Reading Logic as Code or as Natural Language Text
A Randomized Controlled Trial Experiment on the Comprehensibility of Object-oriented Source Code in Comparison to Natural Language Text

Patrick Rein
patrick.rein@hpi.uni-potsdam.de
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

CCS CONCEPTS
• Software and its engineering → General programming languages; Collaboration in software development; • General and reference → Empirical studies.

KEYWORDS
program comprehension, source code, natural language text, domain-specific languages, controlled experiment, Smalltalk

ACM Reference Format:

1 INTRODUCTION
Creating useful software systems requires a solid understanding of the application domain of the system [1, 9]. Software development teams thus often communicate with domain experts to elicit domain knowledge and requirements for the system [4]. To capture the resulting shared understanding of the domain, software developers and domain experts create artifacts such as graphical models or glossaries [5]. Source code is another interesting format for capturing the shared understanding of the domain as source code files are the definitive documents for the actual behavior of a software system.

At the same time, source code is a technical artifact and thus might be unsuitable for the communication between non-programming domain experts and the software development team. Domain-specific languages and graphical modeling languages are often proposed to enable domain experts to participate in the modification of source code documents [2, 6, 10]. These approaches assume that general-purpose programming languages result in source code documents that are not accessible enough for non-programmers to work with them.

We conducted a randomized controlled trial experiment to investigate this assumption. In detail, we investigated the hypothesis that:

Given a problem domain with simple rules, people with little to no programming experience score worse on a text comprehension task given a Smalltalk program in comparison to an English text document.

We chose the object-oriented paradigm, as it is used for describing domain models [4]. We chose Smalltalk [7] as the programming language because its syntax was designed to be minimal and resemble English text.

The insights from our study are related to the empirical evidence gathered on the effects of language features on programming novices [3, 12]. Further, our results also relate to results from studies investigating the effects of a domain-specific language on the productivity of experienced programmers [8]. However, in contrast to these studies, we focus on non-programming domain experts which do not intend to become proficient programmers.

2 EXPERIMENTAL DESIGN
We conducted a fixed-setup study through a 2x2 factorial experiment [11]. The first factor was the format of the document: Smalltalk source code or English text. To mitigate carry-over effects between rounds, we introduced two scenarios as the second factor (see Figure 1). For each scenario, we devised ten questions. We designed the scenarios to be of equal complexity to reduce the noise introduced by the scenarios.

We operationalized the comprehension level as the number of correctly answered questions on the content of the scenarios and
We conducted the experiment through Amazon Mechanical Turk (MTurk). In total, we recruited 36 participants which had to hold a U.S. bachelor’s degree. We ensured proper engagement with the task through control questions. After discarding submissions from participants with prior programming experience and submissions which answered any control question wrongly, we had 31 participants.

3 RESULTS
As we employed a within-subject design, we used a paired t-test for comparing the results (individual data see Figure 3 and Figure 4). Therefore, we determined the mean of the differences between the scores for the Smalltalk and the English language document of each participant. The assumption of normality was met. The mean difference is $-1.42$ (standard deviation 1.747) and the difference between the scores is significantly different from zero ($t_{31} = 4.524$, $p < 0.001$). This difference means that on average participants scored fewer points on questionnaires about a Smalltalk document than they did on questionnaires about a text document.

To check our assumption that the two different scenarios do not influence the results, we also analyzed the difference between the scores for the conference registration and the shop checkout scenario. Again, assumption of normality for the differences was met. The mean difference between the two scenarios is not significantly different from zero ($t_{31} = 1.656$, $p = 0.108$).

3.1 Threats to Validity
We identified the following major threats to the validity of the experiment. An external threat is the assumption that readers have no programming background. In practice this might not hold for domain experts working with software developers. An internal threat to validity is the little amount of time invested by participants (see Figure 2). Another internal threat is the potential difference in task difficulty as suggested by outliers in the data (see Figure 4).

4 CONCLUSION AND FUTURE WORK
The results of our experiment suggest that object-oriented source code written in Smalltalk is less comprehensible to non-programming domain experts for the tested types of scenarios. At the same time the difference is less pronounced than we expected. Overall, the results should be regarded as preliminary, as outliers suggest remaining issues with the comparability of the scenario difficulty.

For future work, we will revise the experiment setup to eliminate any potential influence from the different scenarios so we can use the experiment as a baseline for further experiments into the comprehensibility of source code for non-programmers.
REFERENCES


