Psychological aspects of equation-based modelling

Alexander Pollok; Andreas Klöckner; Dirk Zimmer

Psychological aspects of equation-based modelling languages like Modelica are under-represented in literature. This does not reflect the growth of the corresponding userbase. In this paper we try to close this gap by tackling the problem from three sides: we conduct expert interviews, we conduct an experiment based on self-reported timings to analyse the effects of inheritance on understandability, and we conduct an online experiment to analyse the effects of model representations on the performance at modelling tasks. The expert interviews suggest that experienced modelling experts tend to develop their models from the top-down, while novices do the opposite. Results from the second experiment indicate that the effect of inheritance on the time to understand a model is both significant and large. The results of the last experiment imply that graphical representations outperform block-diagrams for several metrics. These results open a broad research field on the theory ofgoodmodelling practice.

[1] Alexander Pollok, Andreas Klöckner, and Dirk Zimmer. Psychological aspects of equation-based modelling. *Mathematical and Computer Modelling of Dynamical Systems*, 25(2), 25 March 2019. `doi:10.1080/13873954.2019.1594310`.

Psychological Aspects of Equation-based Modelling

Alexander Pollok[a,b], Andreas Klöckner[a,c] and Dirk Zimmer[a]

[a]Institute of System Dynamics and Control, DLR German Aerospace Center, Oberpfaffenhofen, Germany
[b]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
[c]Student at the Faculty of Cultural and Social Sciences, FernUniversität in Hagen, Germany

ARTICLE HISTORY
Compiled April 9, 2018

ABSTRACT
Psychological aspects of equation-based modelling languages like Modelica are underrepresented in literature. This is in stark contrast to the growing userbase of these languages. In this paper we try to close this gap tackling the problem from three sides: (1) we conduct expert interviews, (2) we conduct an online experiment to analyse the effects of inheritance, and (3) we conduct a second online experiment to analyse the effects of model representations. It is found that the effect of inheritance on the time to understand a model is both significant and large, at 26.65 s per inheritance level ($t(90) = 2.8, p = 0.006$). We also show that graphical representations outperform representations by block-diagrams for several metrics ($p < .001$). Textual models based on physical equations or algorithms rank in between these two representations. Our results show that the way a physical system is modelled has strong influences on the way it is understood and used by others. These results have implications not only for modelling practice, but also open a broad research field on the theory of *good* modelling practice.

1. Problem definition

The every-day business of a modelling and simulation expert contains a significant amount of thinking of how to model a physical system in an adequate way. To make related tasks easier, modern modelling and simulation languages as well as tools have evolved with performance, flexibility, and usability in mind. Tool developers and language maintainers usually concern themselves with each of those aspects. The situation appears a bit different in scientific literature, where the first two aspects dominate the discussion, while the topic of usability is often neglected.

There are several relevant reasons for this mismatch. First, usability in general is hard to measure in any meaningful way. While benchmarks for simulation performance are readily available, no such thing exists for the understandability of models. Second, the usability of tools and the comprehensibility of models are features that are often taken for granted if done correctly. Only blatant flaws are noticed consciously, and even then, often forgotten.

In industrial practice, the performance, quality and integrity of simulation models can be limited by the comprehension of the modelling expert, if models get complex enough. This illustrates the significance of the corresponding research gap.

{alexander.pollok,andreas.klöckner,dirk.zimmer}@dlr.de

Of course, related research has been done in connected fields. In [1], general guidelines on programming language design are addressed. [2] and others tried to predict the time it takes to perform an atomic programming task, like inserting a word in a text editor. The authors of [3] conducted a review about usability aspects of software and coding styles. The field of software ergonomics deals with the effect of software design on the performance of the user [4]. This relates mostly to usability topics, implementation issues are addressed less often. [5] presents a framework of cognitive dimensions that play a role in understanding, learning and using a programming environment. In [6], this framework is used to analyse some aspects of the C++ programming language. The works of [7] and [8] mention several categories to evaluate programming languages. The following papers are diving into the domain of physical modelling: [9] describes how cognitively efficient visual notations can be designed, this relates directly to graphical layers of modelling languages. [10] presents a method to evaluate information modelling methods like unified modelling language (UML). Pedagogical aspects of modelling languages are addressed in [11], where an interactive teaching environment for the *Modelica* language is presented. These citations barely skim over the vast amount of work done in this research area. However, to the best knowledge of the author, no research regarding language design and no quantitative usability studies have been published in the context of equation-based modelling languages.

In this paper, we identify interesting psychological aspects in equation-based modeling languages and tools, Modelica in particular. A first attempt at quantification is made. We also want to lay a foundation for further research regarding the development of equation-based languages and corresponding tools. To tackle this research gap, we proceed as follows: Section 2 introduces equation-based modeling languages to unfamiliar readers. In Section 3, the subset of findings from literature, that should also be relevant for equation-based modelling, is presented. The results from a series of explorative expert surveys are summarized in Section 4, trying to stake out the problem-space of this work. Furthermore, two experiments are performed, analysed and discussed: The influence of inheritance on the understandability of equation-based models is investigated in Section 5 (this experiment is based on the work presented in [12]). In Section 6, different representations for physical models were compared with regard to several usage metrics. The findings of this paper are discussed and the paper is concluded in Section 6.3.

2. Short introduction to equation-based modeling languages

The studies performed in this paper are all based on the equation-based language Modelica. A quick review of its major design principles shall hence serve as an introduction to the unfamiliar reader. However, a single section cannot replace learning the actual language. The examples here are thus not fully explained but offer a first impression. Readers who want to learn more are referred to [13], [14] and [15].

On the lowest level, a model consists in the declaration of parameters, variables and corresponding equations. The variables can be of various types and the equations represent mostly differential algebraic equations. Listing 1 displays a corresponding example, where *der* represents the derivative operator.

Instead of formulating a complete system of equations where the number of equations matches the number of variables, the modeler can model only a subsystem: a component. Using an interface like an electric pin, he can describe the equations of a capacitor as in Listing 2.

Listing 1 Modelica Code of a simple Capacitor system

```
model CapacitorSystem
  import SI = Modelica.SIunits;
  parameter SI.Capacitance C;
  SI.Voltage v;
  SI.Current i;
equation
  v = 3*time;
  i = C*der(v);
end CapacitorSystem;
```

Listing 2 Modelica Code of a simple Capacitor

```
model Capacitor
  import SI = Modelica.SIunits;
  parameter SI.Capacitance C;
  SI.Voltage v;
  SI.Current i;
  PositivePin p;
  NegativePin n;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
  i = C*der(v);
end Capacitor;
```

Such a component can be linked to graphical icons such as the capacitor symbol and a total system such as an electric circuit can be graphically composed by the means of the language. Figure 1 displays such an example. It is a design decision whether to use graphical or textual code but typically engineers prefer graphical code for more complex systems. Graphical components can themselves be built upon graphically modeled subsystems or textual code. Even a combination is possible.

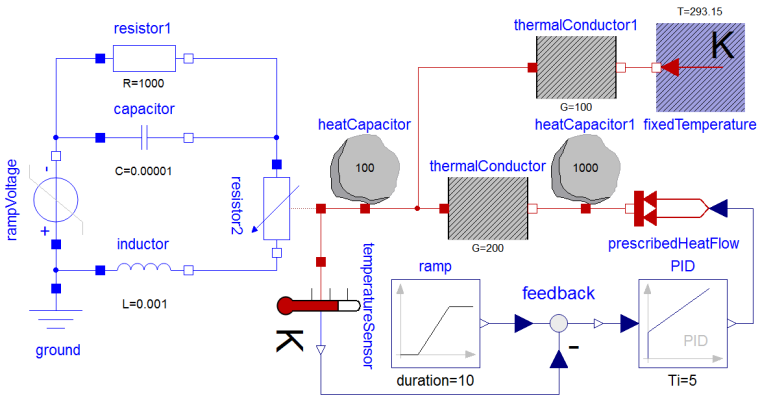

Figure 1. electric circuit modelled in Modelica

Figure 1 also shows that there are graphical models for various physical systems such as mechanics or electrics but also there are graphical components for control signals as many modelers know from Simulink. Many components that belong to the same domain also reuse the same equations. For instance, many electric components have a current that runs through its two pins. Partial models can be used here. Listings 3 and 4 show an example from the Modelica Standard Library [16].

Listing 3 simplified partial model from the Modelica Standard Library

```
partial model OnePort
  SI.Voltage v;
  SI.Current i;
  PositivePin p;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
end OnePort;
```

Listing 4 simplified extending model from the Modelica Standard Library

```
model Capacitor
  extends Interfaces.OnePort;
  parameter SI.Capacitance C;
equation
  i = C*der(v);
end Capacitor;
```

A model of a capacitor can now reuse these equations by an *extends*-statement. This is the Modelica version of inheritance. Also multiple inheritance is possible. In addition to this, there are also language constructs that enable to use models or class of models as parameters. Furthermore the language supports not only differential equations for continuous systems but also the formulation of discrete events. Causal assignments like in a conventional programming language can be used to express causal relations.

What finally results is a declarative modeling language that describes the dynamic behavior of a system. The model code does not state what to do with this description. Whether to simulate, optimize or transform this system is (in principal) not part of this language. The models become hence self-contained. Modelica model libraries are hence a valuable knowledge source and not only processed by a simulation program, they are also intensively studied by other modelers to learn more for their own models. Human readability and ease of understanding are thus of major importance.

The design principles of Modelica are also shared (to some degree) by other equation-based languages such as gPROMS [17], 20-Sim [18] or Matlab Simscape [19]. The research here is hence also relevant for other languages. We have chosen Modelica since this is an open standard shared by many tools with a healthy community that also hosts user groups in various countries around the globe. It is used by many industry companies and features a free Modelica standard library that is popular among its users.

The concrete design of Modelica builds upon predecessors such as Dymola [20] and Omola [21]. Since 1997, Modelica became a de-facto standard and the non-profit Modelica Association refines the design of the language by a committee of experts. Whereas these experts do their job to the best of their knowledge, no systematic studies have been performed on the user-base regarding readability, learnability and understandability. The success of many design decisions has hence never been appropriately measured. This paper is hence a first approach to this subject.

3. Literature summary

In this section, results of previous research undertakings are presented. Most of the mentioned papers examined programming languages. The survey is especially focused on results that should (by the intuition of the author) also be relevant for equation-based modelling languages. Despite the quantity of papers growing a lot since around the year 2000[1], it was found that the percentage of interesting results is larger in older

[1] A search on Google Scholar for the term `(software OR programming) AND (usability OR comprehension OR ergonomics OR psychology)` found 39.500 results for the years 1975 to 1985, 141.000 results for the years

papers. Therefore, they make up the majority of mentioned sources.

When evaluating programming languages, analysing simple tasks like writing a single line of code is not adequate. Broad-brush analysis is more suitable, because it avoids "death by detail" [5].

For mechanical problems, different representations (diagram vs. symbolic) can carry the same information but differ greatly regarding cognitive effort [22].

A notation cannot highlight every aspect of information at once, but a good notation makes obscured (non-highlighted) information more visible [23]. This is especially relevant when looking at the ongoing discussion regarding custom annotations in Modelica [24].

Programs are represented mentally at a higher level than just code [5]. The representations are different for programs written in Basic, LabView and spreadsheets [25]. Mentally absorbing information is much easier (and more performant) if there is a match between the external representation and the mental representation [26].

In the work of Sporer and Soloway [27], two folk wisdoms about beginner mistakes are tested. The first one - "just a few types of bugs can account for a majority of the mistakes in students programs" - seems to hold true. The second one - "most bugs can be attributed to student misconceptions about language constructs" - does not seem to correspond to reality. Instead, bugs seem to arise as a result of plan composition problems, or, "difficulties in putting the pieces of a program together". The second result was later confirmed in [28].

In [29] it was postulated that the biggest difference between novice and expert programmers is the amount of strategies (for planning or debugging) that are available to the expert. From this it can be derived that an ideal modelling language should make at least some planning- and debugging strategies available to the user. However, in [30] it is mentioned that experts talk more in terms of abstractions while novices tend to concern themselves with single words of code.

Low-level primitives are a potential cognitive barrier for programming novices ([31] as cited in [32]). Users are happy, if the language primitives map directly onto their problems, instead of having to build high-level constructs from low-level primitives themselves [33].

Newer research seems to suggest that at least some developers try to avoid comprehension of programs whenever possible, for example by copy-pasting code snippets that are known to work [34]. In the same work it was also observed that developers tried to put themselves in the role of the end-user whenever possible.

Programming ability does not seem to be correlated with age, sex or educational attainment. The effect of general intelligence is unclear, with some studies showing a positive correlation, while others do not. Some known predictors are consistency during the solving of ambiguous problems, possession of a mechanical mental model of computers[2] and expressed self-confidence before the first programming course [35][36][37].

1985 to 1995, 874.000 results for the years 1995 to 2005, 781.000 results for the years 2005 to 2015.

[2]Some participants tried to be gentle to the computer, touching the computer mouse softly. They hoped that the computer would return the favor by doing what they wanted. This would be the opposite of a mechanical mental model.

4. Explorative expert surveys

As seen in section 3, a large amount of research has been conducted on cognitive aspects of programming languages. The same cannot be said for equation-based modelling languages, where nothing similar has been published to the best knowledge of the author. For an initial stake-out of this new field, a series of structured surveys was conducted.

4.1. Method

Five voluntary participants were recruited from the colleagues of the author at the German Aerospace Center (DLR) Institute of System Dynamics and Control and connected working groups. Among participants three were male and two female. The participants were aged between 22 and 50 years. They had previous experience with Modelica between 2 and 180 months. As an introduction, the following statement was given:

"We are doing an explorative expert survey, where experts for object-oriented modelling of differing experiences are taking part. We want to know how people are approaching modelling tasks, and how modelling languages or tools are helping or constraining them. We invited you to this survey since you are using the equation-based modelling language Modelica productively. There are 15 questions to this topic. Later we filter out interesting aspects from your answers, possibly translate, and summarise them. For this reason there are no right or wrong answers. Feel completely free to skip as many questions as you like."

Participants were then asked if they wanted to conduct the survey in writing by filling out a word-document or verbally, including a recording of the interview. All participants chose the written version. The questions given to the participants were as follows:

(1) How did you obtain your Modelica knowledge?
(2) Which kinds of tasks do you use object-oriented modelling languages like Modelica for?
(3) Which aspects of the Modelica language took a long time to understand? Which aspects of the language do you still not understand?
(4) What are the first steps you take while approaching a modelling task?
(5) How do you split up your time for different parts of your modelling tasks? [concept, development, validation, documentation, productive usage and maintenance]
(6) What do you like/dislike about the Modelica tools that you know?
(7) Do you use a certain coding style?
(8) Do you know some style templates?
(9) What do you like/dislike if you have to use code developed by other people?
(10) When developing big models, how do you test intermediate results?
(11) When developing big models, how often do you test intermediate results?
(12) How much do you trust models that you have developed yourself?
(13) How much do you trust models that are developed by others?
(14) What is needed for you to gain trust?
(15) Is there anything else regarding this field that you want to share?

4.2. Selected results

(1) Experienced participants mostly got their knowledge from colleagues, and from learning by doing. Novices mostly relied on books.
(2) Participants are using Modelica for simulation of physical systems, development and tuning of control systems, derivation of linearised models (using the *Linear-Systems-library* [38]) and simulation of failure cases.
(3) The relationship between text- and graphical modelling, connecting vector- and matrix-based blocks, the replaceable/redeclare functionality, event-handling and embedding of C-code were seen as obstacles during the learning phase. None of these points except replaceable/redeclare was mentioned by more than one participant.
(4) While novices start new modelling tasks by *divide-and-conquer* strategies, experts often start with a simple model which is later augmented.
(5) Expert modellers refused to give numbers, based on the case dependency of the question. Novices gave wildly varying numbers: 1-20% for the concept phase, 19-45% for development phase, 15-70% for validation, 5-20% for documentation, and 0-5% for both productive use and maintenance.
(6) [there were no usable answers to this question]
(7) Participants did not know any formalised coding style. One participant remarked the usage of a recurrent structure: import, public parameters, protected parameters, public models, protected models, equations, connects, documentation. The same participant noted that the use of graphical modelling makes structuring of the code layer difficult.
(8) One participant mentioned the *DLR implementation guidelines for Modelica libraries*, the remaining participants did not know of any style guides.
(9) Almost all participants positively mentioned documentation and comments to help with the understanding of models developed by other people. They also liked structured libraries, a consistent coding style, and examples. Two participants strongly disliked a mixture of graphical and textual modelling in a single component.
(10) The usual strategy reported by novices and experts alike is to develop *Wrapper*models where the actual model is subjected to typical boundary conditions, or, if available, test data. This wrapper model also serves as a usage example for others.
(11) Most modellers test their models every time a submodel, or a new physical functionality, is implemented. Only one participant had another strategy, where the complete system is developed before the testing-phase starts.
(12) Modelica developers had medium to high levels of trust regarding their own models. There does not seem to be a correlation between trust in the own models and modelling experience.
(13) Trust regarding models from other developers seems to be mixed. Model source (colleagues vs. anonymous website) and code quality were mentioned as influencing factors.
(14) The following features were mentioned as trust-gaining factors: code clarity, quality of documentation, availability of test cases, availability of validation data, discussions between associates.
(15) [there were no usable answers to this question]

5. The cost of inheritance

There is a reason why inheritance is less popular in equation-based languages than in conventional programming languages: typically in equation-based models, only the leaves of the inheritance tree are directly usable and concrete components. The inner elements of the tree mostly are all abstract and not ready for use. This forms a contrast to programming languages where often also the inner elements of the tree represent concrete and fully functional classes. Inheritance would probably be less popular among programming languages if there would be only abstract father-classes. On the other hand, inheritance introduces inherent intricacy to a model [39].

The situation could be improved for equation-based languages if not only variables, or parameters can be added but also changed (or even removed). In this way, also code from concrete, usable components can be reused.

Nevertheless, an experiment was set up to quantify the extra cognitive load derived from inheritance in Modelica models.

5.1. Method

5.1.1. Design

Subjects were asked to open and study the code of models from the Modelica Standard Library (MSL) until they would feel comfortable using them in bigger models. The models were selected such that they did not contain relevant graphical annotations (as in: no composite models that where connected on the graphical layer) and varied in complexity. Subjects were told to record the time they needed to understand the model in the sense explained above. In particular, they were asked to also study inherited models.

The Modelica models selected for this study varied in complexity according to three factors: (1) number of symbols, (2) number of equations, and (3) total number of *extends* clauses used in the model. The number of symbols was counted using Word after deleting the main *annotation* blocks containing the model documentation and excluding whitespace. The subjects were told not to examine *extends* clauses leading to mere *Modelica.Icons* models.

The expectation was that each of these factors independently would lead to increased time to understand the models. Additionally, it could be expected that experts would understand complex models quicker than participants with less Modelica experience.

5.1.2. Participants

Twelve voluntary participants were recruited from colleagues at the DLR Institute of System Dynamics and Control. One of the participants only reported estimated timing measures and was therefore excluded from the study. Among the remaining 11 participants 10 were male and 1 female. The participants were aged between 22 and 35 years. They had previous experience with Modelica between 2 and 132 months (mean (M) = 49.36, standard deviation (SD) = 44.55).

5.1.3. Material

Ten models were used for the experiments as summarised in Table 1. The table contains the path in the MSL, the number of symbols *sym*, the number of equations *eq*, and

the number of inheritance levels *lvl*. The models were listed in random orders for each participant.

	Complexity			Time to understand	
Component Name	sym	eq	lvl	M[s]	SD[s]
Blocks.Discrete.FirstOrderHold	1514	7	2	269	62
Blocks.Continuous.SecondOrder	1285	2	1	94	21
ComplexBlocks.ComplexMath.Add	584	2	1	55	10
ComplexBlocks.Sources.ComplexStep	738	2	2	83	9
Electrical.Analog.Basic.Capacitor	634	6	1	58	16
Electrical.Analog.Ideal.IdealDiode	1452	10	2	389	89
Mechanics.Rotational.Interfaces.Flange_a	171	0	0	26	5
Mechanics.Translational.Components.Mass	1071	7	1	91	21
Thermal.HeatTransfer.HeatCapacitor	489	4	0	59	15
Thermal.HeatTransfer.Sources.PrescribedHeatFlow	566	2	0	59	9

Table 1. Models used in the experiments

5.1.4. Procedure

The experiments were conducted in the office of each participant. We asked the participants to open a window of the *Dymola* tool and load the MSL. We then presented them with the list of models and asked them to examine one model after the other until they would understand them. The participants were asked to take the time it took to understand each model and were then left alone. After the experiment, the results were collected and the intention of the study was explained. In the following discussions, some subjects reported problems during the experiment, such as being interrupted or not having understood a model at all. The timing of these models were subsequently excluded from the analysis. This resulted in 102 considered cases out of the 110 complete measurements.

5.2. Results

The time to understand a model varied between 10s and 567s (M = 96.30s, SD = 94.64s) for the valid measurements, indicating a wide range of considered model complexity. Averaged over all measurements for each participant, the mean time to understand a model varied between 65.3s and 213.5s (M = 121.39s, SD = 56.29s). This variation will be explained mostly by effects of the participants' experience in section 5.2.2.

5.2.1. Correlation of the independent variables

Since all three independent variables measure the complexity of the model in a way, they are naturally heavily correlated ($r > 0.53$) and they all increase the time it takes to understand a model. In order to better condition the regression problem, the number of equations was not used as an indicator of the overall model complexity. Preliminary analyses had shown the influence of the number of equations on the time to understand a model to be significant but small, when correcting for the other two factors.

5.2.2. Controlling for experience

In order to check the assumption that experience influences the time to understand, a regression analysis is conducted according to

$$time = a + b_{exp} \cdot exp + b_{sym} \cdot sym + b_{\times} \cdot exp \cdot sym, \tag{1}$$

fitting the time to understand on the participants' experience ($b_{exp} = 0.191$ s/month, $t(98) = 0.6$, $p = 0.58$), the number of symbols in the model ($b_{sym} = 0.200$s/1, $t(98) = 8.1$, $p < 0.001$) and, especially, the interaction of both variables ($b_{\times} = -0.001$s/month, $t(98) = -2.9$, $p = 0.005$).

Note that values of $p < 0.05$ indicate a statistically significant contribution of the respective predictor variable with effect size b. In order to obtain this significance level, $t(n)$ tests in $n = 98$ degrees of freedom are conducted. The tests compare each effect size b against its associated standard error. Assuming an effect size of zero and normally distributed measurements (null hypothesis), the probability to observe the actual effect size b (or stronger) is given by p. That means, small values in p indicate that the null hypothesis has to be abandoned and that the observed effect is indeed statistically significant.

The regression results show that each additional symbol contributes with $b_{sym} = 0.2$s to the time to understand a model. Additionally, this contribution decreases with growing experience ($b_{\times} < 0$). The main effect of the experience is not statistically significant in this analysis. In the following, in order to account for the interaction between experience and number of symbols, the number of symbols is regarded a random effect pertaining to repeated measures on the subjects (see section 5.2.3).

Figure 2 qualitatively illustrates the effect of experience on the time to understand. The mean time to understand a model decreases with growing experience. The same pattern is observed for the influence of the number of symbols on the time to understand. This indicates that the differences in mean time to understand a model are indeed caused by the complexity of the model, and not by a main effect of experience. Recall that the mean number of symbols in this study is 850.4 (see Table 1). This number corresponds roughly to the ratio of mean time to understand and the symbol effects.

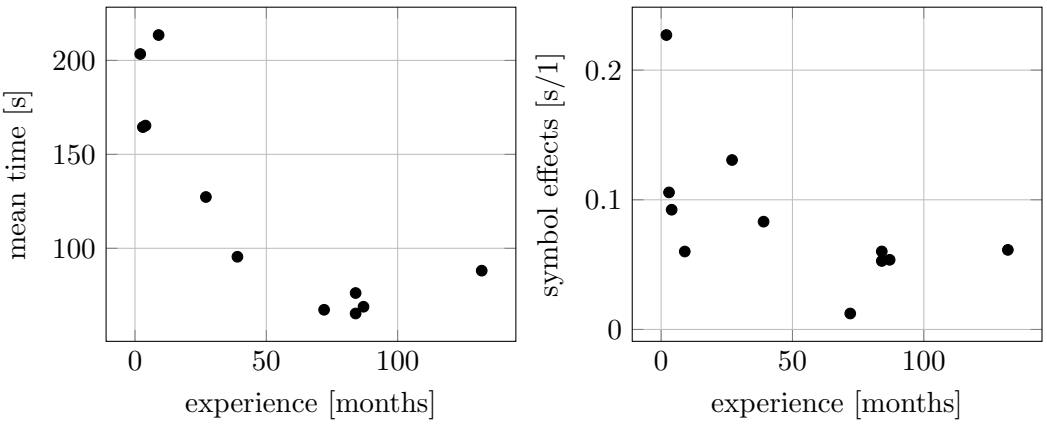

Figure 2. Random effects as a function of subjects' experience

5.2.3. Effect of hierarchy levels

The effect of hierarchy levels within a model on the time to understand is estimated using mixed-effects regression with random effects of the number of symbols according to

$$time = a + b_{lvl} \cdot lvl + b_{sym}(exp) \cdot sym.$$

In this model, a first regression is conducted for each participant, estimating the random effects $b_{sym}(exp)$ which are different for each participant. These are the values shown in Figure 2. In the second step, a regression on the residual data set is conducted in order to estimate the fixed effect b_{lvl} which is assumed fixed for all participants. The fixed effect of hierarchy levels is found significant in this analysis ($b_{lvl} = 26.65\text{s}/1, t(90) = 2.8, p = 0.006$). A fixed offset is again not found to be statistically significant ($a = -1.25\text{s}, t(90) = -0.1, p = 0.9113$).

5.3. Discussion

We could provide some amount of experimental evidence for one particular piece of the given modelling advice: the significant effect of hierarchy levels on the time to understand a Modelica model supports the hypothesis that flat hierarchy levels are easier to understand than deeply nested model hierarchies.

However, this study can merely be regarded as a pledge for more evidence based support of modelling strategies. There are many things, which are not addressed in the experiments. First, many variables typically controlled in psychological experiments are not regarded, such as gender or age. (Admittedly, at least the first variable is difficult to cover in engineering environments.) Second, the experiment was not conducted in a controlled environment. For example, participants reported very different procedures for opening extended classes, suggesting that the additional time per *extends* clause is partly due to finding the extended model in the MSL. Some development tools like Dymola offer a (albeit hidden) functionality to flatten a model and represent the variables and equations in a single file (*flat Modelica*), but none of the participants knew about this functionality during the experiment. Third, Modelica models have many more properties, such as documentation and graphical annotations, which need to be considered for sustainable modelling advice. Finally, time to understand may be argued not to be the best indicator for good modelling style as compared to, e.g., finding errors or actually using the models.

6. Viscosity of representations

In section 3, several findings from literature are cited concerning representations of information or programs. To summarise those findings, mental representations of computer programs are different for various programming language, and programming performance is higher if there is a match between the external representation and the mental representation ([5], [22], [25], [26]). We postulate that a similar dependency exists in the field of physical modelling.

In equation-based languages, several different representations are common. This is also true within the scope of individual equation-based languages. For example, in Modelica it is possible to model a system by writing down the necessary physical

equations, or alternatively by joining together the necessary subcomponents graphically.

Based on this, an experiment was set up to quantify several performance metrics for different representations.

6.1. Method

6.1.1. Design

Participants were shown four different Modelica models, or representations, of physical systems. They were asked to *identify* those systems, *predict* the transient response, and *rate* their confidence for those models. The models were created in such a way, that they represented each physical system in four different ways. The time needed by the participants for the identification and prediction tasks was measured.

For each of the four physical systems considered, four different Modelica models were created, to a total of 16 different models. Participants were unknowingly split up into random groups of equal size. Each group was presented with one representation of each physical system. To eliminate first order training and carryover effects, a balanced latin square design[3] as described in [40] was used to assign representations to physical systems for each group of participants.

Effects of physical systems are not considered in this work, therefore the order of physical systems was the same for each group. The effect is thus indistinguishable from training effects and it is accounted for by the balanced latin square design.

The expectation was that the percentage of correctly executed tasks, as well as timings and reported confidence would vary between the different representations.

6.1.2. Participants

Since knowledge in Modelica was necessary for participation in the experiment, participants were recruited from the Modelica User Groups Sachsen, North America, Japan, Hamburg and Baden-Württemberg, as well as colleagues of the author at the Institute of System Dynamics and Control. Participants were not compensated. Over the course of one month, 98 participants took part in the experiment. From those 20 participants were excluded based on having not completed the tasks (5), unrealistic timings and reported technical problems (5), reports of being interrupted (9), or reporting a Modelica experience of zero (1).

Among the 78 remaining participants, 8 reported being female, 69 male, and 1 other. They were aged between 23 and 63 years (mean $M = 36.7$, standard deviation $SD = 8.7$). They had previous experience with Modelica between 0.1 and 17 years (mean $M = 5.2$, standard deviation $SD = 4.4$). Participants reported their professional background based on the provided categories shown in Table 2. Notably, most participants were engineers and there were no computer scientists. Table 2 shows the place of living of the participants ordered by continents. Most of them lived in Europe or North America.

[3]If participants have to do multiple tasks, one after another, several effects are at work. One task might be more difficult than another, resulting in participants taking longer to complete this task. On the other hand, if the tasks are similar, participants will have some experience by the time they get to the last task, resulting in a shorter time to complete the task. Also, some of the tasks might be a good preparation for others, and vice versa. To isolate this effects, a balanced latin square design can be used. Here, the participants are divided into several groups. The groups are assigned to tasks in a specific order, designed to eliminate the experience (or training) and preparation (or carryover) effects under the assumptions that these effects are first order.

Mechanical Engineering or similar	43
Electrical Engineering or similar	11
Mathematics	9
Engineering (other)	8
Physics	3
Other	2
Science (other)	2
Computer science	0
(not answered)	1

Europe	51
North-America	14
Asia	12
(not answered)	1

Table 2. Origin and background of participants

systems		representations	
SD	a parallel spring-damper with connected mass	MSL	graphical models built from MSL components
OC	an electric resonant circuit without damping	EQ	models based on physical equations
T	two bodies in thermal contact	BLOCK	block diagrams built from the MSL.Block-package
BB	a mass bouncing on a floor	ALG	models implemented as algorithms

Table 3. Physical systems and representations used in the experiment

6.1.3. Material

For four different physical systems, four models were created each in Modelica, resulting in 16 models in total. The types of systems and representations are listed in Table 3.

Models were developed in such a way that they were behaving equally for each system, while keeping the models as simple as possible. All models for the Spring-Damper system are shown in Figures 3, 4, 5 and 6.

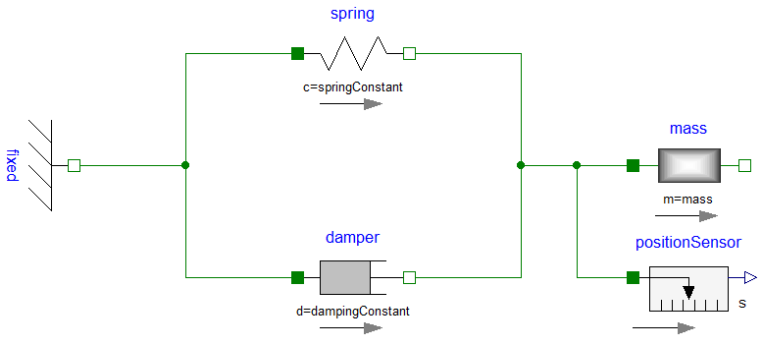

Figure 3. MSL - graphical model of a Spring-Damper

For each of the systems, eight names were created. Of those eight names, four were physically motivated, four were motivated from a system dynamics point of view. From each group of four, one of them correctly described the system. For example, the

```
model Test1
  parameter Modelica.SIunits.Length          position0;
  parameter Modelica.SIunits.DampingConstant dampingConstant;
  parameter Modelica.SIunits.SpringConstant  springConstant;
  parameter Modelica.SIunits.Mass            mass;

  Modelica.SIunits.Length       position;
  Modelica.SIunits.Velocity     velocity;
  Modelica.SIunits.Acceleration acceleration;
equation
  //force balance
  0 = -    acceleration      * mass
      -      velocity        * dampingConstant
      - (position-position0) * springConstant;
  der(velocity) = acceleration;
  der(position) = velocity;
end Test1;
```

Figure 4. EQ - equation-based model of a Spring-Damper

```
model Test2
  parameter Modelica.SIunits.Length          position0;
  parameter Modelica.SIunits.DampingConstant dampingConstant;
  parameter Modelica.SIunits.SpringConstant  springConstant;
  parameter Modelica.SIunits.Mass            mass;

  Modelica.SIunits.Length   position;
  Modelica.SIunits.Velocity velocity;
  Modelica.SIunits.Force    force;
algorithm
  force         := -      velocity       *dampingConstant
                   - (position-position0)*springConstant;
  der(velocity) := force/mass;
  der(position) := velocity;
end Test2;
```

Figure 5. ALG - algorithm-based model of a Spring-Damper

Spring-damper system was assigned the following names (correct names written in bold):

(1) **parallel spring-damper with connected mass** (2) force amplifying system with damping (3) electric resonant circuit without damping (4) operational amplifier (5) **second-order system with damping** (6) first-order system with connected integrator (7) proportional, integral, and derivative systems connected in parallel (8) non-linear second-order system without damping.

Furthermore, for each of the systems, nine images were created that illustrated possible trajectories. Of those trajectories, only one had an adequate connection to the system.

Figure 7 shows the possible answers for the spring-damper system (correct answer framed).

6.1.4. Procedure

The experiment was conducted online using *SoSci* survey [41]. Tasks were implemented as multiple-choice tests. Timings were measured using asynchronous java-script and extended mark-up language (XML) with an accuracy of a few milliseconds, independently of the internet connectivity.

At the start of the experiment, participants were shown a starting page, and were

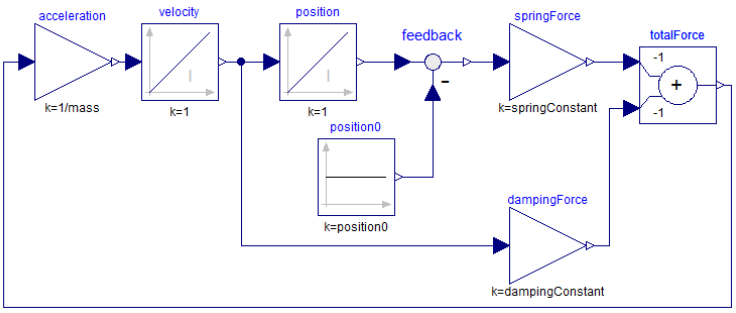

Figure 6. BLOCK - block-diagram model of a Spring-Damper

Figure 7. Answer options for the transient responses of the damped spring-damper system (correct answer framed)

unknowingly assigned to one of four groups. Assignment was based on the number of completed experiments for all groups.

Before each task, participants were given a short introduction to the task. For the first two tasks, this introduction also included a request to *"hurry up"*.

During the first task of the experiment, participants were shown four different combinations of systems and experiments and had to identify the systems in questions. The available answers for each combination were scrambled randomly. The assignment of the 16 combinations of systems and representations was done using a balanced latin square approach in the representations and constructed as shown in [42]. The assignment can be seen in Table 4.

	subtask 1	subtask 2	subtask 3	subtask 4
group 1	OC-BLOCK	T-EQ	SD-MSL	BB-ALG
group 2	OC-EQ	T-ALG	SD-BLOCK	BB-MSL
group 3	OC-ALG	T-MSL	SD-EQ	BB-BLOCK
group 4	OC-MSL	T-BLOCK	SD-ALG	BB-EQ

Table 4. Assignment of system/representation-combinations to groups

During the second task, participants were shown the same four combinations. This time, they were asked to predict the transient response. Again, the order of answers were scrambled.

During the third task, participants were again shown the same four combinations in the same order. They were asked how much they agree with the following statement: *"I would feel confident using this model."* Participants could choose from a five-point *Likert* scale[4], ranging from *"-2 (Strongly disagree)"* to *"+2 (Strongly agree)"*. This

[4]A Likert scale is a psychometric scale to quantify responses. If participants are asked: "How much do you love your dog?", some answers might be: "a lot" or "with all my heart" or "meh...". Mapping these answers to a love-metric will be highly arbitrary at best. Here, Likert scales should be used instead. The better question will be: "How much do you agree with the following statement: I love my dog.", while participants can choose from three, five or seven (fun fact: the human brain cannot meaningfully differentiate between more than seven

time, the order of answers was not scrambled.

For each task, the first subtask does not give valid timing results since the browser has to load the content first, and the participant has to understand the style of question. Therefore, for each task, an additional training subtask was added at the beginning. For example, at the beginning of the identification task, participants were shown a picture of bananas, together with the following question: *"Let us start with a small warmup question to get into the mood: What system is represented here (click one of the 2 correct options)?"*. Possible answers were: *"bananas"*, *"apples"*, *"yellow fruits"*, *"green fruits"*.

Performance in any kind of task can be effected by so called *stereotype threat*[5]. To lower the effects of stereotype threat on the results of this study, the actual tasks were executed by the participants at the beginning of the experiment. Only after conducting all three tasks (identification, prediction of transient response, rating confidence), personal information about the participants was gathered.

Participants were asked if they were interrupted during the experiment. They were also asked to estimate their Modelica experience in years, their professional background (dropdown-menu), age, gender and place of living (dropdown-menu with continents). Finally, they could make a guess regarding the goal of the experiment and could give miscellaneous remarks. Interested participants could also leave their E-Mail address to be notified of the experiments results, those addresses were stored independently, for the results to remain anonymous.

6.1.5. Internal review

Models were individually reviewed and modified by Alexander Pollok, Andreas Klöckner and two additional colleagues at the Institute of System Dynamics and Control, to keep code and model quality equally high for each of the 16 models.

Both system names and trajectories were checked for ambiguity by the same people. In some cases the problem definition was augmented to ensure that each problem had a unique solution, for instance by adding information about the sign of a damping constant.

The complete experiment was run as a pretest by eight colleagues of the author to ensure functionality and comprehensibility. The data from those experiments was discarded.

Participants involved in pretests and internal reviews were compensated generously with cookies.

6.2. Results

The basic answer characteristics for the three main tasks are shown in Table 5. There are no significant differences in error rates of identification ($\chi^2(3) = 0.69, p = 0.875$) nor

grades of a single concept) predefined answers, ranging from "agree" to "disagree". Odd numbers are used, because a neutral response has to be included. These answers are then translated to numbers. Often, equal distances between two neighboring answers are assumed. These numbers constitute the new attribute, which is far easier to analyse.

[5] if participants are under the impression that a sociocultural group they belong to is negatively stereotyped for that particular task, performance gets distorted to fit that stereotype [43, 44, 45]. For instance, women underperformed men in a math test, when the test got described as producing gender differences; the difference could be eliminated when the test got described as not producing gender-difference [46].

	identification type		identification errors		prediction errors		rating	
	Math.	Physical	Correct	Wrong	Correct	Wrong	Mean	SE
MSL	10	68	59	13	61	13	0.87	0.16
EQ	25	53	54	9	51	14	0.43	0.16
ALG	19	59	64	10	61	11	0.27	0.15
BLOCK	24	54	65	11	57	17	−0.31	0.15

Table 5. Summary of answers grouped by type of representation

	MSL	EQ	ALG
EQ	0.043	—	—
ALG	0.399	1	—
BLOCK	0.058	1	1

Table 6. Identification answer types post-hoc p-values with Holm correction[47]

prediction ($\chi^2(3) = 1.74, p = 0.627$)[6]. The type of answer (mathematical or physical) in the identification task has significant differences over the four representations ($\chi^2(3) = 9.64, p = 0.022$). The p-values of the pairwise differences are assessed in Table 6. In summary, the *MSL* representation leads to significantly more physical answers as compared to representation as equation (*EQ*). Table 5 also shows the mean rating estimates and standard errors for each representation. Models are rated best, if they are built with MSL, and worst, if they are built as block diagram. Details on these values are described later in this section.

The influence of the type of answer in the identification task on the other measures is shown in Table 7. Holm correction[47] is used[7]. There are no significant effects of the answer type on identification correctness ($\chi^2(1) = 3.80, p = 0.051$) nor prediction correctness ($\chi^2(1) = 0.24, p = 0.627$). Estimated rating means are included for illustration. There is no significant effect on these either.

The reaction times and ratings are subjected to a mixed-effects regression with

[6] $\chi^2(4-1)$ is a metric to test the statistical independence between two attributes. Here, the independence of the error rate and the representation is tested. The smaller the number, the more probable it is that both attributes are independent.

[7] If multiple hypothesis are tested against a given significance level, the risk of having at least one false positive is automatically increased. Holm correction is used to counteract this effect by adjusting the p-values.

	identification		prediction		rating	
	Correct	Wrong	Correct	Wrong	Mean	SE
Mathematical	77.3 %	22.7 %	78.1 %	21.9 %	0.37	0.11
Physical	87.6 %	12.4 %	81.6 %	18.4 %	0.26	0.15

Table 7. Summary of answers grouped by type of representation

		identification			prediction			rating		
	df	res	F	p	res	F	p	res	F	p
(Intercept)	1	202	599.4	0.000	188	462.3	0.000	202	44.3	<0.001
repr.	3	202	12.8	<0.001	188	1.1	0.343	202	15.2	<0.001
id_type	1	202	3.4	0.068	188	30.3	<0.001	202	1.2	0.284
id_correct	1	202	1.4	0.238	188	5.5	0.020	202	8.3	0.004

Table 8. Test statistics of linear regression models for the identification, prediction, and rating tests

random intercepts

$$time = a + b(participant) + b(representation) + b() + b(id\_correct) \tag{2}$$

where a represents a fixed mean intercept, $b(participant)$ represents an additional intercept for each participant, and $b(representation)$, $b()$
as well as $b(id\_correct)$ represent fixed effects of representation, type of answer in the identification task as well as correctness of the identification task. Table 8 shows the respective significance tests. There are significant effects for all fixed intercepts, for the influence of representation on identification reaction time ($F(3, 202) = 12.8, p < 0.001$)[8] and on rating ($F(3, 202) = 15.2, p < 0.001$).

There is no significant difference for representation forms in the prediction task ($F(3, 188) = 1.1, p = 0.343$). This could be due to the same model being presented twice, such that the reaction time is not influenced anymore by the representation of the model, but rather by the classification result from the prior task. This is supported by the finding that the type of answer from the naming task has indeed a significant influence on the timing for the prediction task ($F(1, 188) = 30.3, p < 0.001$). A physical answer type leads to a 8.8 s quicker response than a mathematical answer type ($t(188) = 5.11, p < 0.001$). In addition, a correct answer in the identification task carries over to the reaction time in the prediction task ($F(1, 188) = 5.5, p = 0.020$), indicating a more confident reaction in the second task for correct answers. This is confirmed by a significant effect from identification correctness on the actual rating ($F(1, 202) = 8.3, p = 0.004$).

Estimated means of the fitted models per representation type are shown in Table 8 along with their 95% confidence interval and letter displays for non-significant differences[9]. They are computed using the fitted values and averaging over participants, the different types of identification answer, and the correctness of the identification answer. Letter displays indicate groups, in which no significant differences between the representation forms are found. For example, there is no significant difference in the identification reaction time between MSL representation and EQ representation. But there is a significant difference between MSL representation and ALG representation.

[8] $F(4-1, 203-1)$ is a measure for the variance of the 203 results between the 4 groups, compared to the variance of the results inside of the groups. If this number is high, the variance between the groups is comparatively high. This implies, that the groups are a good predictor for the data.

[9] Letter displays indicate groups, in which no significant differences between the representation forms are found. For example, there is no significant difference in the identification reaction time between MSL representation and EQ representation. For this reason, they both contain the letter 1. There is a significant difference between MSL representation and ALG representation. For this reason, they do not have a letter in common.

Table 9 shows the values of these differences along with the respective test statistics.

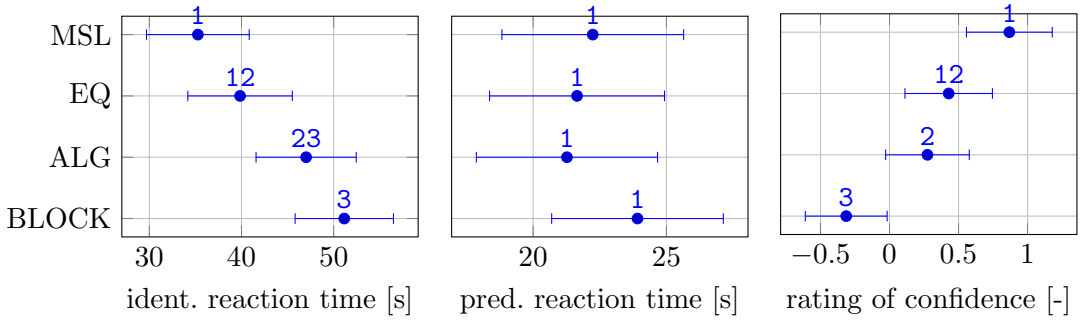

Figure 8. Predicted reaction times and rating with 95%-confidence intervals and letter displays for non-significant differences.

reaction times for naming					
contrast	estimate	SE	*df*	*t*	*p*
MSL - EQ	-4.58	3.10	202	-1.47	0.455
MSL - ALG	-11.75	2.90	202	-4.05	<0.001
MSL - BLOCK	-15.88	2.91	202	-5.46	<0.001
EQ - ALG	-7.17	3.01	202	-2.38	0.084
EQ - BLOCK	-11.30	2.99	202	-3.79	0.001
ALG - BLOCK	-4.14	2.84	202	-1.46	0.466
rating of preference					
contrast	estimate	SE	*df*	*t*	*p*
MSL - EQ	0.44	0.19	202	2.30	0.101
MSL - ALG	0.59	0.18	202	3.30	0.006
MSL - BLOCK	1.18	0.18	202	6.58	<0.001
EQ - ALG	0.15	0.19	202	0.83	0.842
EQ - BLOCK	0.74	0.18	202	4.03	<0.001
ALG - BLOCK	0.59	0.19	202	3.34	0.005

Table 9. Predicted differences between forms of representation for reaction times and rating

The graphical MSL representation performs best in both the identification and in the rating task. The *BLOCK* diagram representation performs worst. There are no significant differences between the EQ equation and the ALG algorithm representations. This can likely be explained by the two representations resembling each other strongly.

6.3. Discussion and Conclusion

Modern modelling languages give experts a large amount of freedom regarding the choice on how to represent a physical system. It was found that this choice has a

significant impact on the performance of modelling experts for varying tasks.

Based on the conducted experiments, graphical representations using the MSL should be preferred in most cases. Tasks were performed faster, and participants showed a higher amount of confidence. The opposite can be said for the representation by block diagrams. Achievable time savings can be rather important. A difference of up to 15.88 s was found between MSL and BLOCK representations. This corresponds to roughly one third of the absolute reaction time. Maximal rating difference was 1.18 between the same two representations, which also corresponds to one third of the total scale used for rating the model.

Textual models, represented by equations or by algorithms, can be grouped between graphical MSL representations and block diagrams. No significant difference was found between these textual models. While equation-based representations might be more common in the Modelica community, the final choice will usually depend on the use case. For example, algorithms might be used more often when controllers are modelled, or when non-physical computations are represented.

Interestingly, the amount of errors was similar for all representations, while the necessary time was significantly different. This suggests that modelling experts completely compensate for tasks of higher difficulty by just taking more time. This is consistent with general finding on the speed-accuracy tradeoff and also manifests in the amount of confidence expressed explicitly.

While no direct influence of type of representation was found on the reaction time for the prediction task, there were significant influences mediated by the mental representation of the model. This can be counted as evidence towards more subconscious confidence in a physical than in a mathematical mental model. Another mediator variable is the correctness of the identification answer, which has significant influences not only on subconscious measure of prediction time, but also on the consciously expressed confidence in the model. Together, these results also suggest to prefer physical MSL models over equation-based or block-based models, in order to increase the confidence of users in the model and to increase the ease of working with these models.

6.3.1. External Validity

Results of these experiments are restricted to engineers and other users of equation-based modelling languages. This is due to the recruiting of the participants from the Modelica user groups. However, from an industry standpoint, this is also the group of people where such results are useful.

While the individual timings of the experiment will surely be dependent on the background of the participants, the study design should compensate any such effects for the derived results. Even if a mechanical engineer will be faster at recognizing a mechanical spring-damper system compared to a computer scientist, this will be true for any of the different representations, which are compared relatively to each other.

Results hold mainly for the western culture. Although a few asian participants took part in the experiment, there is no reasonable evaluation as to what differences exist between these cultural groups. However, no such differences would be expected anyway.

For reasons of practicality, the used models were quite small. Due to the lack of similar studies, not much is known about the effects of model size and complexity. It is completely plausible that the best representation is dependent on the size and complexity of the system. Further studies are necessary to make generalizations in that direction.

Many other typical threats to external validity should not be relevant for this study. While participants might show increased performance due to the awareness of being observed (Hawthorne effect), this increase will be in effect for all of the tasks. Since only relative statements are derived from the resulting data, the Hawthorne effect should not decrease the external validity of this study. Similar arguments can be made for pre- and post-test effects, situational specifics, or Rosenthal effects (higher expectations lead to increased performance). Second order effects can not be ruled out, but their influence is expected to be small.

7. Acknowledgments

We thank the Modelica User Groups Saxony, North America, Japan, Hamburg and Baden-Württemberg for their help with the recruiting of participants. We thank all participants of the experiments for their support and time.

References

References

[1] CA Hoare. Hints on programming language design. Technical report, DTIC Document, 1973.

[2] Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983.

[3] Françoise Détienne and Frank Bott. *Software Design–Cognitive Aspect*. Springer Science & Business Media, 2002.

[4] Jon A Turner and Robert A Karasek Jr. Software ergonomics: effects of computer application design parameters on operator task performance and health. *Ergonomics*, 27(6):663–690, 1984.

[5] Thomas R. G. Green and Marian Petre. Usability analysis of visual programming environments: a cognitive dimensions framework. *Journal of Visual Languages and Computing*, 7(2):131–174, 1996.

[6] Steven Clarke. Evaluating a new programming language. In *13th Workshop of the Psychology of Programming Interest Group*, pages 275–289, 2001.

[7] James Howatt. A project-based approach to programming language evaluation. *ACM SIGPLAN Notices*, 30(7):37–40, 1995.

[8] Jarallah Al Ghamdi and Joseph Urban. Comparing and assessing programming languages: basis for a qualitative methodology. In *Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice*, pages 222–229. ACM, 1993.

[9] Daniel L Moody. The physics of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009.

[10] Keng Siau and Matti Rossi. Evaluation of information modeling methods-a review. In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, volume 5, pages 314–322. IEEE, 1998.

[11] Eva-Lena Lengquist Sandelin, Susanna Monemar, Peter Fritzson, and Peter Bunus. An interactive tutoring environment for modelica. In *Proceedings of the 3rd International Modelica Conference*, 2003.

[12] Alexander Pollok and Andreas Klöckner. The use of ockham's razor in object-oriented modeling. In *Proceedings of the 7th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pages 31–38. ACM, 2016.
[13] Dirk Zimmer. Equation-based modeling with modelica–principles and future challenges. *SNE*, page 67, 2016.
[14] Michael Tiller and Dietmar Winkler. Where impact got going. 2015.
[15] Peter Fritzson. Introduction to modelica.
[16] Modelica-Association. The Modelica Standard Library. *Online, URL: http://www.modelica.org/libraries/Modelica*, 2008.
[17] Paul Inigo Barton and CC Pantelides. gproms-a combined discrete/continuous modelling environment for chemical processing systems. *Simulation Series*, 25:25–25, 1993.
[18] Jan F Broenink. 20-sim software for hierarchical bond-graph/block-diagram models. *Simulation Practice and Theory*, 7(5):481–492, 1999.
[19] Steve Miller and T MathWorks. Modeling physical systems as physical networks with the simscape language. In *6th Vienna International Conference on Mathematical Modelling (MATHMOD 2009), Vienna, Austria, February*, pages 11–13, 2009.
[20] Hilding Elmqvist. Dymola-a structured model language for large continuous system. *Report TFRT*, 7175, 1979.
[21] Mats Andersson. Omola: an object-oriented language for model representation. 1990.
[22] Jill H Larkin and Herbert A Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive science*, 11(1):65–100, 1987.
[23] TRG Green, ME Sime, and MJ Fitter. The art of notation. *Computing skills and the user interface*, pages 221–251, 1981.
[24] Dirk Zimmer, Martin Otter, Hilding Elmqvist, and Gerd Kurzbach. Custom annotations: Handling meta-information in Modelica. In *Proceedings of 10th International Modelica Conference*, volume 10, 2014.
[25] Thomas RG Green and R Navarro. Programming plans, imagery, and visual programming. In *Human Computer Interaction*, pages 139–144. Springer, 1995.
[26] Iris Vessey and Dennis Galletta. Cognitive fit: An empirical study of information acquisition. *Information systems research*, 2(1):63–84, 1991.
[27] James C Spohrer and Elliot Soloway. Novice mistakes: Are the folk wisdoms correct? *Communications of the ACM*, 29(7):624–632, 1986.
[28] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin*, volume 37, pages 14–18. ACM, 2005.
[29] David J Gilmore. Expert programming knowledge: a strategic approach. *Psychology of programming*, pages 223–234, 1990.
[30] Thomas D LaToza, David Garlan, James D Herbsleb, and Brad A Myers. Program comprehension as fact finding. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 361–370. ACM, 2007.
[31] Clayton Lewis and Gary Olson. Can principles of cognition lower the barriers to programming? In *Empirical studies of programmers: second workshop*, pages 248–263. Ablex Publishing Corp., 1987.
[32] John Pane and Brad Myers. Usability issues in the design of novice programming systems. *Carnegie Mellon University, School of Computer Science Technical Report*, 1996.

[33] Bonnie A Nardi. *A small matter of programming: perspectives on end user computing*. MIT press, 1993.

[34] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. How do professional developers comprehend software? In *Proceedings of the 34th International Conference on Software Engineering*, pages 255–265. IEEE Press, 2012.

[35] Saeed Dehnadi and Richard Bornat. The camel has two humps (working title). *Middlesex University, UK*, pages 1–21, 2006.

[36] Peter Chalk, Tom Boyle, Poppy Pickard, Claire Bradley, Ray Jones, and Ken Fisher. Improving pass rates in introductory programming. In *4th Annual Conference of the LTSN Centre for the Information and Computer Sciences*, 2003.

[37] Sally Fincher, Anthony Robins, Bob Baker, Ilona Box, Quintin Cutts, Michael de Raadt, Patricia Haden, John Hamer, Margaret Hamilton, Raymond Lister, et al. Predictors of success in a first programming course. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, pages 189–196. Australian Computer Society, Inc., 2006.

[38] Marcus Baur, Martin Otter, and Bernhard Thiele. Modelica libraries for linear control systems. In *Proceedings of 7th International Modelica Conference*, pages 20–22, 2009.

[39] Michael Tiller. Parsing and semantic analysis of modelica code for non-simulation applications. In *Proceedings of the 3rd International Modelica Conference*, 2003.

[40] EJ Williams. Experimental designs balanced for the estimation of residual effects of treatments. *Australian Journal of Chemistry*, 2(2):149–168, 1949.

[41] Dominik J Leiner. Sosci survey (version 2.5.00-i)[computer software]. 2014.

[42] James V Bradley. Complete counterbalancing of immediate sequential effects in a latin square design. *Journal of the American Statistical Association*, 53(282):525–528, 1958.

[43] Claude M Steele. A threat in the air: How stereotypes shape intellectual identity and performance. *American psychologist*, 52(6):613, 1997.

[44] Ap Dijksterhuis and Ad Van Knippenberg. The relation between perception and behavior, or how to win a game of trivial pursuit. *Journal of personality and social psychology*, 74(4):865, 1998.

[45] Margaret Shih, Todd L Pittinsky, and Nalini Ambady. Stereotype susceptibility: Identity salience and shifts in quantitative performance. *Psychological science*, 10(1):80–83, 1999.

[46] Steven J Spencer, Claude M Steele, and Diane M Quinn. Stereotype threat and women's math performance. *Journal of experimental social psychology*, 35(1):4–28, 1999.

[47] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.