Exploring the Effects of Contextualized Problem Descriptions on Problem Solving

Juho Leinonen
University of Helsinki
Helsinki, Finland
juho.leinonen@helsinki.fi

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Jacqueline Whalley
Auckland University of Technology
Auckland, New Zealand
jwhalley@aut.ac.nz

ABSTRACT

Prior research has reported conflicting results on whether the presence of a contextualized narrative in a problem statement is a help or a hindrance to students when solving problems. On the one hand, results from psychology and mathematics seem to show that contextualized problems can be easier for students. On the other, a recent ITiCSE working group exploring the "problem description effect" found no such benefits for novice programmers.

In this work, we study the effects of contextualized problems on problem-solving in an introductory programming course. Students were divided into three groups. Each group was given two different programming problems, involving linear equations, to solve. In the first group both problem statements used the same context while in the second group the context was switched. The third group was given problems that were mathematically similar to the other two groups, but which lacked any contextualized narrative.

Contrary to earlier findings in introductory programming, our results show that context does have an effect on student performance. Interestingly depending on the problem, context either helped or was unhelpful to students. We hypothesize that these results are explained by a lack of familiarity with the context when the context was unhelpful, and by poor mathematical skills when the context was helpful. These findings contribute to our understanding of how contextualized problem statements affect novice programmers and their problem solving.

CCS CONCEPTS

• **Social and professional topics** → *Computing education.*

KEYWORDS

context, story problems, word problems, contextualization, novice programmers, problem representation, problem description, symbolic problems

ACM Reference Format:
Juho Leinonen, Paul Denny, and Jacqueline Whalley. 2021. Exploring the Effects of Contextualized Problem Descriptions on Problem Solving. In *Australasian Computing Education Conference (ACE '21), February 2–4, 2021, Virtual, SA, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3441636.3442302

ACE '21, February 2–4, 2021, Virtual, SA, Australia

ACM ISBN 978-1-4503-8976-1/21/02…$15.00
https://doi.org/10.1145/3441636.3442302

1 INTRODUCTION

The very first step in the problem solving process is understanding the problem. Prior research has shown that this step can be difficult for novices who sometimes form inaccurate conceptual models about the problem to be solved and then struggle to correct them [40]. One approach for addressing this is the use of interventions independent of the wording of the problem statement, such as requiring students to solve test cases before writing code [13]. In this research we focus on the presentation of the problem statement itself, and the context provided by its narrative.

In mathematics education problems that are presented with only minimal, essential, information are known as no-context, numeric, or symbolic problems. Such problems are presented in the form of mathematical symbols and equations. Problems that include context in their problem description in the form of a story or narrative are known as "story problems" or "word problems" [7]. Arguably, these "story problems" include unnecessary information that adds additional cognitive load to the problem solving process. In order to solve a story problem, an extra step in the problem solving process is necessary where relevant information must be extracted from the story. Conversely, it is possible that context can assist students in relating to a problem and thus help them understand what they are supposed to do in order to solve it.

A 2016 ITiCSE working group studied how context in the description of a problem can affect problem-solving in introductory programming courses [2]. Two modified versions of the Rainfall Problem [16, 43, 45] were used, one with context ("the Satellite Problem") and one without ("Just the Numbers"). Participants from six institutions were divided into two groups. One group was given the problem with context and the other the problem with no-context. There was no within-group analysis. Their results showed that adding context to a problem description had no statistically measurable effect on problem-solving. This finding was contrary to earlier findings in, for example, mathematics [29] and psychology [20] where it was found that context can be beneficial for problem-solving.

A follow-up study by Craig et al. [5] involved participants completing a set of five problems each with a context and no-context variant. The contexts used were familiar to the participants as the problem domains had been encountered in prior course work. Their results were mixed with students performing better on the contextualized variant for only one of the five problems. The authors concluded that any advantage that a familiar context might have

provided was hidden by other factors such as the use of complex terminology and the length of the problem description.

The vast majority of research into the effect of problem description and level of context on problem solving success is situated in the fields of psychology and mathematics education. It is generally accepted that computer science (CS) has unique characteristics when compared with other disciplines [42], and that therefore the results found regarding contextualization of problems in other domains may not apply to CS. The little work that exists within the computing education research literature on the effects of context is inconclusive or contradictory. Further research about the effect and its associated factors is therefore warranted.

A better understanding of the effect of context should lead to better designed programming assessments where context is used (or not) – a goal that is of clear interest to computing educators [10]. If context is used then it should be possible to present it in a way that assists rather than hinders learning and assessment outcomes. Work by Mason and Seton [34] examines how learning and assessment tasks in a database course can be adjusted to reduce cognitive load and we believe better understanding of aspects of context that affect problem solving could in a similar manner improve the presentation of programming tasks.

In this paper, we extend the work of Bouvier et al.'s 2016 ITiCSE working group [2] and that of Craig et al. [5]. In contrast to their studies, our participants were asked to complete more math-oriented programming tasks. We were interested in exploring whether it is easier for novice programmers to solve problems when they are provided with only the essential information, or does extra information that places the problem in a familiar context help? In order to study the effect of context we examine how changing the context between two sequential problems, in a single assignment, affects student performance and how this compares to a situation in which the context remains the same in both problems. We also investigate how context affects the type and number of mistakes students make in their programs. We postulate that when assignments are more math-oriented, as in our study, the results will be closer to previous results in mathematics education. Additionally, we hypothesize that changing the context between two problems will have an effect on how quickly and correctly students are able to solve the second problem. Specifically we aim to answer the following research questions:

RQ1. How does contextualization affect correctness?
RQ2. How does contextualization affect time on task?
RQ3. Does contextualization have an effect on the type and number of errors students make?

2 RELATED WORK

In order to understand the literature on the effect of context on problem solving a few definitions are necessary. Two types of problems appear in the literature: *story* (also know as word problems) and *no-context* problems. In mathematics, story problems are presented in the form of a short narrative rather than using mathematical notation or symbols. When compared with no-context problems (also known as symbolic, decontextualized, or numeric problems) story problems have context and the actual problem is often hidden within the details of that context. Koedinger and Nathan [29] introduced an additional type of problem called *word equations* to represent an intermediate mathematical problem type that does not include context but instead presents a verbal equivalent of a numeric problem. Examples of each type of problem are presented in Table 1. In this research, we use the term *no-context problem* to refer to a problem presented without a context narrative and *context problem* to refer to a problem presented with context in the form of a short narrative.

It is clear from the research presented in the literature that the way in which problems are presented influences problem-solving outcomes (e.g., [5, 7, 37, 41]). Minor variations in the linguistic elements used in a problem's description have been found to significantly impact task difficulty [2, 29].

In mathematics, problems can be presented in different ways including *result-unknown* and *start-unknown*. The problems in Table 1 are all presented as start-unknown problems in which the answer or result of the problem (i.e. John's total earnings) and the change values (i.e. hours worked and tips) are known but the starting value (i.e. John's hourly wage) is unknown. We know John has an hourly wage but the story does not tell us what it is. In contrast, if the story was presented as result-unknown the start and the change values would be known but not John's total earnings. Start-unknown problems, with and without context, are considered to be more difficult to solve than result-unknown problems [8, 9, 18, 23, 29]. Result-unknown problems usually are considered to be arithmetic whereas start-unknown problems are considered to be algebraic problems [29]. In this study the problems used include both start-unknown and result-unknown problems in the form of context and no-context problems.

Many papers discuss the advantages of context in problem solving tasks with respect to engagement [50], motivation [14, 15, 17], and retention [21]. Performance on context problems, which were used to measure problem solving ability, has also been found to be an early predictor of performance for novice programmers [38]. Here we are focused not on engagement per se but on the impact that context has on the ability of novice programmers to solve a problem.

2.1 Context, Cognition and Problem Solving

The process of solving context problems is believed to involve an additional phase over and above the problem solving process for no-context problems [6, 22, 29, 31]. This additional first phase is the *Comprehension Phase*. It is here that the story text is processed and the problem solver creates an internal representation of both the situational and quantitative relationships present in the text [37]. This linguistic processing involves the processing of different types of knowledge including, situational, verbal, and symbolic knowledge. Additionally, in this phase the parts of the story relevant to solving a problem need to be extracted from any irrelevant details provided in a context problem's presentation. In essence, the output of this phase is an abstract model of the situation and problem. No-context problems are actually presented as such models, for an example see the symbolic problem given in Table 1.

The second phase of problem solving involves the transition between the problem solvers' internal situation and models to the external representation of the solution to the problem. This phase

Table 1: Examples of Context Types for Start-Unknown Problems in Mathematics. Adapted from [29]

Story Problem	Word Equation	Symbolic Problem
John is a waiter. He worked 6 hours today and made $60 in tips. When he added his tips to the amount he earned today, he found he had earned $91.80. How much does John earn per hour?	Starting with some number, if I multiply it by 6 and then add 60, I get 91.80. What number did I start with?	Solve for x: $x \times 6 + 60 = 91.80$

is known as the *Solution Phase*. It is here that strategies used to process the relevant aspects of the problem are applied such as equation solving, and guessing and testing. Thus, in simple terms the overall process follows a translate-and-solve strategy in which the two phases are, in practice, interleaved and iterative. What is learnt in terms of internal and external representations in one cycle can go on to influence future comprehension cycles [28]. Koedinger and Nathan [29] give an example of this story problem solving process in the context of solving algebraic problems, "the problem text is first translated into written symbolic form and then the symbolic problem is solved" (p. 133). In CS, there is some anecdotal support for an additional comprehension phase in solving problems presented with context. Bouvier et al. [2] provide an example of a student who was uncertain about values of zero in the context of Soloway's rainfall problem [45] . To someone familiar with data from weather stations they would know that this meant the day had no rain. The authors note that "however awkwardly, this student was trying to make a connection between the numerical problem and the context" [2, p. 103].

This theory of solving problems presented in a context is prevalent in the literature and, along with Cognitive Load Theory [32, 46], is used to explain the difficulty of story problems. It has been argued that context increases cognitive load and results in a problem being more difficult. The argument goes that working memory is limited and context requires the consideration of irrelevant information resulting in a higher cognitive load [5]. On the other hand, it has been argued that context can improve problem solving if the context triggers an appropriate solution strategy [29].

In computer programming, a similar theory of problem solving has been proposed involving two types of knowledge: semantic and syntactic [44]. Semantic knowledge is used to translate a problem from its initial representation into a plan or model that represents a computer program. This knowledge is independent of the programming language [38]. This type of knowledge is involved in the Comprehension Phase of problem solving for context problems. According to Nowaczyk, "semantic knowledge is dependent on problem solving ability" [38, p. 149]. Syntactic knowledge is required to translate this plan into the symbols of the programming language. Syntactic knowledge is required during the Solution Phase.

In 2017, Bubnó and Takács presented a method for teaching school children learning computer programming how to solve context problems [3]. Their method employed Pòlya's model [39] for problem solving which maps well to current understandings of problem solving for word problems in mathematics and computer programming. The steps involved are: comprehend and understand the problem; devise a plan (create a model to solve the problem, or find the right algorithm); carry out the plan; and review/extend (reflect and test). The authors note that there are similarities between the steps used to devise and write a computer program to solve a problem and those used for solving mathematical problems.

While the literature in psychology supports the position that context is helpful in problem solving, the seminal literature in mathematics and early work in computer programming has had mixed results. In the next subsections we summarize the research in mathematics and computer programming based on their findings in terms of problem context being helpful, harmful, and benign.

2.2 Context is Helpful

Wason's selection task is a widely studied psychological test of logical reasoning that has been discussed in mathematics education (e.g. [24]). The original selection task involved logical problem solving without context. The test involves the implication rule: *if p then q* and four cards *p*, *not p*, *q* and *not q*. Griggs and Cox [20] provide an illustrative example of the test comprised of four cards showing respectively, A, 4, D and 7, and the rule: *if vowel then even number*. Each card has a letter on one side and a number on the other. Participants are asked to select and turn over only the cards necessary to test whether or not the rule is true. In its basic, context free form, the Wason's task has been found to be very difficult and is typically solved successfully by fewer than 10% of participants [25, 47, 48]. Some follow on studies used context variants of Wason's task finding that context was helpful and led to vastly improved performance on the task [49].

Koedinger and Nathan found that beginning algebra students were more successful at solving simple algebra story problems than solving the equivalent problem posed as a no-context, symbolic problem [29]. Their problems involved more than one mathematical operator and were presented as both start-unknown and result-unknown problems. They claim their results are not just due to situated world knowledge aiding problem solving, but are also related to a fragile understanding of symbolic representations in mathematics for novice learners. The authors go on to emphasise the importance of external representations of problems and their influence on problem solving. They note that when one representation is easier for learners to understand, or when one triggers the retrieval and use of a more reliable solution strategy, it facilitates problem solving. This conclusion echoes that of Griggs and Cox [20] who argued that context works if the student has relevant world experiences of the context and the task is presented in a manner that triggers the student's recall of that experience.

2.3 Context is Harmful

Early research on school children and their story problem solving abilities reported that arithmetic and algebraic context problems are difficult to solve [4, 37] and that more errors are made when solving story problems [19]. Cummins et al. found that math students perform better on numeric problems than on matching problems presented as stories, and go on to claim that story problems continue to be more difficult even as students progress [6]. The authors noted that factors other than mathematical skills and knowledge contribute to story problem solving success. They theorized that the main barrier to success is mapping from linguistic input into the mathematics domain knowledge and that "text comprehension factors figure heavily in word problem difficulty" [6, p. 435]. This view aligns with others in the literature who identify the comprehension phase and its associated cognitive processes as being the primary reason for errors and problem solving difficulties for story problems [7, 29, 31].

2.4 Context has No Effect

Some research has found little difference between problem solving success in context and no-context problems. When context variants of Wason's selection task were explored, some researchers reported that adding context had no effect on problem solving [33]. The 2016 ITiCSE working group found no significant difference in performance between novice programmers who attempted a context problem and those who attempted its no-context variant [2]. A subsequent controlled experiment [5] also reported a lack of measurable context effect. In this case, the context domains were chosen to be familiar to the students, having been included in earlier course work, mitigating concerns about the ability of individuals to retrieve relevant experiences of context. The authors concluded that any advantage in context was outweighed by other factors such as the complexity of terminology and the length of the problem descriptions. Similar findings are confirmed by Lovellette et al. [32] who examined novice programmer performance on Soloway's rainfall [45] problem and its decontextualized "Just the Numbers" variant.

3 RESEARCH METHODS

3.1 Participants and Timing

The study was conducted in an introductory programming course at The University of Auckland. There were 917 students in the course and the problems used in this study were given as part of the final lab (in Week 12) of the course. Each student was given two problems to solve, which were treated as "warm up" exercises for the lab given their relative simplicity compared to assessment tasks typical at the end of a CS1 course.

The students used an automated assessment tool, CodeWrite [12], to complete and submit their answers to the two programming problems. Each problem was presented as a problem description, a method signature (including the return type and parameter list) and one example output of the method along with the specific inputs to generate the output. The problems were solved in the order they were presented to the students. The tool enforced this order, which is important for this research as we examine the relationship between the first problem and the second problem encountered.

Table 2: Group Problem Allocation

	Group A	Group B	Group C
# of students	308	307	302
1st problem	Wage 1	Donuts	No-context 1
2nd problem	Wage 2	Wage 2	No-context 2

3.2 Method

Students were randomly assigned into three groups, which we call groups A, B, and C. Each group was assigned two problems that relate to our study. For all groups, the first problem encountered was a result-unknown problem and the second problem was a start-unknown problem. In the first problem, students had to solve for x in the equation: $a * b + c = x$ whereas in the second problem, students had to solve for x in the following equation: $x * a + b = c$. The exact problem descriptions are given in Figure 2.

The allocation of problems by group is shown in Table 2. Group A were given two context problems, where the context remained the same in both problems. Group B also had contextualized versions of the problems, but the context was different for each problem. For Group C, the problems had no context. In all cases, students were given a description of the problem and the signature of the method they were required to write.

For the analysis, unless otherwise specified, we examined the first submission that a student made for a particular problem. Students received immediate feedback on correctness following a submission, and could make as many submissions as were necessary for them to solve the problem. Students were generally not under any significant time pressure as these problems were relatively simple and acted as "warm up" tasks at the beginning of the lab session.

To answer Research Question 1, *"How does contextualization affect correctness?"*, the correctness of each compiling submission was calculated based on unit test results. A score was given between zero and ten depending on the number of passing test cases: the score was linear so that a submission where all the tests fail would receive a score of zero and a submission where all the tests pass would receive a score of ten. Additionally, we calculated the number of passing, non-compiling, and failing submissions for each problem.

Non-compiling submissions were not included in the correctness analysis based on scores because they could not be tested against the automatic test cases used to mark the large number of student submissions. Moreover, including all non-compiling submissions as failures in the analysis would also likely mean including programs with minor syntactic problems that are correct semantically which would obfuscate the results and incorrectly bias the data with respect to failed submissions.

To answer Research Question 2, *"How does contextualization affect time on task?"*, we analyzed the number of submissions students made for each problem. The *number of submissions* was used as a proxy for time on task. Additionally, as another measure for time

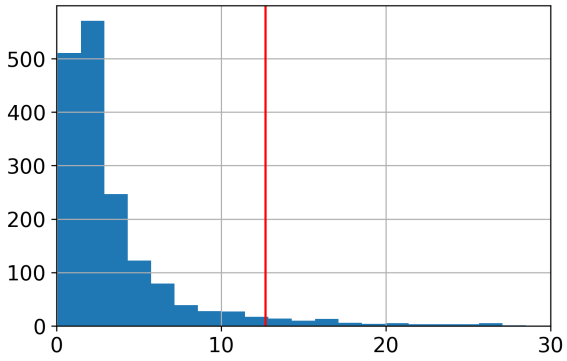

Figure 1: Histogram of the elapsed times between first view and first passing submission. The x-axis is elapsed time in minutes and the y-axis is the number of students. The red line shows the cutoff for the slowest 10% of students. The tail of the distribution is long and students who spent over 30 minutes on a problem are not shown.

spent on task, we calculated the actual *elapsed time* from the moment a student first viewed a problem to the time they submitted a successful solution. For both measures, outliers were removed to improve reliability. The removals were only done for this particular time on task analysis and do not affect the other analyses reported. For the number of submissions, students who made more than 13 submissions for a problem (only 1% of the class) were removed from analysis as this likely indicates a serious level of frustration or difficulty unrelated to the problem wording. For the elapsed time measure students who took the longest, those in the top 10%, were removed from analysis: this means that only students for whom the time between the first view of a problem and first passing submission was less than 761.5 seconds (12.7 minutes) were analyzed. Removing the slowest 10% of students removes those who, for example, took long breaks between reading a problem description and submitting a solution to that problem. The histogram of the elapsed times for a problem, for all of the problems (Figure 1) shows that removing these outliers has little effect on the analysis results. Moreover, this approach to filtering is consistent with much prior work on programming process data in situations where students are unobserved [11, 26, 30, 35].

To answer Research Question 3, *"Does contextualization have an effect on the type and number of errors students make?"*, we examined the total number of failing submissions (that is, compiling but not passing all test cases) that students made for each problem, as well as identifying the most commonly occurring combinations of failed tests to explore the common types of errors being made.

When analyzing statistical significance, we used a Bonferroni correction to account for multiple comparisons [1]. We did a total of 18 statistical significance tests and thus had $n = 18$ out of an abundance of caution, even though the score, number of attempts and time on task likely correlate with each other [30], meaning $n = 18$ could be too conservative [36]. To consider a difference significant, we used a threshold of $p < 0.05$ after the correction. To evaluate statistical significance, we used the Mann-Whitney U test because the data is not normally distributed. For effect sizes, we used Cohen's d.

Table 3: Flesch-Kincaid Grade Levels

Problem	Grade Level
Wage 1	11.1
Donuts	10.7
No context 1	12.3
Wage 2	11.8
No context 2	12.3

3.3 Problem Descriptions

The problems used in this study (see Figure 2) were adapted from Koedinger and Nathan's symbolic and story problem versions of the wages and donuts problems [29]. The first problem (Wage 1, Donuts, or No context 1) presented to each of the groups is a result-unknown problem and the second problem (Wage 2, or No context 2) is a start-unknown problem. The problems were adjusted to suit a computer programming exercise in which a method is written to solve each problem. In our CS variants, the students were given a description of the problem and a method signature. The method signature included the return type and the parameters (name and data types), and students were required to use this signature to solve the problem (see Figure 2). Providing the signature meant that the problem of designing the method was eliminated, and students could focus on the actual arithmetic, algebraic and algorithmic requirements.

We calculated the Flesch-Kincaid Grade Levels [27] for each of the problem descriptions to establish whether the readability of the assignments were equal. These are shown in Table 3. The grade levels of the problems indicate that all of the questions are at a high school readability-level, and thus should not pose problems for the university-level students in our study. More importantly, there are no notable differences between the problems with regards to readability, so any difference in the performance of students is likely unrelated to the complexity of the language used in the problem descriptions.

4 RESULTS

4.1 Correctness of First Submissions

Figure 3 shows the distribution of the outcome of the first submission by problem. Most students solved the problem successfully in their first submission regardless of which problem they were assigned. In all three groups, students had considerably fewer compilation errors in their first submissions for the second (start-unknown) problem than for the first (result-unknown) problem.

Additionally, the proportion of solutions that passed was higher for the second problem than the first problem for all groups, although the increase was considerably less for Group C. The proportion of submissions where at least one of the tests did not pass was quite similar across different problems and groups with two exceptions. Firstly, for Group B, the first problem (B1) had nearly

Problem A1
Group A, 1st problem – Wage 1
context problem — result-unknown

John is a waiter and would like to calculate his total earnings during his last shift. Make a function that calculates the total money earned by John during his shift when he knows how many hours he has worked, his hourly wage, and how much he has earned in tips during his shift.

double TotalEarnedDuringShift(int hours, double wage, double tips)

Problem B1
Group B, 1st problem – Donuts
context problem — result-unknown

Alex owns a donut shop and would like to calculate how much he should charge for a box of donuts. Make a function that calculates how much Alex should charge when he knows how many donuts fit in a box, the price of a single donut, and how much profit he would like to make for each box of donuts.

double ChargeForDonuts(int donuts, double price, double profit)

Problem C1
Group C, 1st problem – No context 1
no-context problem — result-unknown

Make a function that calculates the value of x for the following equation when a, b, and c are given as parameters: $a * b + c = x$.

double SolveEquation(double a, double b, double c)

Problem A2 & B2
Groups A & B, 2nd problem – Wage 2
context problem — start-unknown

Sarah wants to verify that her employer has paid her the correct hourly wage for her last shift. Make a function that calculates the hourly wage (not including tips) for Sarah when she knows how many hours she has worked, how much money she has gotten as tips, and how much she earned in total.

double HourlyWage(int hours, double tips, double total)

Problem C2
Group C, 2nd problem – No context 2
no-context problem — start-unknown

Make a function that calculates the value of x for the following equation when a, b, and c are given as parameters: $x * a + b = c$.

double SolveEquation(double a, double b, double c)

Figure 2: Problem Descriptions

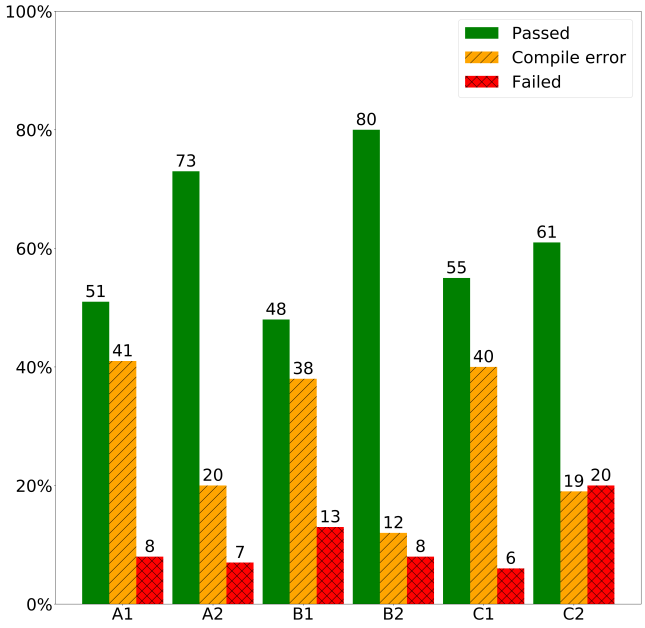

Figure 3: Distribution of Status of First Submissions by Problem and Group.

twice as many failing submissions (13%) when compared to Group A (8%) and C's (6%) first problem (A1 and C1 respectively). Secondly, Group C's second problem (C2) had almost three times as many failing submissions (20%) when compared with Groups A and B's second problem (A2 (7%) and B2 (8%) respectively). In short, the Donuts problem stood out as being difficult for students in Group B, whereas students in Group C struggled more with their start-unknown no-context problem.

The average scores for each problem (out of 10, based on the proportion of test cases passing) are provided in Table 4. Generally, students achieved high scores for their first submissions: for almost all of the problems, the average score is around 9.5 out of 10. Similar to the results shown in Figure 3, there are two exceptions: the first Group B problem and the second problem of Group C, both of which were slightly harder for students. For Group B's first problem (B1), the average score is about 0.6 lower when compared to the average score for the other two groups on their first problem (A1 and C1 respectively), and the difference is statistically significant between B1 and C1 ($U = 14693.5, p = 0.0004, d = 0.42$). For Group C's second problem (C2) the average score is about 1 mark lower than other groups on their second problems and this difference is statistically significant between both A2 vs C2 ($U = 24894.5, p < 0.0001, d = 0.45$) and B2 vs C2 ($U = 27149.0, p < 0.0001, d = 0.43$).

4.2 Time on Task

The average number of attempts per problem are shown in Table 5. There are no notable differences between the groups on either problem. The first problems (A1, B1 and C1) required, on average, slightly more attempts when compared to the second problems (A2, B2 and C2). The largest difference observed was between problems C2 (requiring 2.91 attempts) and A2 and B2 (requiring 2.49 and 2.38

Table 4: Average scores by problem. The average score for problem C2 was statistically significantly lower than the average scores of the other two similar problems (A2 and B2).

Problem ID	Problem	Score avg. (sd)
A1	Wage 1	9.49 (1.47)
A2	Wage 2	9.53 (1.52)
B1	Donuts	8.89 (2.42)
B2	Wage 2	9.51 (1.67)
C1	No context 1	9.68 (1.08)
C2	No context 2	8.57 (2.58)

Table 5: Average number of attempts. Problem C2 required significantly more attempts to solve than A2 and B2.

Problem ID	Problem	Attempts avg. (sd)
A1	Wage 1	3.40 (2.46)
A2	Wage 2	2.49 (1.15)
B1	Donuts	3.43 (2.29)
B2	Wage 2	2.38 (1.08)
C1	No context 1	3.23 (2.34)
C2	No context 2	2.91 (1.77)

attempts respectively). These differences are statistically significant: C2 vs A2 ($U = 39150.5, p = 0.005, d = 0.28$) and C2 vs B2 ($U = 36049.0, p < 0.0001, d = 0.37$).

The average time (in seconds) between the first viewing of a problem and the first passing submission can be found in Table 6. The second problem for all groups was on average faster to complete than the first problem, requiring approximately 2.3 minutes compared with approximately 3.7 minutes.

For the first problem, Group C had the fastest average completion speed of 171 seconds. In contrast, Groups A and B – who had to make sense of the problem statement context – took 206 seconds and 280 seconds respectively. In fact, students in Group B answered their first problem (the Donuts problem, B1) significantly slower than the other two groups: B1 vs A1 ($U = 20307.0, p < 0.0001, d = 0.48$) and B1 vs C1 ($U = 16011.0, p < 0.0001, d = 0.71$). For the second problem, students in the no-context group (problem C2) answered significantly faster compared to the other two groups: C2 vs A2 ($U = 36721.5, p = 0.012, d = 0.08$) and C2 vs B2 ($U = 31098.0, p < 0.0001, d = 0.24$).

4.3 Common Errors

The total number of errors, the count of the most common error, and the number of other errors per problem are shown in Table 7. When looking at the most common errors students had in submissions, where one or more of the test cases failed, we found two errors that were far more frequent than other errors.

Firstly, for the problems encountered first, for all groups the overwhelmingly most frequent error involved using integer values

Table 6: Average time on task in seconds. The time on task for the first problem encountered in each group's assignment (result-unknown) was significantly longer than for the corresponding second problem (start-unknown).

Problem ID	Problem	Time avg. (sd)
A1	Wage 1	206 (143)
A2	Wage 2	132 (98)
B1	Donuts	280 (166)
B2	Wage 2	152 (116)
C1	No context 1	171 (141)
C2	No context 2	124 (115)

Table 7: Frequency of Errors

Problem ID	# Total Errors	# Most Common	# Other
A1	25	20	5
A2	23	6	17
B1	40	18	22
B2	25	6	19
C1	17	16	1
C2	59	33	26

rather than double values in the method body (e.g. see Figure 4). This error was made even though students were given the method signature clearly detailing the requirement for a double return type. The use of integer temporary variables means that the double values computed are truncated upon assignment (although a compiler warning is issued for the type mismatch, the code is still executable). This error occurred in 80%, 45%, and 94% of the submissions with errors present in groups A, B, and C respectively.

```
double totalEarnedDuringShift(int hours,
    double wage, double tips) {
        int earnings, total;
        earnings = hours * wage;
        total = earnings + tips;
        return total;
}
```

Figure 4: Using integers instead of doubles was a common error in the first problem for all groups. An example solution using integers instead of doubles in the Wage 1 problem.

Group B exhibited more and a wider variety of errors in their solutions to their first problem, the Donuts problem, than Groups A and C encountered for their Wage 1 and No context 1 problems. The most common error observed in the Donuts problem was, as for the Wages 1 problem, the use integers rather than doubles. Analyzing

the errors in more detail, we found that many of the errors students made were related to misunderstanding the problem statement. Figures 5 and 6 show examples of solutions that exhibit two of the more common misunderstandings identified.

```
double ChargeForDonuts(int donuts,
    double price, double profit) {
        return donuts * (price + profit);
}
```

Figure 5: A misunderstanding where profit is calculated for each donut instead of the whole box of donuts.

```
double ChargeForDonuts(int donuts,
    double price, double profit) {
        double cost;
        double revenue;
        cost = donuts * price;
        revenue = profit + cost;
        return (revenue/donuts);
}
```

Figure 6: Some students mistakenly calculated the price per donut instead of the price for the whole box of donuts.

For Group C's second problem the most common error that occurred (in 33 of the 59 first submissions with errors) was an algebraic mistake. Students were required to write a program that solves for x in the equation $x * a + b = c$. The common mistake, shown in Figure 7 was caused by incorrectly rearranging the term $a + b$ from the left hand side of the equation to the right hand side by division, ending up with the incorrect equation $x = c/(a + b)$. Solving this problem correctly requires subtraction of b and division by a to rearrange the equation so that $x = (c - b)/a$.

For the second problem, Wage 2, completed by both Groups A and B, the errors were varied in both groups and there was no one common error which dominated in the solutions which failed one or more test cases. For these problems some students accidentally included the waiter's tips in the calculation, others had missing or incorrectly placed parentheses, or returned an integer value.

5 DISCUSSION

Contrary to previous results in introductory programming [2], we found that the context provided in the problem descriptions did have an effect on students' problem-solving. A possible explanation for this difference is that the solutions to our problems were more "mathematical", whereas Bouvier et al. used a variant of the Rainfall problem [2]. The solution to the Rainfall problem is more algorithmic in nature and therefore errors relating to weak knowledge of mathematics may have been less frequent, and harder to measure than algorithmic mistakes.

```
double SolveEquation(double a, double b,
    double c) {
        return c / (a + b);
    }
```

Figure 7: For the second no-context problem of Group C, a common mistake was an incorrect rearrangement of the equation.

In the first problem, students with the Wage problem (A1) and the no context problem (C1) solved their problems significantly faster than the students with the Donuts problem (B1). This difference is not caused by the Donuts problem being more linguistically complex – in fact the Donuts problem had the lowest Flesch-Kincaid Grade Level of all the task descriptions. Additionally, the Donuts problem had the lowest average score of the first set of problems. It is possible that students found the wage context easier to relate to, perhaps due to part-time work experience. In the Donuts problem, students had to assume the role of a business owner calculating profits, which may be a less familiar scenario for many students. This conjecture is consistent with the literature which suggests that an individual's familiarity with the problem context has an effect on problem-solving [20].

Looking at the types of errors students made in the Donuts problem (see Figures 5 and 6), another possible explanation for the difficulty students had may relate to the comprehension phase of problem solving, where relevant details must be extracted from the story. The errors indicate that students might have solved the wrong problem. For example, calculating how much the owner should charge for an individual donut instead of for the whole box. When comparing the narrative for the Donuts problem with that of the Wage 1 problem, the words used in the Donuts problem ("price" and "profit") could be interpreted as referring to individual donuts whereas "hourly wage" and "total earnings" in the Wage 1 problem do not suffer from the same ambiguity. This yields additional support for earlier studies that have found that problems in the comprehension phase cause the most errors when solving story problems [7, 29, 31].

Surprisingly, there were no significant differences between group A and group B with regards to performance and time on task on their second problem, which was the same for both groups (Wage 2). Our initial hypothesis was that since the context changed for group B, they would solve the second problem more slowly and with less success – this was not the case.

Another interesting result is that while the second no-context problem (C2) only took students 47 more seconds on average to complete compared to the first no-context problem (C1), students had lower scores and made more errors in their first submissions for that second problem (C2) compared with the problem versions that provided context (A2 and B2).

Delving into the reasons for this, we found that the most common error in this assignment was related to weak algebraic knowledge. The students were less able to rearrange the start-unknown problem (C2) than the previously attempted result-unknown problem

(C1). This supports earlier results in mathematics education where lack of context has been shown to make assignments harder for students [29] and that start-unknown no-context algebraic problems are more difficult to solve than result-unknown problems [7, 29].

The fact that students performed better and made less errors on A2 and B2 than on C2, may suggest that students are more likely to make logical errors when solving a problem with no-context and that context may help them, in certain circumstances, to avoid algebraic mistakes. For example, the second context problem (Wage 2) explicitly stated that when calculating the hourly wage, tips *should not be included*. This may have helped the students to construct a correct equation since they know that tips must be subtracted from the total earnings. Additionally, the algebraic version of the problem might seem very simple and easy to students at this point in their studies, which might cause them to make basic mistakes they would not make with harder assignments.

An alternative explanation is that some students may have not identified the need to cope with situations that would cause a divide by zero error. This situation only occurs in C2 where to solve for x, the students would need to divide by a – and that will not work if a is zero. In the context versions of this problem, a (hourly wage) cannot be zero, so the problem may have been simpler. However, the test suite did not include a test case where a was zero so it is more likely the task took longer and was more error prone for students who may have considered the divide by zero issue for C2.

Interestingly, in our study it took students in all groups less time to complete their second start-unknown problem than their first result-unknown problem. We might expect that a shorter time to solve the second problem suggests that students found the start-unknown problems easier. However, the scores for Group A and B's problems were relatively similar suggesting that in the case of problems with context there is no difference in difficulty between a result-unknown and a start-unknown problem. These findings contradict those of previous work in mathematics, where it was found that start-unknown problems are harder than result-unknown problems [29]. Additionally, students encountered roughly twice as many compilation errors in the first result-unknown problem when compared to the second problem, and students were more likely to get the second problem correct on their first try. One possible explanation for this is that they were less likely to make the same syntactical mistakes on the second problem.

5.1 Threats to Validity

This study was conducted at the end of the course, with the problems presented to students as "warm up" tasks for their final laboratory. These problems were relatively easy compared to the other problems students were expected to complete near the end of the course. Given their relatively simple nature, students should not have struggled greatly with the syntactic aspects of coding their solutions, and thus we believe that any differences between the groups (A, B and C) are more likely due to effects of the problem descriptions. However, because the problems were in the context of the course "easy", our results may not generalize to scenarios where students are solving more difficult problems. Moreover, the role of context in being helpful or unhelpful to students might be different for problems with different levels of complexity.

The scoring of the problems was based on automated assessment via test cases, i.e. whether specific test cases passed or not. While the test cases were the same for the different groups (since they were essentially solving the same equation), it is possible that the test cases did not capture all possible mistakes students might have made in their solutions.

Our analysis of the errors that students encountered was based on the first submission they made to each problem. Some of the errors we observed may not be due to the provided context of the problem. For example, students were not penalised for making multiple submissions and so some students may have submitted incorrect code deliberately to simply obtain feedback from the tool. However, there is no obvious reason why such behaviour would be more common in one group than another.

6 CONCLUSIONS

In this work, we studied how the context provided in problem descriptions affected the performance of introductory programming students. We conducted a randomized controlled A/B study where students were split into three groups. Each group had to program two functions that essentially solve the same two linear equations. For two of the groups, the problems were contextualized and for one there was no context. For the two groups with context, one group had the same context (calculating wages) for both problems, and for one group the context changed (from calculating the profits from donut sales to calculating wages).

Our results indicate that, contrary to previous results in introductory programming [2, 5], context might have an effect on problem solving in programming. In our case, students found a contextual version of a start-unknown problem to calculate wages easier to solve than a non-contextual version which presented only the mathematical formula. One possible explanation for the difference is that the previous studies in introductory programming had more algorithmic problems, whereas in this study the problems were more mathematical in nature. Our hypothesis is that appropriate contextual information in the problem description can help students with poor mathematical skills avoid making algebraic errors. This conjecture is supported by earlier studies in mathematics education that have found context to be helpful for students [29].

Our results suggest that context can be harmful for students with respect to time on task. Students solving a problem where they calculated profits for a donut shop spent significantly more time solving the problem when compared with both the students solving a non-contextualized version of the problem and the students solving the same problem with a different context. One possible explanation for this effect is that students might have had trouble extracting relevant information from the story even though the problems were, according to their Flesch-Kincaid grade levels, of a similar linguistic complexity.

Based on the results presented here, contextualizing the problem descriptions for relatively simple programming tasks can be both helpful and harmful with respect to solution correctness and time on task. Thus, more research is needed to study the aspects of context that affect problem solving in introductory programming assignments, and to explore these effects across more complex tasks.

REFERENCES

[1] Richard A. Armstrong. 2014. When to use the Bonferroni correction. *Ophthalmic and Physiological Optics* 34, 5 (2014), 502–508.

[2] Dennis Bouvier, Ellie Lovellette, John Matta, Bedour Alshaigy, Brett A. Becker, Michelle Craig, Jana Jackova, Robert McCartney, Kate Sanders, and Mark Zarb. 2016. Novice Programmers and the Problem Description Effect. In *Proceedings of the 2016 ITiCSE Working Group Reports* (Arequipa, Peru) *(ITiCSE '16)*. ACM, New York, NY, USA, 103–118.

[3] Katalin Bubnó and Viktor L. Takács. 2017. The mathability of word problems as initial computer programming exercises. In *2017 8th IEEE International Conf. on Cognitive Infocommunications (CogInfoCom)*. 39–44.

[4] John Clement. 1982. Algebra Word Problem Solutions: Thought Processes Underlying a Common Misconception. *Journal for Research in Mathematics Education* 13, 1 (1982), 16–30.

[5] Michelle Craig, Jacqueline Smith, and Andrew Petersen. 2017. Familiar Contexts and the Difficulty of Programming Problems. In *Proceedings of the 17th Koli Calling International Conf. on Computing Education Research* (Koli, Finland) *(Koli Calling '17)*. Association for Computing Machinery, New York, NY, USA, 123–127.

[6] Denise Dellarosa Cummins, Walter Kintsch, Kurt Reusser, and Rhonda Weimer. 1988. The role of understanding in solving word problems. *Cognitive Psychology* 20, 4 (1988), 405 – 438.

[7] Gabriella Daroczy, Magdalena Wolska, Walt Detmar Meurers, and Hans-Christoph Nuerk. 2015. Word problems: a review of linguistic and numerical factors contributing to their difficulty. *Frontiers in Psychology* 6 (2015), 348. https://www.frontiersin.org/article/10.3389/fpsyg.2015.00348

[8] Erik De Corte and Lieven Verschaffel. 1981. children's solution processes in elementary arithmetic problems: Analysis and improvement. *Journal of Educational Psychology* 73, 6 (1981), 765–779.

[9] Erik De Corte and Lieven Verschaffel. 1981. The effect of semantic structure on first graders' strategies for solving addition and subtraction word problems. *Journal for Research in Mathematics Education* 18, 5 (1981), 363–381.

[10] Paul Denny, Brett A. Becker, Michelle Craig, Greg Wilson, and Piotr Banaszkiewicz. 2019. Research This! Questions That Computing Educators Most Want Computing Education Researchers to Answer. In *Proceedings of the 2019 ACM Conf. on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. ACM, New York, NY, USA, 259–267.

[11] Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2012. All Syntax Errors Are Not Equal. In *Proceedings of the 17th ACM Annual Conf. on Innovation and Technology in Computer Science Education*. ACM, New York, NY, USA, 75–80.

[12] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. 2011. CodeWrite: supporting student-driven practice of java. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. 471–476.

[13] Paul Denny, James Prather, Brett A. Becker, Zachary Albrecht, Dastyni Loksa, and Raymond Pettit. 2019. A Closer Look at Metacognitive Scaffolding: Solving Test Cases Before Programming. In *Proceedings of the 19th Koli Calling International Conf. on Computing Education Research* (Koli, Finland) *(Koli Calling '19)*. ACM, New York, NY, USA, Article 11, 10 pages.

[14] Janine DeWitt and Cynthia Cicalese. 2006. Contextual Integration: A Framework for Presenting Social, Legal, and Ethical Content across the Computer Security and Information Assurance Curriculum. In *Proceedings of the 3rd Annual Conf. on Information Security Curriculum Development (InfoSecCD '06)*. Association for Computing Machinery, New York, NY, USA, 30–40.

[15] Sarah Esper, Stephen R. Foster, and William G. Griswold. 2013. CodeSpells: Embodying the Metaphor of Wizardry for Programming. In *Proceedings of the 18th ACM Conf. on Innovation and Technology in Computer Science Education* (Canterbury, England, UK) *(ITiCSE '13)*. Association for Computing Machinery, New York, NY, USA, 249–254.

[16] Kathi Fisler. 2014. The Recurring Rainfall Problem. In *Proceedings of the Tenth Annual Conf. on International Computing Education Research* (Glasgow, Scotland, United Kingdom) *(ICER '14)*. Association for Computing Machinery, New York, NY, USA, 35–42.

[17] Andrea Forte and Mark Guzdial. 2005. Motivation and Nonmajors in Computer Science: Identifying Discrete Audiences for Introductory Courses. *IEEE Trans. on Educ.* 48, 2 (May 2005), 248–253.

[18] Ana García, Juan E. Jiménez, and Stephany Hess. 2006. Solving arithmetic word problems: An analysis of classifications as a function of difficulty in children with and without arithmetic LD. *Journal of Learning Disabilities* 39, 3 (2006), 270–281.

[19] David C. Geary. 1994. *Children's mathematical development: Research and practical applications*. American Psychological Association.

[20] Richard A Griggs and James R Cox. 1982. The elusive thematic-materials effect in Wason's selection task. *British journal of psychology* 73, 3 (1982), 407–420.

[21] Mark Guzdial. 2010. Does Contextualized Computing Education Help? *ACM Inroads* 1, 4 (Dec. 2010), 4–6.

[22] Rogers Hall, Dennis Kibler, Etienne Wenger, and Chris Truxaw. 1989. Exploring the Episodic Structure of Algebra Story Problem Solving. *Cognition and Instruction* 6, 3 (1989), 223–283.

[23] James Hiebert. 1982. The position of the unknown set and children's solutions of verbal arithmetic problems. *J. Research in Mathematics Ed.* 13, 5 (1982), 341–349.

[24] Matthew Inglis and Adrian Simpson. 2004. Mathematicians and the Selection Task. In *Proceedings of the 28th Conf. of the International Group for the Psychology of Mathematics Education*, Vol. 3. 89 – 96. https://eric.ed.gov/?id=ED489556

[25] Philip N. Johnson-Laird and Peter C. Wason. 1970. A theoretical analysis of insight into a reasoning task. *Cognitive Psychology* 1, 2 (1970), 134–148.

[26] Ioannis Karvelas, Annie Li, and Brett A. Becker. 2020. The Effects of Compilation Mechanisms and Error Message Presentation on Novice Programmer Behavior. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 759–765.

[27] Peter J. Kincaid, Robert P. Fishburne Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation Of New Readability Formulas (Automated Readability Index, Fog Count And Flesch Reading Ease Formula) For Navy Enlisted Personnel. *Institute for Simulation and Training* 56 (1975), 48 pages. https://stars.library.ucf.edu/istlibrary/56

[28] Walter Kintsch. 1998. *Comprehension: A Paradigm for Cognition*. Cambridge University Press, New York.

[29] Kenneth R Koedinger and Mitchell J Nathan. 2004. The real story behind story problems: Effects of representations on quantitative reasoning. *The journal of the learning sciences* 13, 2 (2004), 129–164.

[30] Juho Leinonen, Leo Leppänen, Petri Ihantola, and Arto Hellas. 2017. Comparison of time metrics in programming. In *Proceedings of the 2017 acm Conf. on international computing education research*. 200–208.

[31] Anne Bovenmyer Lewis and Richard E. Mayer. 1987. Students' miscomprehension of relational statements in arithmetic word problems. *Journal of Educational Psychology* 79, 4, 363–371.

[32] Ellie Lovellette, John Matta, Dennis Bouvier, and Roger Frye. 2017. Just the Numbers: An Investigation of Contextualization of Problems for Novice Programmers. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 393–398.

[33] K. I. Manktelow and J. St B. T. Evans. 1979. Facilitation of reasoning by realism: Effect or non-effect? *British Journal of Psychology* 70, 4 (1979), 477–488.

[34] Raina Mason and Carolyn Seton. 2020. Assessing International Students: The Role of Cognitive Load. In *Proceedings of the Twenty-Second Australasian Computing Education Conf.* (Melbourne, VIC, Australia) *(ACE'20)*. Association for Computing Machinery, New York, NY, USA, 160–166.

[35] Davin McCall and Michael Kölling. 2019. A New Look at Novice Programmer Errors. *ACM Trans. Comput. Educ.* 19, 4, Article 38 (July 2019), 30 pages.

[36] Matthew D Moran. 2003. Arguments for rejecting the sequential Bonferroni in ecological studies. *Oikos* 100, 2 (2003), 403–405.

[37] Mitchell J. Nathan, Walter Kintsch, and Emilie Young. 1992. A Theory of Algebra-Word-Problem Comprehension and Its Implications for the Design of Learning Environments. *Cognition and Instruction* 9, 4 (1992), 329–389.

[38] Ronald H. Nowaczyk. 1984. The relationship of problem-solving ability and course performance among novice programmers. *International Journal of Man-Machine Studies* 21, 2 (1984), 149–160.

[39] George Pòlya. 1971. *How to Solve It* (2 ed.). Princeton University Press.

[40] James Prather, Raymond Pettit, Brett A. Becker, Paul Denny, Dastyni Loksa, Alani Peters, Zachary Albrecht, and Krista Masci. 2019. First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts. In *Proc. of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. ACM, New York, NY, USA, 531–537.

[41] Kurt Reusser. 1988. Problem solving beyond the logic of things: contextual effects on understanding and solving word problems. *Instr Sci* 17 (1988), 309–338.

[42] Anthony Robins. 2019. Novice programmers and introductory programming. In *The Cambridge Handbook of Computing Education Research*, Sally Fincher and Anthony Robins (Eds.). Cambridge University Press, Cambridge, UK, Chapter 12, 327–376.

[43] Otto Seppälä, Petri Ihantola, Essi Isohanni, Juha Sorva, and Arto Vihavainen. 2015. Do We Know How Difficult the Rainfall Problem Is?. In *Proceedings of the 15th Koli Calling Conf. on Computing Education Research* (Koli, Finland) *(Koli Calling '15)*. Association for Computing Machinery, New York, NY, USA, 87–96.

[44] Ben Shneiderman and Richard Mayer. 1979. Syntactic/semantic interactions in programmer behavior: A model and experimental results. *International Journal of Computer and Information Sciences* 8 (1979), 219–238.

[45] Elliot Soloway. 1986. Learning to program= learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (1986), 850–858.

[46] John Sweller. 1988. Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science* 12, 2 (1988), 257–285.

[47] P. C. Van Duyne. 1974. Realism and Linguistic Complexity in Reasoning. *British Journal of Psychology* 65, 1 (1974), 59–67.

[48] Peter C. Wason and Philip N. Johnson-Laird. 1972. *Psychology of Reasoning: Structure and Content*. Harvard University Press, Cambridge, MA.

[49] Peter C. Wason and Diana Shapiro. 1971. Natural and contrived experience in a reasoning problem. *Quarterly J. of Experimental Psychology* 23, 1 (1971), 63–71.

[50] Svetlana Yarosh and Mark Guzdial. 2008. Narrating Data Structures: The Role of Context in CS2. *J. Educ. Resour. Comput.* 7, 4, Article 6 (Jan. 2008), 20 pages.