A Practical Guide to Testing the Understandability of Notations

Susanne Patig[1,2]

1 Otto-von-Guericke-University of Magdeburg, FIN-ITI, Universitätsplatz 2, D-39106 Magdeburg, Germany
2 University of Bern, IWI, Engehaldenstrasse 8, CH-3012 Bern, Switzerland

patig@iti.cs.uni-magdeburg.de, susanne.patig@iwi.unibe.ch

Abstract

Model-driven development is the process of creating models of a software system and transforming them into source code. Since the stepwise transformations can be done automatically or by hand, the notations of the models should be both precise and understandable. This is especially important if the software system is developed by a large, international team where the persons who model differ from the ones who implement the source code based on the models' content. Understandability and precision can be experimentally tested. This paper presents a guideline for planning and conducting such experiments. The guideline is derived from a theoretical framework and designed to yield valid and statistically significant results by a simple experimental procedure. Additionally, an open-source tool is provided that supports the suggestions. Guideline and tool have been successfully applied in an industrial context: Experiments revealed that a graphical notation used for model-driven development within SAP AG is as precise as a textual notation, but more difficult to understand.

Keywords: Model-driven development, Empirical Investigation, Understandability, Conceptual models.

1 Practical Motivation

Modelling software systems gains new importance in the context of *model-driven development* (MDD). In MDD, models are not only supplementary artefacts for documentation or communication, but strict, formal abstractions of the software system to be developed (Mellor et al. 2004). Usually, several levels of abstraction exist: The bottom level hides details of the hardware platform, whereas the top level abstracts from software architecture. MDD aims at defining transformations between models at distinct levels of abstraction, so that 'model compilers' can automatically transform the most abstract models (in several steps) into source code (OMG 2007).

Altogether, MDD shifts the focus of software development from programming to modelling. However, automatic model transformations are often difficult to achieve. In such situations the models can also be transformed by hand (OMG 2007). Then, developers are forced to implement exactly what the models prescribe, and any deviation between models and source code should be prevented by strict approval and model update.

When SAP AG, the world market leader in the field of enterprise software, decided to use MDD for the development of its latest application system, it became quickly clear that most model transformations had to be done by hand. Since describing the new system's architecture by common modelling languages like UML would have required massive language modifications and extensions, SAP AG started to create a set of new modelling languages, each representing another view on the system.

In MDD, modelling languages are called *metamodels* (Mellor et al. 2004, OMG 2007). They define the syntax and the semantics of a family of *models*, i.e., particular descriptions of some content. Such descriptions are also termed *schema* (Elmasri and Navathe 2000) or *script* (Gemino and Wand 2004).

The syntax of a metamodel is divided into an abstract and a concrete one (Kleppe et al. 2003): *Abstract syntax* (synonym: *grammar,* Gemino and Wand 2004) comprises a set of constructs and rules for combining them to create 'statements'. *Concrete syntax* refers to the *notation* of the statements, which can be either graphical or textual. By providing concepts to describe systems, metamodels equal *data* (Elmasri and Navathe 2000) or *conceptual models* (Gemino and Wand 2004).

SAP AG invented the new metamodels quite from scratch: The abstract syntax was designed to fit the intended system architecture, which is service-oriented (Woods and Mattern 2006). The concrete syntax, however, had to be precise and understandable to enable manual model transformations. Both properties are especially important for SAP AG, where the models are built by a team of designers and architects and implemented by more than hundred developers in several countries. Since the new application system is finally assembled at the level of the models (and not at the source code level), any deviation between implementation and models causes delays in development and, hence, additional costs.

Acknowledging this fact, SAP AG asked me to assess the precision and understandability of their (currently 11) metamodels. Actually, this assessment was triggered by some redundancy I detected in models of complex data types, where alternative concrete syntax was used to represent the same abstract syntax (see Section 5.1). Immediately, the question arose which one of the notations was more precise and easier to understand and, hence, should be preferred. Beyond it, since the concrete syntax of SAP AG's metamodels can be changed without negative consequences for implementation, the company was interested in a 'method' to test the understandability

of any invented notation before it is rolled out. These tests should be simple and, nevertheless, yield valid and statistically significant results.

Section 4 of this paper makes practical suggestions on carrying out such tests. The guideline is derived from a theoretical framework that is presented in Section 3. The framework combines current research on understandability in computer science (see Section 2 for a review) and knowledge of cognitive science. Section 5 reports the results obtained by applying the guideline to test the understandability of a particular metamodel of SAP AG.

2 Research on Understandability in Computer Science

Understandability can be seen as a (pragmatic) quality of conceptual models or metamodels (Moody et al. 2003): It constitutes an aspect of ease of use, namely the effort required to read and correctly interpret some model, which was created by applying a metamodel. Interpretation comprises identifying a particular connection of constructs (*parsing*) and assigning meaning to it (Anderson 2000).

Unfortunately, cognitive processes like interpretation and mental effort cannot be observed. To cope with such unobservability, cognitive psychology uses introspection, behaviourism and neuroscience.

Neuroscience, which measures brain activities in reading, has not been applied in computer science yet. In *introspection,* humans report the contents of their own consciousness and reasoning (Anderson 2000). The earliest discussions of ease of use in computer science were introspective. They took place in artificial intelligence, where examples have been used to praise the merits of either predicate logic (Hayes et al. 1974, Moore 1982), which is usually[1] written as text, or visual representations (diagrams) (Sloman 1971, Larkin and Simon 1987). Nowadays, since many mental processes (like reading) cannot be described this way and the introspective reports are purely individualistic, cognitive psychology rejects introspection as a valid method of investigation (Wessells 1982). I adopt this opinion for computer science.

Behaviourism tries to avoid the pitfalls of introspection by using observable behaviour as an indicator for mental processes (Anderson 2000). Methodologically, behaviourism has promoted experimental research, which has also been used to examine understandability in computer science. The list in Table 1 is not intended to be exhaustive, but merely to illustrate the variety of ways available to conduct such experiments.

An *experiment* is a scientific investigation in which one or more independent variables are systematically manipulated to observe their effects on one or more dependent variables (Sarafino 2005). The *independent variable* is the one whose effect is to be examined (e.g., 'data model'); the values it can take during manipulation are called levels (e.g., ERM, RDM) (Sarafino 2005). In contrast to the independent variable (which is given by the research question), the *dependent variable* can be chosen freely, provided that it serves as a measure of the effect (Robinson 1981). The dependent variables in Table 1 measure ease of use.

The variables and levels, the number of participants, and the statistical procedure, all vary among the studies in Table 1. All these decisions relate to *experimental design*, i.e., the way participants are selected and assigned to experimental conditions (Robinson 1981).

Because of the variation in methods, Table 1 does not help one in designing experiments to test the understandability of notations. That is why guidelines (Parsons and Cole 2005) and frameworks (Gemino and Wand 2004) have been proposed.

Understandability relates to the contents described by a model. Experiments dealing with such research questions should obey four criteria (Parsons and Cole 2005): (1) The notations tested should be informationally equivalent, i.e., all of the information in one notation should also be inferable from the other (Siau 2004). (2) Problem-solving tasks as well as (3) read-to-recall tasks, where the model is removed after some time, are to be avoided. (4) Subject matter experts should not participate as test persons.

All the experiments in Table 1 that deal with modelling or specification tasks violate the second criterion of Parsons and Cole (2005). Furthermore, at first sight, the fourth criterion contradicts the widely acknowledged influence of application and domain knowledge on understanding (Khatri et al. 2006). Both observations will be discussed in the Sections 4.2 and 6. Many experimental designs satisfy all of the criteria mentioned above; practicians need a guideline for choosing among them.

The framework proposed by Gemino and Wand (2004) is based on (1) the dimension of affecting factors and (2) the dimension of affected variables (outcomes). *Affecting factors* comprise (1) the content described, (2) the representation of the content, and (3) the characteristics of the persons involved in the experiment. *Affected variables* can focus on the process or on the product of using a conceptual model. For both, one can measure effectiveness and efficiency. Effectiveness assesses how well the conceptual model achieves its goal(s), e.g., correctness, whereas efficiency concentrates on the resources required to use the conceptual model.

Gemino & Wand (2004) successfully applied their framework to compare empirical investigations on conceptual models. Unfortunately, it is not equally well suited for planning experiments, since recommendations on appropriate factors and measures are missing. Additionally, some of the measures proposed for affected variables cannot be observed (e.g., 'cognitive model in the viewer') or are introspective (e.g., 'perceived ease of use'). Unobservability renders measures useless for experimental investigations (even if they are applied, see Table 1).

Consequently, there is still a need for a set of suggestions that guide the planning and conducting of experiments on the understandability of notations. To guarantee the validity of these experiments, a theoretical framework is necessary, which is presented in Section 3.

[1] Conceptual graphs (http://conceptualgraphs.org/) are a graphically notated predicate logic.

Study	Independent Variables (Levels)	Tasks (Number)	Dependent Variables	Experimental Design	N	Statistical Procedure	Results
(Batra et al. 1990)	Data model (EER, RDM)	Modelling (1 case)	Correctness (review), perceived ease of use	2 groups, matched in experience	42	t-test of means	• EER leads to higher correctness • No difference in perceived ease of use
(Bock and Ryan 1993)	Conceptual data model (EER, KOOM)	Modelling (1 case)	Correctness (review)	2 groups, random assignment	38	matched-pairs t-test of means	• Mostly no differences in correctness • Higher correctness of EER only for some facets
(Chan 1995)	Graphical query languages	Comprehension (32), Query specification (14)	Correctness (review)	1 group, repeated measurement	27	χ^2-test on distribution	Graphical queries are: • Easy to comprehend • Not easy to specify
(Jamison and Teng 1993)	Database representation style (graphical, textual), database complexity	Query specification (20)	Correctness (review), solution time, perceived ease of use	2 x 2 factorial design, repeated measurement	36	3 way analyses of variance (ANOVA)	Compared to textual representations, graphical representations lead to: • Shorter time to specify a query • Higher correctness of the queries • Higher perceived ease of use
(Lee and Choi 1998)	Conceptual data models (EER, SOM, ORM, OMT)	Modelling (2 cases)	Correctness (review), modelling time, perceived ease of use	4 groups, random assignment	100	Duncan test	Increased correctness and faster solutions for EER and OMT
(Palvia et al. 1992)	Conceptual models (DSD, ERM, OOM)	Comprehension (30)	Correctness, solution time	3 groups	121	ANOVA, correlation analysis	• Highest correctness for OOM • Shortest solution time for OOM, followed by DSD, ERM
(Bajaj 2004)	Conceptual data models (ERM variants with different numbers of constructs)	Comprehension (40)	(a) Correctness, (b) inverse of solution time, (c) learnability = improvement of (a) and (b) over time	2 groups, random assignment	64	t-test of difference between means	Models with more constructs_ • Lead to more accurate conceptualization of the domain • Increase time to process a schema • Are faster to learn
(Parsons and Cole 2005)	Conceptual data models (ER, EER), familiarity with application domain	Comprehension: syntactic (10), semantic (20), problem-solving (6)	Correctness (number of answers and review)	2 x 2 factorial design	81	paired t-test of means	• IS knowledge affects problem solving • Application knowledge does aid problem solvers in solving more demanding tasks

Abbreviations: DSD: Data Structure Diagram, EER: Extended Entity-Relationship Model, (K) OOM: (Kroenkes) Object Oriented Model, N: Total number of participants, SOM: Semantic Object Model, ORM: Object Role Model, OMT: Object Modelling Technique, RDM: Relational Data Madel

Table 1: Experiments on the Understandability of Models or Languages

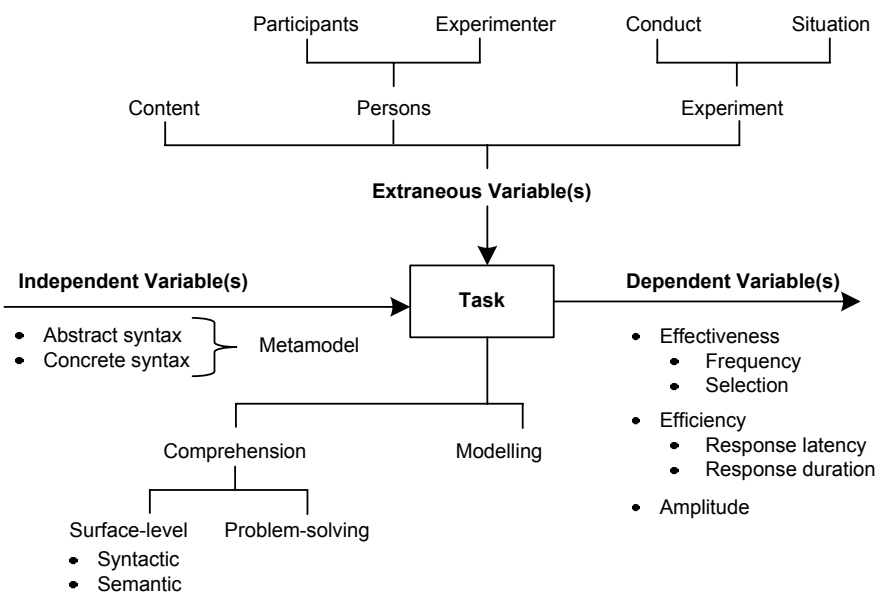


Figure 1: Theoretical Framework for Investigations on Understandability

3 Theoretical Framework

3.1 Overview

Experiments are always guided by the hypothesis that the independent variable(s) will *cause* the changes in the dependent variable(s) (Sarafino 2005). Formulating such hypotheses requires some measurable theory about understanding. Following behaviourism, this theory should be grounded in observable behaviour.

Instead of developing a completely new theory (which was not necessary to solve the practical question of SAP AG), Figure 1 consolidates variables and tasks that have been proposed by cognitive psychology or applied in computer science to test understandability. All variables have been theoretically justified by the authors that used them.

Figure 1 forms a superset of testable hypotheses, out of which particular hypotheses for experiments can be chosen. Section 4 contains practical suggestions for selecting among the variables and tasks explained in the next subsections.

3.2 Affecting Factors

The term 'affecting factors' stems from the framework proposed by Gemino and Wand (2004). In more detail, understandability is affected by independent and extraneous variables (Sarafino 2005). *Independent variables* are the ones whose effect is to be examined. Metamodels or their constituents, namely abstract syntax (e.g., the number of constructs, Bajaj 2004) or concrete syntax (notation), are the primary independent variables in experiments on understandability.

However, several experiments and frameworks indicate that understandability additionally depends on the task to be solved (Jamison and Teng 1993), on the content described (Gemino and Wand 2004) or on the knowledge of the participants concerning information systems, metamodels and domain (Khatri et al. 2006). These factors can be seen either as secondary independent variables, which have to be incorporated into the experimental design (Robinson 1981), or as extraneous variables that must be controlled (see Section 4.4). *Extraneous variables* are not the primary objects of study, but systematically influence the dependent variables and, thus, confound the experimental results (Sarafino 2005).

All the factors mentioned so far exclusively apply to experiments on understandability. Beyond it, cognitive psychology has identified several factors that affect *every* experiment, regardless of the research question. Hence, these factors always constitute extraneous variables. They are related to:

- The **persons** involved in an experiment (Robinson 1981):
  - **Participants**: Their age, sex, mental ability, motivation (to complete all tasks or to boycott the experiment).
  - **Experimenter**: The ability to instruct participants, *bias* (expecting a particular outcome unconsciously distorts the experimenter's behaviour or data gathering).
- The **conduct** of the experiment:
  - **Position effect**: Performance depends on the timely distance of a task from the start of the experiment (e.g., fatigue, getting bored, learning) (Mook 2004).
  - **Carry-over effect**: The performance achieved in some task depends on whether or not some other task has been done before (Sarafino 2005).
- The **experimental situation**: The location (noise, room temperature), the time of day (e.g., experiments in the morning usually yield better results than ones after lunch), and the equipment (failures, calibration) (Clark-Carter 2004).

3.3 Dependent Variables

The *dependent variable* is the one on which the effect of the independent variable is measured. Behaviourism requires the dependent variable to refer to behaviour. In cognitive psychology, the following measures of behaviour – potential dependent variables - are common (Robinson 1981):

1) *Frequency*, e.g., the number of correct answers or solved problems.

2) *Selection*, e.g., which of several notations a participant uses for modelling or which of several answers he or she chooses.

3) *Response* latency (or response time), which is concerned with how long it takes for a behaviour to be emitted, e.g., how quickly a participant reacts.

4) *Response duration*, i.e., the length of time some behaviour occurs (e.g., how long a participant deals with a task).

5) *Amplitude*, measuring the strength of response, e.g., the brain activities in performing a task.

Amplitude is mainly a measure of neuroscience. Frequency and selection assess the effectiveness of a metamodel, whereas response latency and duration refer to its efficiency (see the framework of Gemino and Wand 2004).

In the experiments summarized by Table 1, frequency is the measure underlying correctness, solution time often refers to response latency and modelling time to response duration. Additionally, some experiments apply the measure 'perceived ease of use', which I have excluded from Figure 1, since it does not express observable behaviour, but is introspective, and introspection conflicts with behaviourism.

If correctness is verified by multiple-choice questions (e.g., Appendix B in Khatri et al. 2006), it is based on the measure 'selection'. Alternatively, selection can be used as a measure of behaviour when the task consists in modelling. In general, there is a strong relationship between the task to be solved in an experiment and the measures and dependent variables applicable. Typical experimental tasks are sketched in the next section.

3.4 Experimental Task

Table 1 shows that experiments on ease of use are based on two types of tasks (Gemino and Wand 2004): comprehension tasks (reading models) and modelling/specification tasks (creating new abstract descriptions).

Comprehension tasks require the participants of an experiment to answer questions on some model. If the questions refer to the constructs of the model (e.g., "How many attributes describe the entity type ORDER?"), the *comprehension* task is called *syntactic* (Khatri et al. 2006). In contrast, *semantic comprehension* tasks assess the understanding of the *contents* described (e.g., "Every employee has (a) a unique employee number, (b) more than one employee number.") (Khatri et al. 2006).

Both types of comprehension tasks refer to surface-level understanding. Deeper-level understanding is addressed by *problem-solving tasks*, where participants are requested to determine whether and how certain information can be retrieved from a model (Shanks et al. 2003).

The questions in comprehension tasks can be open or closed. In closed questions, the available options for answers are given (multiple-choice form), and the participants have to select among them. Open questions, on the other hand, do not suggest answers.

4 Practical Suggestions

4.1 Overall Considerations

Planning an experiment amounts to selecting the variables, formulating a hypothesis and determining the experimental task. In the light of the practical motivation of this paper, these actions are directed to two goals: First, the experimental results should be valid and statistically significant. *Internal validity* aims at maximizing the degree to which the variation of the dependent variable can be attributed to the independent variable. *External validity* refers the generalizability of the results (Sarafino 2005). Secondly, the experiment should be easy to prepare, conduct and evaluate. More complex research designs should only be used if the simple causal relationships are understood (Parsons and Cole 2005).

Simplicity of design and evaluation is achieved by limiting the number of independent variables. In order to, nevertheless, preserve internal validity, any variable that influences understandability, but has not been chosen as an independent variable, must be treated as an extraneous one and, hence, be controlled. This can be done by several techniques, which are inherent to well-known types of experimental design (see Section 4.4).

The next subsections contain practical suggestions on the selection of variables, task and experimental design if simple, valid experiments have to be conducted to answer the following research question: Which of several alternative notations for the same abstract syntax is easier to understand? The suggestions are supported by the tool /`notate`/, which is available for downloading from the website http://sourceforge.net/projects/notate. The tool is suitable for experiments on understandability in general. Since it is an open-source project, it can be adapted to specific experimental needs.

4.2 Selecting the Experimental Task

If understandability (and not general ease of use) is the research objective, comprehension tasks should be used, not modelling tasks. This suggestion is derived from the fact that modelling is an act of language generation, and cognitive psychology knows more about language comprehension than about language generation (Anderson 2000). Moreover, the mental process of modelling cannot be observed, and although the product of modelling (the model) can be reviewed, most reviews remain subjective to some extent (experimenter bias, see Section 3.2) (Bourne et al. 1966).

The understandability of notations is to be tested by semantic comprehension tasks (second criterion of the guideline by Parsons and Cole 2005), since problem-solving tasks address deeper levels of domain understanding, and syntactic comprehension tasks do not require interpretation effort.

The correctness of interpretation is usually verified by asking questions about some subject matter (Anderson 2000, Bourne et al. 1966) or, in the context of this paper, about the contents of some model. The models underlying the questions should describe comparable contents. Otherwise, wrong answers can be the consequence of more complicated information and not necessarily of a

less understandable notation. However, researchers should *not* represent *exactly the same* content by alternative notations, since this encourages carry-over effects (see Section 3.2), i.e., participants reuse the solution of some task (content they have understood) to answer questions that refer to another notation.

4.3 Selecting the Variables

The suggested experimental task (answering questions about models that apply alternative notations) already hints at notation as an inevitable *independent variable* and the number of correct answers for each notation (frequency) as *dependent variable*.

Using more than one measure of behaviour increases the reliability of the findings, especially, if the measures are independent (Bourne et al. 1966). Consequently, I suggest response time as an additional dependent variable. Since it also accepts wrong solutions, it is independent of correctness. Moreover, evaluating mental processes by response time has a long tradition in psychology (Anderson 2000, Mook 2004).

In the context of this paper, applying both correctness and response time as dependent variables allows us to assess the equivalence of notations from two complementary perspectives (Siau 2004): Correctness is mainly related to the *informational equivalence* of notations (i.e., whether the same kind of information can be inferred from both notations), whereas response time indicates the effort required for such inferences (*computational equivalence*).

In most experiments of Table 1, correctness is determined by reviews, which can be rather subjective (experimenter bias). Subjectivity in deciding about correctness can be avoided if the questions have multiple-choice form and allow only one ultimate solution (Bourne et al. 1966). The solutions chosen by the participants constitute an additional (but correlated) dependent variable (selection).

Multiple-choice questions with three answers reduce the possibility of answering correctly by chance and, at the same time, do not confound the response time with the time for reading a long list of answers. Positively formulated answers are to be preferred, since negations distort the solution time (Anderson 2000). Moreover, the correct answers should be spread over the options participants can choose.

Since the dependent variable must be tested under at least two conditions, at least two levels of the independent variable are necessary (Sarafino 2005). This can be achieved by contrasting either two notations or a notation and a control group. In a *control group*, the independent variable is not manipulated, e.g., software systems are described by some natural language instead of a notation.

Rarely more than four levels of an independent variable are investigated. Examining more than one independent variable requires a factorial design (see Section 4.4), which renders planning, conducting and evaluating the experiment more complicated. Consequently, for simple experiments about understandability, I suggest using just one independent variable; validity can be guaranteed by controlling the other variables.

4.4 Control and Experimental Design

Extraneous variables that potentially influence the outcomes of an experiment, but have not been chosen as independent variables, must be controlled, i.e., removed, kept constant or randomized.

Mainly extraneous variables referring to the experimental situation can be easily *removed* by, e.g., using a quiet room. *Constancy* guarantees that all conditions are identical except for the manipulation of the independent variable. It is often applied to methodological aspects of an experiment, e.g., all experiments are conducted at the same time of day and by the same experimenter. The equivalence of the contents described by the models is another kind of constancy.

Whenever possible, *randomization* should be preferred to constancy since it increases the external validity of an experiment (Robinson 1981). Randomization is used when the influence of an extraneous variable is not known (e.g., differences between male and female participants in understanding notations), must be neutralized (e.g., position effects, carry-over effects) or should be equated (e.g., age, knowledge).

An *experimental design* can be regarded as a general plan for (types of) experiments that joins independent variables and control techniques. The main experimental designs are between/within-subjects, block and factorial design.

In a *between-subjects design*, the participants are randomly assigned to several groups (Clark-Carter 2004). Each group is treated by only one level of the independent variable (a particular notation). Thus, carry-over effects between alternative notations cannot occur. However, in small samples (number of participants in a group $n_i < 15$, total number of participants $\Sigma\, n_i = N < 30$) randomization may lead to groups that are unequal concerning individual characteristics of the participants (e.g., IQ, experience); this can skew the experimental outcomes.

A *block design* avoids unequal groups. Here, at least one matching factor strongly connected with the dependent variable is determined (e.g., experience). The participants are classified according to the levels of this factor (e.g., high, medium, low experience) and randomly assigned to groups so that each factor level is represented in all groups by the same number of participants (Sarafino 2005). In other words, the matching factor is kept constant. The effort this design causes is only worth if the matching factors (which are difficult to detect) are highly correlated.

The *within-subjects* design is appropriate to yield significant results with a small sample (Clark-Carter 2004): One group is tested for all levels of the independent variables, i.e., for all notations. In this setting, all extraneous variable related to the persons involved in the experiment remain constant. The disadvantages of this design consist in carry-over effects and experimenter bias (e.g., the favourite notation is explained better).

If more than one independent variable is investigated, a *factorial design* must be applied (Robinson 1981). Now, the groups are the result of combining each level of an

independent variable (e.g., notation) with each level of another one (e.g., knowledge). Since each participant is assigned to only one group, a large sample is required. The more levels or independent variables are considered, the more difficult become the planning and the statistical evaluation of the experiment.

Table A in the Appendix summarizes the experimental designs and their (dis-) advantages. It also lists the statistical procedures that are allowed to evaluate the experimental outcomes and recommends on the sample size N, i.e., the total number of participants in all groups.

The within-subjects design suggests itself as a simple one that requires few participants. Its main drawback – carry-over effects – can be avoided by randomizing the order of tasks and by keeping the described content equivalent, but not identical. To reduce the danger of experimenter bias, not only the notations of interest should be tested, but also an additional 'placebo notation' (which is, however, not evaluated).

All experimental designs have to cope with position effects. Negative position effects (e.g., getting bored or tired) do not occur if the tasks are as brief and as interesting as possible (Sarafino 2005). Learning is a positive position effect that, however, may lead to skewed results (Jamison and Teng 1993). A *warm-up phase*, i.e., several tasks whose solutions are not evaluated, encapsulates very strong learning in the beginning of an experiment (Sarafino 2005). It also makes participants familiar with the equipment and thus reduces errors in operation.

5 Applying the Guideline

5.1 Background

One of SAP AG's metamodels is intended to describe complex data types (e.g., 'invoice position'), which combine several simple data types (e.g., 'identifier', 'description', 'amount'). Complex data types facilitate integration in a service-oriented architecture and the generation of source code.

The *abstract syntax* of the metamodel defines the constructs 'complex data type', 'attribute' and 'element data type' (which is a simple data type). These constructs can be combined as follows: (1) Each attribute is assigned to exactly one element data type. (2) Complex data types consist of several attributes, which can be optional (cardinality '0...1') or not (cardinality '1'). The term 'cardinality' is slightly misleading since multi-valued attributes are currently not allowed.

Two types of concrete syntax are simultaneously in use (see Figure 2): The *graphical* one represents complex and element data types by parallelograms, and attributes by rectangles. Moreover, it uses special arcs to show whether (—|▷) or not (—▷) attributes are optional. The *textual notation* expresses the same information by tables where attributes, their data types and cardinalities are contained in rows[2]. Both notations are informationally equivalent (Siau 2004), since they represent the same abstract syntax.

One can argue about the distinctive features of graphical and textual notations. In this paper, it is discriminated as follows (Larkin and Simon 1987): The constructs of *textual notations* are (strings of) signs (characters, numbers, or symbols) that are allowed to form sequences or arrays. In contrast, the constructs of *graphical notations* are geometric figures (e.g., circles, rectangles, or arcs), which can be arbitrarily connected.

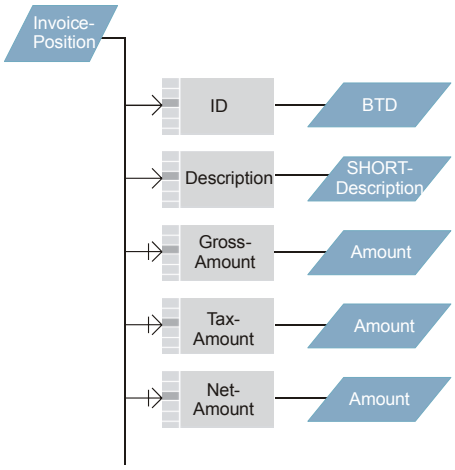
5.2 Hypothesis and Control

SAP wanted to know, which of the notations was easier to understand and more precise. Since neither experimental results (Table 1) nor psychological theories (Anderson 2000, Wessells 1982) indicate that a particular type of representation is easier understandable for humans, the experiment tested the following *hypothesis* (two-tailed):

H_0: There is no difference in the understandability of the graphical and the textual notation of the SAP metamodel for complex data types.

The precision of the notations was tested indirectly by using the number of correct answers for each notation as a dependent variable. The reliability of the experimental results is increased by recording response time as an additional dependent variable (see Section 4.3). According to the guideline (see Section 4.3), notation was the single independent variable, with its levels comprising of a) and b) of Figure 2 and a placebo (UML class diagrams).

a) Graphical notation



b) Textual notation

NDT	Name	Data Type	Card.
Expense-Report			
	ID	ID	1
	EmployeeID	ID	1
	Payment-Currency	Currency-Code	1
	Stay-Description	MEDIUM-Description	0..1
	StayLocation-Name	MEDIUM-Name	0..1

Figure 2: Notations for Complex Data Types

[2] The SAP tables contain additional columns for value ranges, integrity conditions, etc. I omitted them in my experiment to keep the comparison fair.

The within-subjects design has been suggested in Section 4.4 as the simplest one to test the above hypothesis. To ensure the validity of the experiment, the following control techniques have been applied:

- *Constancy*: The experiments were conducted three times (since three universities have been involved) by the same instructor at the same time in the morning. All participants were instructed by the same (written) material. The modelled content was equivalent.
- *Randomization*: The participants were randomized concerning knowledge, age and sex. The tool /notate/ showed the tasks to be solved in random order for each participant.

5.3 Experimental Process

Sample Size: In a within subjects-design, 15 participants are necessary to detect a large effect (d = 0.8) by a paired t-test (two-tailed) with a power of 0.8 ($\beta = 0.2$) and $\alpha = 0.05$ (Clark-Carter 2004). Under the same conditions (d, α, β), non-parametric tests need 16 to 24 subjects (Clark-Carter 2004). Surplus participants are desirable to buffer against outliers.

Sample: For a small financial incentive, 42 subjects (35 male, 7 female) volunteered to participate in the experiment. Six of them already had graduate degrees in computer science, the other 36 were students (undergraduates: 17, graduates: 19) of computer science, computational visualistics, data and knowledge engineering, and business informatics from three universities. The sample size relevant to the statistical test procedures is smaller than 42 because outliers (detected by SPSS® boxplots) had to be removed.

Conduct: Each participant was seated at a PC where the tool /notate/ was installed. The participants received written instructions of the notations to be tested. After 15 minutes, the warm-up phase began. When all participants had finished this phase, the actual experiment started. The subjects were told to answer all questions correctly and as quickly as possible. During both the warm-up and the experiment, the questions appeared randomly on each PC.

Tasks: The participants had to answer questions concerning the content of models of complex data types, which used the notations of Figure 2 and a placebo notation (UML class diagrams). Hence, semantic comprehension tasks (see Section 3.4) had to be solved. I used real models of SAP AG's latest application system; reproducing them here would violate the non-disclosure agreement. The models in the experiment looked like the ones depicted in Figure 2, but differed in their size (see Table 2). These sizes reflect the variation in real-world complexity.

All questions were in multiple-choice form, dealt with cardinalities and had only one correct solution. For example, a question referring to Figure 1b) would be: What does the description express? a) In each expense report, at least one stay location is given; b) Expense reports where the name of the stay location is missing are allowed; or c) Each expense report must contain exactly one stay location. Answer b) is correct. To check whether or not the questions were clearly formulated, a pre-test involving five participants was run; this data is omitted in Table 2.

The warm-up phase consisted of six questions: Two for each level of the independent variable, respectively related to a small and a large model. The experimental data was gathered from another 27 questions.

5.4 Data Analysis

Table 2 shows the descriptive statistics of the experiment. Concerning the correctness of answers, there is no significant difference between the notations a) and b) of Figure 2 (paired t-test of means: t = -1.481, df = 39, N = 40, p = 0.147, two-tailed). The non-parametric sign test leads to the same conclusion (N = 40, p = 0.166; one-tailed in favour of the textual notation), whereas the Wilcoxon matched-pairs signed rank test shows a significant superiority of the textual notation at $\alpha = 0.1$ (z = -1.470, N = 40, p = 0.085, one-tailed).

Notation level	**Graphical** (Fig. 1a)	**Textual** (Fig. 1b)	**Placebo**
Size of the models (number of attributes)			
Very small (≈ 4)	2	2	2
Small (≈ 9)	2	2	2
Medium (≈ 14)	4	4	2
Large (> 18)	2	2	1
Sum	**10**	**10**	**7**
Descriptive statistics			
Number of correct answers	$\mu = 9.63$, $\sigma = 0.59$	$\mu = 9.80$, $\sigma = 0.52$	-
Response time (seconds)	$\mu = 26.41$, $\sigma = 5.12$	$\mu = 23.37$, $\sigma = 4.98$	-

μ: Arithmetic mean, σ: Standard deviation

Table 2: Number of Models and Descriptive Statistics

A significantly shorter response time indicates the better understandability of the textual notation (N = 39): The difference is detected by the paired t-test of means (t = 5.625, df = 38, p = 0.00, two-tailed), while both one-tailed tests – the sign test (z = -4.163, p = 0.00) and the Wilcoxon test (z = -4.585, p = 0.00) - favour the textual notation. All statistics are calculated by the program SPSS® version 14.0.1.

Concentrating only on statistical significance can be misleading since it is affected by sample size: Larger samples easier achieve significant results than smaller ones (Clark-Carter 2004). *Effect size* expresses the magnitude of a result (the actual difference in the understandability of notations) independently of sample size (Clark-Carter 2004). The measures of effect size and what constitutes a large, medium or small effect depend on the statistical test procedure (Cohen 1988).

The t-test could not detect a significant difference in correctness because the effect is small (d = 0.30; $\sigma_{adjust} = 0.56$), whereas the effect for the Wilcoxon-test is almost medium ($r_{es,z} = 0.23$). The significantly shorter response time for the textual notation should also not be overemphasized, since it results from only a medium effect in

the t-test (d = 0.60; $\sigma_{adjust} = 5.05$); in the Wilcoxon-test, however, the effect is large ($r_{es,z} = 0.73$).

Altogether, the notations SAP AG has invented are equally capable of unambiguously describing complex data types. The textual notation is just easier to read.

6 Discussion

Three issues have to be discussed: the contribution of the experiment to research, the limitations of the experiment and the contribution of the guideline to practice.

Bodart et al. (2001) argue that optional attributes should be replaced by subtypes with mandatory attributes if users have to draw inferences from the models. This argument does not apply to the investigated metamodel of SAP AG because it was designed to create implementation models for MDD. In this context, with focus on content-preserving model transformations, the optionality construct may be retained (Bodart et al. 2001).

In contrast to all the papers that either glorify the 'easy to understand' graphical notations (e.g., Sloman 1971, Larkin and Simon 1987) or complain about their vagueness (e.g., ter Hofstede 1992), the results of the experiment show that a graphical notation can be as precise as a textual one – and harder to understand.

It is open to dispute whether or not the notation in Figure 2b) really is a textual one (see the comprehensive discussion in Shimojima 2001). At least, the definitions used here stem from cognitive science.

A limitation of the conducted experiment is the nature of the sample population, which mainly consists of computer science students. Nevertheless, the external validity of the experiment is supported by two arguments: First, the population was randomized and included both undergraduate and graduate students as well as six professionals. Secondly, Parsons and Cole (2005) recommend that experts should not participate in such experiments (see Section 2) because they are able to answer questions solely by using their knowledge, even if the notations are not understandable. This suggestion is respected here.

Another limitation consists in the simple experimental design. Clearly, more complex designs are necessary to gain deeper insight into the nature of understandability, but they require more planning and evaluation effort. To answer simple research questions like the one posed by SAP AG, the within-subjects design is suitable. This leads me to the practical contribution of the guideline:

Mainly when it comes to manual model transformations in MDD, the understandability of models *gets* important. If several alternatives types of concrete syntax exist, the one that is easier to understand should be chosen. The proposed guideline and the tool /`notate`/ help companies to conduct simple, yet valid experiments concerning understandability, which simultaneously (by using correctness as another dependent variable) test the precision of the notation.

Understandability is just one criterion for evaluating metamodels. It is appropriate when abstract syntax, dynamic semantics and model usage are fixed. However, to select among metamodels, companies should also consider other criteria, which can be found in comprehensive checklists, e.g., (Siau and Wang 2007).

In acknowledging my results, SAP AG admitted that the graphical notation became necessary because the tool containing the repository of the new application system is unable to display tables. So, redundant modelling is unavoidable here. In an upcoming experiment SAP AG will apply my guideline to test the understandability of alternative notations for other metamodels as well.

Acknowledgments I would like to thank SAP AG, especially Peter Zencke and Stefan Kaetker, for allowing this investigation and the use of real-world examples. Thanks also to Baoguo Chen from Beijing Normal University for discussions about the nature of psychological experiments and for the inspiration to create /notate/.

References

Anderson, J.R. (2000): *Cognitive Psychology and its Implications*. 5th ed., New York, Worth.

Bajaj, A. (2004): The effect of the number of concepts on the readability of schemas: an empirical study with data models. *Requirements Engineering* **9**: 261-270.

Bourne, L.E. and Battig, W.F. (1966): Complex Processes. In *Experimental methods and instrumentation in psychology*. 541-576. Sidowski, J.E. (ed.). New York, McGraw-Hill.

Bodart, F., Patel, A., Sim, M. and Weber, R. (2001): Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests. *Information Systems Research* **12**(4): 384-405.

Chan, H.C. (1995): Naturalness of Graphical Queries Based on the Entity Relationship Model. *Journal of Database Management* **6**(3): 3–13.

Clark-Carter, D. (2004): *Quantitative psychological research*. 2nd ed., Hove, Psychology Press.

Cohen, J. (1988): *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed., Hillsdale, Erlbaum.

Elamasri, R. and Navathe, S.B. (2000): *Fundamentals of Database Engineering*. Reading, Addison-Wesley.

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

Hayes, P.J. (1974): Some Problems and Non-Problems in Representation Theory. *Proc. of the AISB Summer Conference*. 63-79, University of Sussex.

Jamison, W. and Teng, J.T.C. (1993): Effects of Graphical Versus Textual Representation of Database Structure on Query Performance. *Journal of Database Management* **4** (1): 16–23.

Khatri, V., Vessey, I., Ramesh, V., Clay, P. and Park, S.-J. (2006): Understanding Conceptual Schemas: Exploring the Role of Application and IS domain Knowledge. *Information Systems Research* **17**(1): 81-99.

Kleppe, A., Warmer, J., and Bast, W. (2003): *MDA Explained: The Model Driven Architecture – Practice and Promise*. Boston, Addison-Wesley.

Lee, H. and Choi, B.G. (1998): A Comparative Study of Conceptual Data Modeling Techniques. *Journal of Database Management* **9**(2): 26-35.

Mellor, S.J. et al (2004): *MDA Distilled: Principles of Model-driven Architecture*. Boston, Addison-Wesley.

Moody, D.L., Sindre, G., Brasethvik, T. and Sølvberg, A. (2003): Evaluating the Quality of Information Models : Empirical Testing of a Conceptual Model Quality Framework. *Proc. of the 25th International Conference on Software Engineering ICSE*. 295 – 305. IEEE, Los Alamitos.

Mook, D. (2004):*Classic experiments in psychology*. Westport, Greenwood.

Moore, R.C. (1982): The Role of Logic in Knowledge Representation and Commonsense Reasoning. *Proc. AAAI '82*. 428–433, Menlo Parc, AAAI.

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

Object Management Group (OMG): MDA™ Guide Version 1.0.1, Document number omg/2003-06-01. http://www.omg.org/. Accessed June 12, 2007.

Palvia, P.C., Liao, C. and To, P.-L. (1992): The Impact of Conceptual Data Models on End-User Performance. *Journal of Database Management* **3**(4): 4-15.

Parsons, J. and Cole, L. (2005): What do the pictures mean? Guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modelling techniques. *Data & Knowledge Engineering* **55**: 327-342.

Robinson, P.W. (1981): *Fundamentals of Experimental Psychology*, 2nd ed., Englewood Cliffs, Prentice-Hall.

Sarafino, E.P. (2005): *Research Methods: Using Processes and Procedures of Science to Understand Behavior*. Upper Saddle River, Pearson.

Shanks, G., Nuredini, J., Tobin, D., Moody, D. and Weber, R. (2003): *Representing things and properties in conceptual modeling: An empirical investigation*. Proc. 11th Eur. Conf. Inform. Systems, Italy (June 16-21).

Shimojima, A. (2001): The Graphic-Linguistic Distinction: Exploring Alternatives. *Artificial Intelligence Review* **15**: 5–27.

Siau, K. (2004): Informational and Computational Equivalence in Comparing Information Modeling Methods. *Journal of Database Management* **15**(1): 73-86.

Siau, K. and Wang, Y. (2007): Cognitive Evaluation of information modeling methods. *Information and Software Technology* **49**(5): 455-474.

Sloman, A. (1971): Interactions Between Philosophy and Artificial Intelligence. *Artificial Intelligence* **2**: 209-225.

ter Hofstede, A.H.M. and van der Weide, T.P. (1992): Formalization of techniques: chopping down the methodology jungle. *Information and Software Technology* **34** (1): 57 65.

Wessells, G. (1982): *Cognitive Psychology*. New York: Harper & Row, 1982.

Woods, D. and Mattern, T. (2006): *Enterprise SOA – Designing IT for Business Innovation*. Beijing, O'Reilly.

Appendix

Design	**Between-subjects**	**Within-subjects**	**Block (Matched)**	**Factorial**
No. of IV (levels)	1 (n)	1 (n)	1 (n)	m > 1 (n)
No. of groups	n	1	n	m × n
Pro:	No carry-over effects	• Simple • Small samples • Constancy of individual characteristics	• Precise • No carry-over effects • Individual differences balanced	Interactions between IV can be examined
Contra:	• Unequal groups possible • Large samples	• Carry-over effects • Experimenter bias	• Effort • Matching factor must exist	• Large samples • Difficult to interpret for m > 3
Statistical test procedures				
Metric DV	♦: independent t ∗: F-test, ANOVA	♦: paired t-test of means ∗: MANOVA		MANOVA
Ordinal DV	♦: Mann-Whitney U ∗: Kruskal-Wallis H	♦: Wilcoxon signed rank test (matched) ∗: Friedman's χ^2		-
Nominal DV	♦/∗: χ^2 contingency test	♦: Sign test, McNemar's test of change ∗: Cochran's Q-test		-
Sample Size ♣	1-t: n_i = 20 [50] 2-t: n_i = 25 [60]	1-t: N = 11 [26] 2-t: N = 15 [35]	see between-subjects	2-t only, m = 3: n_i = 20 [50]

♣ To detect a large [a medium] effect with $(1-\beta) = 0.8$ and $\alpha = 0.05$ (Cohen 1988).

Abbreviations: IV: Independent variable, DV: Dependent variable

Table A: Summary of Experimental Designs and Statistical Test Procedures