hypothesis is falsified if at least one example can be found where this dependency is not satisfied.

A starting point to find a hypothesis for the experiment outlined in section 2.1 could be the following:

When investigating program specifications, a visual representation X (as compared to a textual representation Y) ***significantly facilitates*** *comprehension of information Z.*

Following the aforementioned criteria, the above hypothesis is a scientific hypothesis because it can be rephrased as "whenever a program specification is represented using a visual notation X, it is easier to comprehend (with respect to information Z) than an equivalent representation using a textual notation Y". In this statement, the possible values (treatments) of the independent variable (factor) are "visual/not visual" and the possible values of the dependent variable are "easier to comprehend/not easier to comprehend". The claimed dependency would be "visual $\rightarrow$ easier to comprehend". The statement could be falsified by showing that visual notation X is not easier to comprehend than textual notation Y (with respect to information Z).

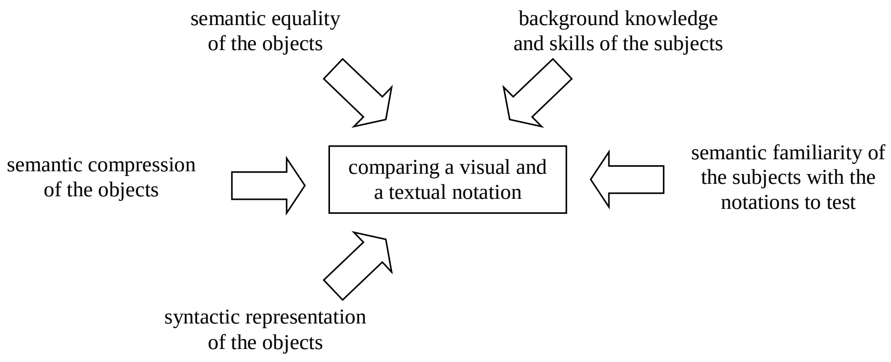

semantic equality of the objects
background knowledge and skills of the subjects
semantic compression of the objects
comparing a visual and a textual notation
semantic familiarity of the subjects with the notations to test
syntactic representation of the objects

Fig. 1. Confounding impacts.

2.3 Variable Selection

Turning a the preliminary idea of a hypothesis into a testable hypothesis which is thoroughly rooted on objectively observable data is a challenging task in developing an empirical test. For example, since comprehensibility by itself is difficult to observe, another variable must be found whose values are considered to inherently depend on the level of comprehension of a tester. A commonly accepted variable measuring the level of comprehension, for example, is "correctness", i.e. the number of correct answers[1] given to a (test) questions (cf. [8, 7, 6]). However, as pointed out by [7], correctness is only one facet of comprehensibility. Another variable is "comprehension speed", e.g. the number of seconds that the subjects looked at the object (or maybe even "easy to remember", i.e. the number of times that the subjects

[1] if the correct answer consists of multiple elements, it could be some mean of precision and recall [2] (cf. [6]).

looked at the objects; cf. [7]). The inherent effect of the variable that is of interest on the variable that is measured must be substantially elucidated (and defended) in the discussion on the (construct) validity of the test.

The only factor (independent variable) in the experiment would be "kind of presentation" with the treatments (values) {visual, textual}.

One of the big challenges when investigating the casual dependencies between the (dependent and independent) variables is to reduce confounding impacts (see Fig. 1) as much as possible, and thus to maximize the validity of the experiment (cf. [10]). Otherwise, the "true" dependency could possibly be neutralized (at least, in parts), or might even be turned into its reciprocal direction (in the worst case).

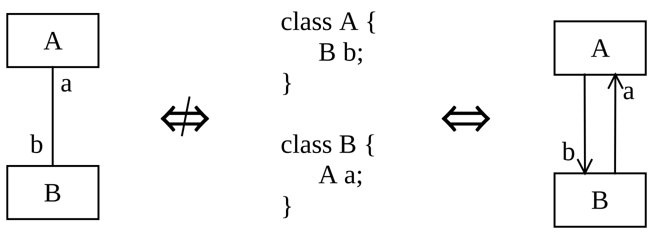
In the following sections, some means are presented which should be taken in order to improve the validity of an experiment comparing a visual and a textual notation. The authors believe that these means are general enough to be applied to other evaluating experiments in the domain of MDE approaches, too.

3 Preparing Objects – Ensuring Construct Validity (I)

Construct validity refers "to the extent to which the experiment setting actually reflects the construct under study" [10]. In particular, this means to ensure that the objects of the experiment which are given to the subjects in order to perform the tests represent the cause well (i.e. a visual vs. a textual representation, in this case).

3.1 Semantic Equality

One obvious, yet carefully to ensure, requirement is to compare (visual and textual) representations that have equal semantics, only. It would be illegal and meaningless to compare any two representations with different semantics.



```
class A {
    B b;
}

class B {
    A a;
}
```

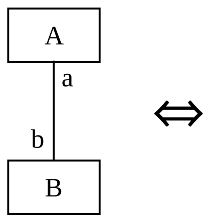
Fig. 2. Ensure semantic equality.

Fig. 2 shows an example. It would be illegal to compare the visual representation on the left with the textual representation in the middle since they mean different things. The bidirectional association between classes `A` and `B` in the UML model in the left of Fig. 2 denotes that two instances of class `A` and `B` are related to each other such that the instance of class `A` can navigate to the instance of class `B` via property `b`, while at the same time the instance of class `B` can navigate to the instance of class `A` via property `a` (meaning `a = a.b.a` is always true). The Java program code in the middle of Fig. 2, however, does not imply that an instance of class `A` which is associated with an instance of class `B` (via its property `b`) is the same instance which that associated instance of class `B` can navigate to via its property `a` (meaning `a = a.b.a` does not need to be true).

Hence, in an empirical test comparing the performance[2] of visual vs. textual representations of associations, it would be more appropriate (in fact, obligatory) to compare the textual representation in the middle of Fig. 2 with the visual representation in the right of Fig. 2. Now, the semantic meaning of one notation is equally represented in the other notation, and comparing the results of their individual performance is valid[3].

3.2 Equal Degree of Compression

Apart from semantic equality, the expressions being compared need to be expressed at an equal degree of compression (here, the degree of compression shall refer to the degree with which semantic information is condensed into one language construct). Otherwise, "better" performance of one notation could be induced by the fact that one notation uses a "higher" compression (e.g. one language construct of that notation conveys the same semantic information than four language constructs of the other notation) rather than that it uses a "better" representation.



```
class A {
    B b;
    B getB() { return b; }
    void setB(B b) { this.b = b; b.a = this; }
}

class B {
    A a;
    A getA() { return a; }
    void setA(A a) { this.a = a; a.b = this; }
}
```

Fig. 3. Do not test expressions of unequal degree of compression.

Fig. 3 gives an example. Other than in Fig. 2, the Java code now contains extra lines which states that an instance of class `A` which is associated with an instance of class `B` (via its property `b`) must be the same instance to which that associated instance of class `B` can navigate via its property `a` (meaning `a = a.b.a` is always true). Hence, the Java expression in the right of Fig. 3 now equates to the semantics of the UML expression in the left of Fig. 3.

[2] in this paper, "performance" refers to "the notation's ability to be read and understood" rather than computation speed.

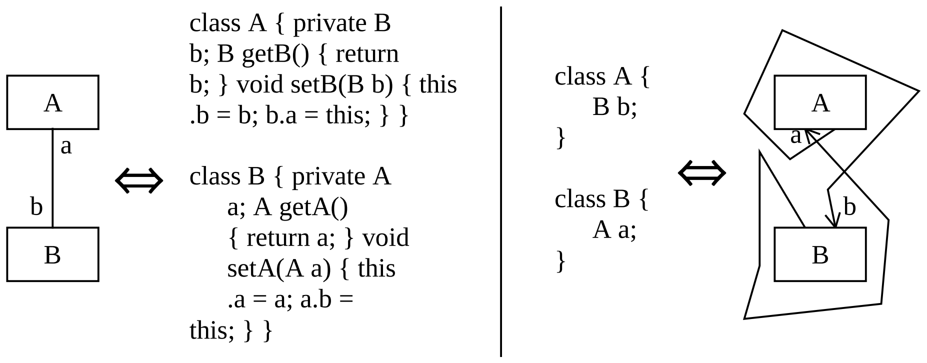
[3] note that asserting the semantic equality of two notations is not trivial. For example, there is no general agreement on how a UML class diagram should be transformed into Java code.

If – in a test – the UML expression should actually yield "better" results than the Java expression now, it is unclear (and highly disputable) whether the "better" performance is due to the visual representation or due to the higher degree of compression (i.e. the fact that we need to read and understand four method definitions in the Java code as compared to just one association in the UML diagram).

3.3 Presenting Objects

Apart from equal semantics and equal degree of compression, the expressions have to be appropriately formatted, each to its cleanest and clearest extent. This is because the authors estimate that disadvantageous formatting of expressions could have a negative impact on the test outcome, whereas advantageous formatting of expressions could improve the test results.

Fig. 4 gives an example. In the left part of Fig. 4, the Java code has been formatted in a way which is tedious to read. In the right part of Fig. 4, the UML representation has been formatted disadvantageously. With expressions formatted like this, it is assumed that the respective notation is condemned to fail in the performance test.



```
class A { private B
b; B getB() { return
b; } void setB(B b) { this
.b = b; b.a = this; } }

class B { private A
     a; A getA()
     { return a; } void
     setA(A a) { this
     .a = a; a.b =
this; } }
```

```
class A {
     B b;
}

class B {
     A a;
}
```

Fig. 4. Format expressions to their cleanest and clearest extent.

Unfortunately, there usually is no (known) optimal solution for the formatting task. Therefore, expressions should be formatted clearly and consistently following some strict and predefined guidelines (e.g. some formatting guidelines such as the [9]). It is important to keep in mind, though, that even though uniform guidelines are used to format the expressions, the effects of those formatting guidelines on the test outcomes are unclear. Moreover, the effects may even be different for each notation. Consequently, the (unknown) impact of formatting guidelines on the test results needs to be respected in the discussion of the (construct) validity of the test.

Likewise, syntactic sugar is to be avoided. That means, all means that are not related to the semantics of the underlying notation, such as syntax highlighting in textual expressions, or different text formats and different line widths in visual expressions, should not be used. Syntactic sugar (fonts, line width, colors, etc.) are likely to draw the attention of the testers to different parts of the expressions and thus may confound the pure comparison between their visual and textual representation.

Evaluating the impacts of formatting, fonts, line width, and colors on the comprehensibility of a notation is an interesting test of its own. However, that test should focus on the comparison of different style guidelines for *one* notation rather than on the comparison of (different) guidelines for different notations.

4 Preparing Subjects – Ensuring Internal Validity

To ensure internal validity, it must be ensured that a relationship between a treatment and an outcome results from a causal relationship between those two, rather than from a factor which has not been controlled or has not been measured (cf. [10]). In particular this means how to "treat", select, and distribute the subjects such that no coincidental unbalance exists between one group of testers and another.

4.1 Semantic Familiarity

The imperative necessity of comparing semantically equivalent "expressions" (see section 3.1) is complemented with the necessity that testers are equally trained in, and familiar with, both notations. Otherwise, i.e. if the testers of one notations are more experienced with their notation than the testers of the other notation with their notation, a "better" test result of the former notation could be induced by the fact that its testers have greater experience in using/reading it rather than by the fact that it is actually "better" (in whatsoever way). This is particularly probable whenever the performance of *new* notations shall be evaluated in contrast to *existing* ones.

One way to control the knowledge of the tested notations is to look for testers that are not familiar with both notations, and have them take a course in which they learn the notations to test. This approach seems particularly practicable in academia – even though the test results will usually assert the performance of "beginners", and thus make extrapolation to the performance of "advanced" software developers in industrial settings difficult (which does not mean that assessing the benefits of visual notations for "beginners" isn't worthwhile and interesting). This problem represents a threat to the external validity of the experiment (cf. [10]).

The goal of teaching the notations to novices is to ensure that the testers of each notation attain similar knowledge and skill with their notation. The challenge here is to defined what it means that testers are "equally familiar" (i.e. equally knowing and skilled) with their notations. It also needs to be investigated how the knowledge and skills of an individual tester with his/her notation can be actually assessed (so that we can decide afterwards whether or not "equal familiarity" has been reached). Another challenge is how "equal familiarity" can be achieved by a teaching course in a timely and didactically appropriate manner (e.g., what is to be done if a particular group of testers encounters unforeseen comprehension problems with their notation).

The knowledge and skill test could occur prior to the actual performance test, or intermingled with the performance test (in the latter case, some questions test the knowledge and the skills of the testers, while other questions test the performance of the notations). If the knowledge and skill test reveals that the semantic familiarity of the testers with their notation is extremely unbalanced (between the groups of testers), the test outcome must be considered meaningless.

5 Measuring Outcomes – Ensuring Construct Validity (II)

Once the hypothesis is sufficiently clear, the next challenging step is to formulate questions that are suitable to test the hypothesis and to find a test format that is suitable to poll the required data. This is another facet of construct validity, according to which the outcome of the test needs to represent the effects well (cf. [10]).

In this section, considerations and experiences are presented that have been made in designing a test evaluating the comprehensibility of a visual notation.

5.1 Test Format, and How to Measure?

Multiple Choice tests (when carefully designed; cf. [3]) are considered to be a good and reliable way to test the knowledge of a person, in particularly in comparison to simple True/False tests. Hence, Multiple Choice tests would have a higher construct validity with respect to the correctness of comprehension than True/False tests. A question format with free answer capabilities would be more realistic (and thus would increase the external validity of the experiment; cf. [10]). However, such short-answer test is much more laborious because it requires manual post-processing in order to detect typos and/or semantically equivalent answers.

When it comes to measuring the response time, it is important to discriminate between the time to find the answer in the expression and the time to understand the question. This is because if testers need 30 sec. to understand a question and then 10 sec. to find the answer in the textual expression and just 5 sec. to find the answer in the visual expression, it makes a difference whether 40 sec. are compared to 35 sec., or 10 sec. to 5 sec. Not to discriminate between the time to find an answer and the time to understand a question is only valid, if the ratio is reciprocal, i.e. if the time to understand a question is negligible short in comparison to the time to find the answer.

If the test outcome consists of more than one data, it is a big challenge to define how the outcomes can be combined in order to obtain a meaningful interpretation. In this case, for example, it needs to be decided how "correctness of answers" and "response time" can be combined to indicate a "level of comprehension". One option would be to disregard all incorrect answers, and consider the response time of correct answers, only.

5.2 Volatile (Time) Measurements – Problems of A First Test Run

Preliminary and repeated test runs of a test evaluating simple analysis of one textual expression[4] (with the same person) have shown that the measured time needed to answer the question (exclusive of the time needed to understand the question; cf. section 5.1) is rather short (in average ~10 sec.) and varies tremendously (3 sec. to 30+ sec., even for same or similar questions!). It seems as if the measured time is heavily confounded by some external factor (maybe slight losses of concentration). This is problematic because due to the short (average) response time, even the slightest disturbance (of about 1 sec.) could confound the measured (average) time significantly (e.g. by one tenth, in this case).

[4] in another case than association relationships

Another problem was to strictly discriminate between the time to find the answer in the expression and the time to understand the question (which, again, was essential due to the short (averaged) response time). The testers were required to explicitly flip to the expression once they have carefully read (and understood) the question (which was shown first). As it turned out, however, testers sometimes realized that they have not fully understood the question after they have already flipped to the expression. As a result, the measured response time was partly confounded.

It is currently being investigated how the problem of high variation in measurements can be tackled. One option would be to pose questions that are more difficult to answer, and thus takes more time. This will only work, though, if the confounding effects do not grow proportionally. Another option would be to repeat the test countless times (with the same person and similar questions) in order to get a more reliable average response time. A big problem of this approach is to ensure that the testers will not benefit from learning effects in the repeated tests.

A promising solution to properly discriminate between the time to find the answer in the expression and the time to understand the question has been found in [7].

6 Related Work

In 1977, Shneiderman et al. [8] have conducted a small empirical experiment that tested the capabilities of flow charts with respect to comprehensibility, error detection, and modification in comparison to pseudo-code. Their outcome was that – statistically – the benefits of flow charts was not significant. Shneiderman et al. did not measure time, though.

This was determined to be inevitable by Scanlan [7]. Scanlan formulated five hypotheses (e.g. "structured flow charts are faster to comprehend", "structured flow charts reduce misconceptions", to name just the two which are closely related to this paper). Scanlan's test design is very interesting: Scanlan separated comprehension (and response) time of the question from comprehension time of the expression. To do so, testers could *either* look at the question *or* look at the expression (an algorithm, in this case). This is an interesting solution for the aforementioned problem of separating comprehension time and response time (see section 5.1). Scalan's outcome was that structured flow charts are beneficial.

7 Conclusion

This paper has presented preliminary thoughts which have been conducted in designing an empirical experiment to assess the comprehensibility of visual notations in comparison to textual notations. The paper has discussed shortly how a corresponding hypothesis could be developed. Furthermore, it has presented several recommendations that aim at the reduction of disturbances in the measured data, which are considered to be helpful for other experiments in the domain of MDE, too. Finally, the paper has reported on initial experiences that have been made while formulating the test questions.

It needs to be emphasized that this paper presents preliminary considerations rather than sustainable outcomes. On the contrary, each of the presented contemplations could be subject of an empirical evaluation of itself (e.g. whether or not advantageous formatting really has an positive effect on comprehensibility). Also, decisions need to be made about how to execute the test (e.g. how textual and visual expressions are shown to the testers, if they can use zooming or layouting functions, etc.) . The authors plan to pursue the considerations presented here and, ultimately, come up with a test design. Getting there will require many (self-)tests before finally a test design will be found that is capable to assess the specified hypothesis reliably.

Acknowledgement

The authors thank the anonymous reviewers for their patients with the tentativeness of these contemplations and for their productive comments which have helped to further advance the test design.

References

[1] Bortz, J., Döring, N., Forschungsmethoden und Evaluation für Sozialwissenschaftler (Research Methods and Evaluation for Social Scientist), Springer, 1995
[2] Frakes, W.B., Baeza-Yates, R., Information Retrieval: Data Structures and Algorithms, Prentice-Hall, 1992
[3] Krebs, R., Die wichtigsten Regeln zum Verfassen guter Multiple-Choice Fragen (Most Important Rules for Writing Good Multiple-Choice Questions), IAWF, Bern, 1997
[4] Popper, K., Logik der Forschung, 1934 (The Logic of Scientific Discovery, 1959)
[5] Prechelt, L., Kontrollierte Experimente in der Softwaretechnik (Controlled Experiments in Software Engineering), Springer, 2001
[6] Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M., The Role of Experience and Ability in Comprehension Tasks supported by UML Stereotypes, Proc. of ICSE'07, IEEE, pp. 375-384
[7] Scanlan, D.A., Structured Flowcharts Outperform Pseudocode: An Experimental Comparison, IEEE Software, Vol. 6(5), September 1989, pp. 28-36
[8] Shneiderman, B., Mayer, R., McKay, D., Heller, P., Experimental investigations of the utility of detailed flowcharts in programming, Communications of the ACM, Vol. 20(6), 1977, pp. 373-381
[9] Sun, Code Conventions for the Java Programming Language, April 20, 1999, http://java.sun.com/docs/codeconv/
[10] Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., Wesslen, A., Experimentation in Software Engineering - An Introduction, Kluwer, 2000