Lukkarinen, Aleksi; Hellas, Arto; Haaranen, Lassi

# A Kingdom for a Button: Students' Thoughts about Buttons

*Published in:*
Proceedings of 21st Koli Calling International Conference on Computing Education Research, Koli Calling 2021

*DOI:*
10.1145/3488042.3490170

Published: 18/11/2021

*Document Version*
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

*Please cite the original version:*
Lukkarinen, A., Hellas, A., & Haaranen, L. (2021). A Kingdom for a Button: Students' Thoughts about Buttons. In O. Seppälä, & A. Petersen (Eds.), *Proceedings of 21st Koli Calling International Conference on Computing Education Research, Koli Calling 2021* (pp. 1-6). Article 21 ACM. https://doi.org/10.1145/3488042.3490170

**ACM Reference format:**
Aleksi Lukkarinen, Arto Hellas, and Lassi Haaranen. 2021. A Kingdom for a Button: Students' Thoughts about Buttons. In *21st Koli Calling International Conference on Computing Education Research (Koli Calling '21), November 18–21, 2021, Joensuu, Finland.* ACM, New York, NY, USA, 6 pages.
https://doi.org/10.1145/3488042.3490170

# A Kingdom for a Button: Students' Thoughts about Buttons

Aleksi Lukkarinen
aleksi.lukkarinen@aalto.fi
Aalto University
Espoo, Finland

Arto Hellas
arto.hellas@aalto.fi
Aalto University
Espoo, Finland

Lassi Haaranen
lassi.haaranen@aalto.fi
Aalto University
Espoo, Finland

## ABSTRACT

In learning programming and learning to construct applications with graphical user interfaces, there exists a large body of concepts from everyday life that are used to guide students. In this study, we explore whether such concepts from everyday life that we might believe require no explanation are actually understood in different ways by students in an introductory browser-based applications course. Our analysis focuses on an elementary interactive element: *a button*. Analyzing survey data from 185 students, we observe that even the simple concept of a button may be understood in a myriad of ways and that the context in which the concept is represented significantly influences beliefs of how the concept behaves. Our results indicate that students can rarely disentangle a concept from the context, and that some even believe that the text shown on a button in a graphical user interface is used to define the functionality of the button.

## CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; • **Software and its engineering** → *Publish-subscribe / event-based architectures.*

## KEYWORDS

button, graphical user interface, contextualization, programming education, computer science education

## 1 INTRODUCTION

In the contemporary world, graphical user interfaces (GUI) are commonplace in many computerized technologies, including web sites, desktop applications, mobile devices, as well as high-tech household appliances and industrial machinery. Actionable parts, such as buttons, are an essential part of any interactive GUI. Similar tangible elements have existed in physical user interfaces of both electrical and mechanical devices long before any computer devices. Naturally, the physical objects that preceded the virtual ones have affected the way buttons are displayed on the screen.[1] Understanding—or misunderstanding—how these graphical user interfaces and their components have evolved over time influences our interactions with them [6, 10]. We are not aware of any literature of how CS students understand GUIs.

As teachers, we often base our teaching on preconceived ideas of students' existing knowledge, trying to tap into that knowledge and build on top of it. Our ideas of previous knowledge often stem from learning objectives of prerequisite courses, common knowledge, and intuition, i.e., *beliefs of how the world works* [16]. Although not necessarily explicitly, we may align ourselves with *Platonic realism* [see, e.g., 25], believing that there exist essences of all things that objects in the real world imitate. In doing so, it is possible that things that we believe as common knowledge may not actually be as common as we think.

We seek to unveil how students understand specific computing-related concepts that are often considered to be common knowledge. Specifically, we focus on the concept of *a button*: When we talk about a button, discuss interaction with a button, and use the concept within our learning materials, should we believe that students know what a button is and how it might behave? The specific research question of this work is: *how do students understand the concept of a button?* The work in this article has been conducted within an introductory course that teaches how browser-based applications work. The course is held continuously—there are no deadlines or an "end of course" date. The students can enroll and start working on the material whenever they choose, and similarly, they can request to receive the credits once they have completed enough exercises.

In general, our work posits that challenging our beliefs on even such simple (but complex!) concepts might produce insights into the pitfalls that students—and more broadly, people—face when working with and constructing interactive applications. Such information could be used, for instance, to inform the design of contextualized tools for learning programming [15], to provide insight into why some problem descriptions might work and others might not [1, 3, 14], and to help design interventions that could improve students' performance in tasks that require such information as well as avoid task-specific over-generalization [7].

This article is organized as follows: In § 2, we discuss related works within and outside of the computing education literature, outlining streams of pedagogical practice that might benefit from similar analyses. This is followed by a methodological description in § 3, which outlines the course context, the data, and the analyses. Finally, we outline the study's results in § 4 and discuss them in § 5.

[1] Often referred to as *skeuomorphism.*


*Koli Calling '21, November 18–21, 2021, Joensuu, Finland*

ACM ISBN 978-1-4503-8488-9/21/11...$15.00
https://doi.org/10.1145/3488042.3490170

## 2 BACKGROUND

When it comes to technical objects, Hamman argues that commonly a person *"understands technical objects as tools whose use-function is fixed and nonnegotiable"* [8]. This is aligned with Heidegger's view that the use of equipment transforms the tool as a thing itself to its use-function [5, 9]. Thus, the object itself "disappears" and it is only viewed through the function that it provides. However, this interpretation of tools does not extend to mathematical and scientific concepts. Rather, different persons might have varying interpretations about any particular concept [21].

Regarding individuals, Säljö further points out that learning new concepts is based on collective cultural experience instead of happening in a vacuum [24]. In learning, the interaction between the teacher—or the material created by them—and the learner becomes crucial, especially regarding the interpretations. Thus the interaction between the two should also be investigated [11].

Teaching a subject with a context is a well researched area, especially in elementary education in mathematics. For example, providing a short narrative about a math problem might lead students to understand and interpret the problem differently, and importantly, they might not be able to transfer it to the domain of mathematics [26]. Reference world, or reference domain, has been used to explain this: *"The more natural the representations, that is, the more the representation domain coheres because of its implicit properties as a reference world, the more difficult it may be to see past those properties to perceive its structure as an abstract symbol system"* [22]. This in turn, might lead to difficulties in abstracting the ideas and transferring them into the domain of mathematics [22, 26].

In teaching CS, the use of contextualization in programming is still under investigation. While contextualization may increase retention (or at least pass rates of introductory courses), it is still unclear what sorts of differences one observes in learning outcomes when comparing so called traditional introductory programming courses and contextualized introductory programming courses [1, 7]. However, there is some indication that contextualization might not actively *hinder* learning to program [1].

Differences between the way people commonly think about concepts, whether in CS or mathematics, might explain some difficulties that novice programmers encounter. For instance, indexing arrays is known to be hard for novices [e.g., 12]. "Normally" we count from one, but within CS, zero-based indexing is more common.

Miller's research [18] on metonymy, referring to a concept by the name of some closely associated concept, and programming is closely related to our work. He gives three potential sources for students' issues regarding reference-point errors: (1) deficient mental representation [e.g., 19, 23], (2) misunderstanding of what the system can do (a "superbug" [20]), and (3) reliance on implicit habits for communication [e.g., 4]. Effectively, Miller observes that students' errors might be due to their conceptual models differing from the effective models.

## 3 METHODOLOGY

The study was conducted as a survey within an introductory course on browser-based application development, offered by Aalto University. The course is a 2 ECTS[2] life-long learning course: Most participants are in working life and are taking courses to learn additional skills. The course is held continuously: Students enroll to it when they are ready to begin, work on the material on their own pace, and after finishing the exercises they can request the credits.

The course introduces the learners to the concept of Internet, to the principles of how browser-based web applications work, and to how one can build simple browser-based applications. There are no prerequisites for attending the course, but many of the participants have taken two small introductory courses: one in programming and one discussing the concepts of data and information.

The course material is delivered online and has eight chapters of learning content. The third chapter is involved with event-driven programming and manipulating the web page. When students began working on the third chapter, they encountered a link to an optional survey—created using Google Forms—from which the data for this study was gathered. At that time, students have not yet created interactive browser-based applications with buttons, but many have written simple (text-based) programs that, for instance, ask for input from the user and perform operations on those inputs.

The survey had four substantive questions (Q1–Q4, Table 1). We purposefully did not seek to define them unambiguously, but rather wanted to collect a broad range of answers which would provide insights into how the students perceived the concept of a button in different contexts. Specifically, we wanted to see general conceptions of a button (Q1), a button in a familiar context (Q2), a context where the function of a button is ambiguous (there is no clear "correct" answer) yet familiar (Q3), and how the presence of code influences the answer (Q4). We expected to see a range of answers to these questions influenced by the student's earlier experience with user interfaces, programming, and, of course, buttons.

For demographics (see Table 2), we collected (1) the participant's age group and (2) whether they had experience in developing programs that handle events or have a user interface (this was a *yes/no* question). We did not provide any answering guidelines, such as a suitable length of an answer. The language of the survey was Finnish. In Q4, the program code had neither syntax highlighting nor breaks in the middle of the code statements.

We opened the survey in June 2020; participation was voluntary and unrewarded. We stated that answers would help in basic research, so research consent was given by participating. After deciding to stop accepting more data, we screened the received 191 responses for duplicates (4) and possibly underaged students (2). This left us with 185 responses from June 14, 2020 to May 21, 2021.[3]

We used thematic analysis [2] for our analysis. We built a high-level categorization of the responses, identifying recurring themes within the answers for each question. Then, two researchers analyzed the responses independently, categorizing them using the previously defined high-level categorization, including also an option to describe other aspects of each answer.

## 4 RESULTS

The participants of the survey were mostly nearing their middle-age (Table 2): The largest age group was 36–45 with 37.3 % of the students, which matches the course's positioning towards continuing education, switching careers, and life-long learning. Although

[2] *European Credit Transfer System*; 1 ECTS is approximately 25–30 hours of work.

[3] During that period, 412 students worked on exercises on the survey link page.

**Table 1: Survey questions (see § 3). In the text, Q1 is referred to as "Essence of a Button," Q2 is referred to as "Shopping Mall," Q3 is referred to as "Form for Personal Information," and Q4 is referred to as "Weather Application."**

| ID | Question (translated from Finnish) |
|---|---|
| Q1[4] | How would you describe a thing that is commonly called as "button" or "*«a synonym»*"? |
| Q2 | You are in a shopping mall and see a screen that has a button on it. The button has a text "Touch to Start." What do you think will happen when the button is pressed? Why? |
| Q3 | You are filling a personal information form on a web page. At the end of the form is a button that says "Save." What do you think will happen when the button is pressed? Why? |
| Q4 | There is a weather information application in your mobile phone. The app has a button with a text "Fetch Weather Data." Pressing the button causes execution of the program code listed below. What do you think will happen when the button is pressed? Why? |

```
var button = getButtonByText("Fetch Weather Data");
button.setText("Fetching Data");
var fetch = await HttpRequest.get(
      "https://weather-service.com/weather");
var textfield = getTextFieldById("weatherinfo");
textfield.setText(fetch);
button.setText("Fetch Weather Data");
```

we surveyed the students for differences in experience in developing programs that handle events or have a user interface, we did not find clear differences in the analysis of answers that could be explained by the previous experience. Thus, in the following analyses, we consider the population as a whole.

For each of the four questions (Q1–Q4), only a few individual students left the answer empty, wrote something irrelevant, or expressed that they did not know how to answer. The shortest, average, and longest lengths of answers, in characters, were: Q1: 5, 61.8, 372; Q2: 7, 91.5, 421; Q3: 9, 102.9, 444; and Q4: 3, 124.0, 735.

**The Essence of a Button (Q1).** For a seemingly simple user interface element, a button seems to be difficult to define exhaustively. Figure 1 presents examples of the terms students used to refer to the essence of a button. A majority of the answers contained some vague word that tells practically nothing about the button. The next common group of terms were related to interaction and user interfaces. A small group of students applied an appearance-related definition, while another small group did not make this definition altogether. Regarding terminology, a few students explained a button by referring to a synonym (effectively not saying anything), and a few gave different meanings for words that are commonly considered as synonymous. A few students also recognized that the word *button* can also refer to buttons of clothes.

Whereas a minority of students did not give any purpose for buttons, most of the answers stated that a button causes something to happen. Other purposes included giving a possibility to choose,

[4]We are not aware of English words synonymous to *a button*: The meanings of *knob*, *switch*, and *toggle*, for instance, are different. However, Finnish has several commonly-used terms for the concept of a button; thus the marking *«a synonym»* in Table 1.

**Table 2: Numbers of accepted students per age and experience groups (see §§ 3 and 4). $n_e/n_{ne}$ is the ratio of experienced to non-experienced students. All percentages are of the total number of accepted students (185).**

| Age Group | Experienced | | No Experience | | $\frac{n_e}{n_{ne}}$ | Totals | |
|---|---|---|---|---|---|---|---|
| | n | % | n | % | | n | % |
| 18–25 | 11 | 5.9 | 7 | 3.8 | 1.6 | 18 | 9.7 |
| 26–35 | 21 | 11.4 | 18 | 9.7 | 1.2 | 39 | 21.1 |
| 36–45 | 42 | 22.7 | 27 | 14.6 | 1.6 | 69 | 37.3 |
| 46–55 | 23 | 12.4 | 15 | 8.1 | 1.5 | 38 | 20.5 |
| 56–65 | 7 | 3.8 | 8 | 4.3 | 0.9 | 15 | 8.1 |
| 66– | 4 | 2.2 | 2 | 1.1 | 2.0 | 6 | 3.2 |
| Totals | 108 | 58.4 | 77 | 41.6 | 1.4 | 185 | 100.0 |

sending a command or some input, interacting, progressing forwards in a process, and confirming. Very few students mentioned the possibility of nothing happening at all, and a few individuals recognized that a button might be a toggle, staying in one position until pressed again.

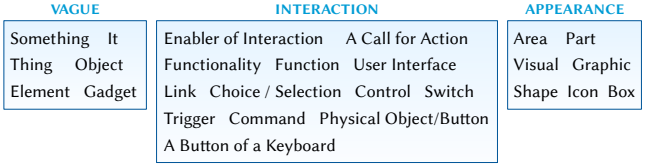


**Figure 1: Examples of terms (translated from Finnish), with which students referred to the essence of a button.**

A small number of students gave buttons some locations, such as a web page, a user interface on a computer's screen, and a physical machine, including computers' keyboards. Some answers defined the content of a button, such as text, an icon, or an image. A slightly larger number of answers defined that content's purpose, such as describing the effect, choice, or interaction. Some mentioned the mouse, the keyboard, and the touch screen as interaction channels. It was more common to mention the interaction method, such as pushing, pressing, clicking, or touching.

**Case: Shopping Mall (Q2).** The most common consequence mentioned by far was that something begins or starts. A few students stated that the button will start a chain of steps, and some suggested that the program advances to the next step. A bit more concrete answer, with approximately a third of the frequency of something beginning, was that the view will change. Still a pinch of more concreteness was in those few answers proposing that a user interface appears, without specifying its purpose. In addition to the more abstract levels of consideration, a popular approach was to suggest some specific features that one could believably encounter in screens at shopping malls. These included the main menu; the map of the building; a listing or a search of services available; a tutorial for the user; general information, such as the opening hours; a customer survey; as well as a self-service checkout.

Interestingly, only one student argued that it is impossible to know what the screen does. Based on this, we could even argue that this was a trick question, as it conveyed the expectation that the respondent should be able to tell what the consequence is. Similarly, one student commented that the scenario is absurd and that the situation always has to include information about the consequences. Two others speculated that as no guidance is available before pressing the button, the system has to give it after the press.

A probable misunderstanding was that a computer or a program starts or wakes up. Although technically possible, this most likely would not happen in the real world. Instead, the whole user interface available through the screen would likely be implemented as one constantly-running application program that is executed by a single computer. Some students expected the button to exist because a screen saver or advertisements is/are being displayed while the screen is not being directly used. If so, this utilization of idle time would likely be integrated to the same application.

**Case: A Form for Personal Information (Q3).** For a consequence of the button, most of the students answered that the information is saved, and a small portion said that the information is submitted/delivered. In the end, it seemed not to be clear, where the information would be stored and whether it would be submitted somewhere. For some, saving meant submitting the information to some organization's server, possibly for storing it into a database. Some students suggested that the information could be saved to a "cache" of the browser or as a file to a local mass storage. Moreover, some said that the data storage location cannot be known, and some individuals speculated about the possibility of having also/only a "submit" button.

A small number of students commented on other aspects of the process that the user of the form is involved with. Some suggested that the user could continue filling the form later, possibly in another place, and that the process might not be complete yet. On the other hand, some students interpreted the existence of the button as a mark for the user of the end of the data input process, and some suggested that the button would cause the process to advance to the next step; both of these might imply saving the data in some manner. In addition, it was mentioned that pressing the button denotes user's acceptance for processing (storing) the given information, and two other students thought that the user and the receiver of the data would be notified about completing the form. Furthermore, being registered to the website was mentioned as a reason for getting access to the saved data later.

**Case: Weather Application (Q4).** Although we listed the program code that the button would trigger, a large majority of the students answered on a higher abstraction level than describing program code. The simplest of these answers were similar to *"one gets the weather data"* (in Finnish: *"saat säätiedot"*). Only a small minority of the students described the functionality on the abstraction level of the given program code. In these answers, fetching references with the `getButtonByText()` and `getTextFieldById()` methods (Table 1, Q4, code lines 1 & 5), was often omitted–that is, students tended to group fetching the reference to a field and setting or retrieving the value of the specific field together. The other code lines were discussed more frequently.

There were several misunderstandings in how the program code works. For instance, regarding retrieving the weather data, the user would enter search terms or a location into a text field, or the program would (1) enter a search term and start the search or (2) direct the user to a web page either to see the weather data or to search for it. Sources of the weather data included a database, the web page/site at the given address, and an application programming interface that the server provides. Furthermore, it was proposed that the program would, for instance, retrieve the weather data but not show it, or create a text field for displaying the search results.

## 5 DISCUSSION

Although a majority of the students did not volunteer reasons for their thinking, we offer highlights of what they said and seemed to assume.

**Button as an Actor.** Perhaps the most common assumption regarding Q1 was that something happens when a button is pressed (e.g., *"if no effect can be observed, I will assume that the button is broken"*). Some students thought that the button and the resulting functionality are inseparable (e.g., *"a button is a functionality"*). Furthermore, some described the button as an actor that actively performs a process instead of triggering it. One source of these ideas might be that the user sees only the button, which makes it a touchpoint to all the functionality that it controls.

**Experience and Expression.** Our assumptions and interpretations of our perceptions are affected by various contextual factors. From a personal side, experience was one of the two reasons that students most often mentioned for their thinking. For all three scenarios (Q2–Q4), some reflected having met similar systems before. As an example, in Q2, one student compared the screen to a self-service point-of-sale system of a well-known fast food chain.

As another example of experience (at least in some cases), in Q4, several students assumed that the fetched weather data would be current by time and/or location. Yet another assumption was that the mobile phone would ask a permission for using a mobile data subscription, and that "nothing" would happen if the mobile data was turned off. While these assumptions might be relatively reasonable in general, the program code does not convey them to the reader and the actual system might work differently.

Also belonging to the personal factors, students may have problems in expressing their thinking accurately in textual form [13], and both the terminology and its meaning are likely to vary between individuals, especially when they have not yet been exposed to accurate terminology and definitions on a subject. It might also be that there is no accepted terminology or clear practice in using it. An example of this is the exact nature of saving in Q3: *To save* does not automatically mean submitting as well, and submitting information does not necessarily imply storing it. Instead, the information could be saved locally, or the submission could be processed at the server in some way and then discarded.

**Environment.** In Q2, some students mentioned inferring results based on the scenario's environment: the shopping mall. How different would the answers have been if the screen had been at the front of an auditorium, in a room of a hospital ward, or in a window? After all, the screen was only located at a shopping mall, and the "screen" and the "button" might not have had anything to do with computers in the first place. Similarly, most of the students assumed that Q1 is about buttons in graphical user interfaces; we

did not observe many mentions of buttons in physical machines. A web development course as the environment of the survey itself has likely had its effect on this.

**Labels in user interfaces.** In Q2–Q4, the other most-frequent reason for expecting a specific outcome was the caption of the button. This naturally includes the assumption that the caption sufficiently describes the functionality that the button truly triggers,[5] which might not be true even for benevolent programs. The situation becomes more interesting in Q4, where we gave the program code and stated that it will be executed when the button is pressed. *Despite this*, a few students clearly stated the caption of the button as the reason for expecting a specific functionality. Even if the students would not understand the program code, they should still know that the code defines what will happen and not the caption. One could hypothesize that these students neglected to mention falling back to the caption because of not understanding the program code. However, if this hypothesis would not hold for all of these students, then the reason behind this phenomenon would remain unclear.

## 5.1 Limitations

The largest threat to validity is the interpretation of students' answers. Often students respond in everyday language that is ambiguous and does not offer an absolute certainty regarding the respondents' understanding. Second, there is intentional ambiguity in the wording of the questions. Third, the assumptions and knowledge of each researcher affect the interpretation of the answers. Fourth, a small limitation was that the survey was anonymous and we were unable to (1) include more types of personal information into the analysis and (2) detect duplicate submissions reliably. Finally, to avoid the few answers that were not to the point, it might have been beneficial to instruct the students in how to answer.

We also acknowledge that asking only for experience in developing programs that handle events or have a user interface is a limitation. In particular, this study did not separate students based on their programming experience.

## 5.2 Conclusions

In this study, we sought to answer the question of *how do students understand the concept of a button*. It seems that some students might be lacking fundamental skills of understanding interactive applications and reasoning about them. Our results demonstrated that while our students had some ideas of what a button is and what might happen in our three scenarios, these ideas were often vague and their descriptions partly ambiguous or incorrect.

Given the contemporary ubiquitous nature of graphical user interfaces, we might be inclined to assume that the existence of this knowledge can be taken for granted. As this fundamental knowledge is important for every professional and useful for everyone, it could be advantageous to increase the amount of basic education on basic components of graphical user interfaces, including buttons, as well as rethinking what the students know and might not know of the concepts that are common knowledge to us.

One interesting finding was that some students thought that the button's caption defined its functionality. To alleviate this, perhaps students could develop an application, which is "complete" and has a "Start" button that does nothing? This could be used to illustrate, for instance, (1) a button itself, (2) that a button is not the functionality, and (3) should nothing happen, most likely the button itself is not broken. Moreover, having the button to do something that contradicts its caption could serve to clarify that (1) the caption does not define the functionality and (2) *still*, the button control itself is not broken—only misconfigured.

[5] McDermott has discussed [17] a similar issue regarding identifiers in program code.

## REFERENCES


[1] Dennis Bouvier, Ellie Lovellette, John Matta, et al. 2016. Novice Programmers and the Problem Description Effect. In *Proc 21st ITiCSE WGR* (Arequipa, PE). ACM, New York, NY, USA, 103–118.

[2] Virginia Braun, Victoria Clarke, Nikki Hayfield, and Gareth Terry. 2019. Thematic Analysis. In *Handbook of Research Methods in Health Social Sciences*, Pranee Liamputtong (Ed.). Springer Singapore, Singapore, 843–860.

[3] Michelle Craig, Jacqueline Smith, and Andrew Petersen. 2017. Familiar Contexts and the Difficulty of Programming Problems. In *Proc 17th Koli Calling* (Koli, FI). ACM, New York, NY, USA, 123–127.

[4] Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expr. *Cogn Sci* 19, 2 (1995), 233–263.

[5] Hubert Lederer Dreyfus. 1990. *Being-in-the-world: A commentary on Heidegger's Being and Time, Division I.* MIT Press, Cambridge, MA, USA. 384 pages.

[6] Jennifer Ferreira, Pippin Barr, and James Noble. 2005. The Semiotics of User Interface Redesign. In *Proc 6th AUIC, Vol. 40* (Newcastle, AU). ACS, AU, 47–53.

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

[8] Michael Hamman. 2000. From Technical to Technological: Interpreting Technology Through Composition. In *Proc Colloquium on Musical Informatics*. University Park, PA, USA, 4 pages.

[9] Martin Heidegger. 1988. *The Basic Problems of Phenomenology (A Midland Book MB 478).* Indiana University Press, Bloomington, IN, USA.

[10] Muhammad Nazrul Islam. 2013. A Systematic Literature Review of Semiotics Perception in User Interfaces. *J Syst Inf Technol* 15, 1 (March 2013), 45–77.

[11] Sanna Järvelä. 1995. The Cognitive Apprenticeship Model in a Technologically Rich Learning Environment: Interpreting the Learning Interaction. *Learn Instr* 5, 3 (1995), 237–259.

[12] Einari Kurvinen, Niko Hellgren, Erkki Kaila, Mikko-Jussi Laakso, and Tapio Salakoski. 2016. Programming Misconceptions in an Introd. Level Programming Course Exam. In *Proc 21st ITiCSE*. ACM, New York, NY, USA, 308–313.

[13] Teemu Lehtinen, Aleksi Lukkarinen, and Lassi Haaranen. 2021. Students Struggle to Explain Their Own Program Code. In *Proc 26th ITiCSE* (Virtual Event, DE). ACM, New York, NY, USA, 206–212.

[14] Ellie Lovellette, John Matta, Dennis Bouvier, et al. 2017. Just the Numbers: An Investigation of Contextualization of Problems for Novice Programmers. In *Proc 48th SIGCSE* (Seattle, WA, USA). ACM, New York, NY, USA, 393–398.

[15] Aleksi Lukkarinen and Juha Sorva. 2016. Classifying the Tools of Contextualized Programming Education and Forms of Media Computation. In *Proc 16th Koli Calling* (Koli, FI). ACM, New York, NY, USA, 51–60.

[16] Di Mayer and Perc Marland. 1997. Teachers' Knowledge of Students: a Significant Domain of Practical Knowledge? *Asia-Pac J Teach Educ* 25, 1 (1997), 17–34.

[17] Drew McDermott. 1976. Artificial Intelligence Meets Natural Stupidity. *SIGART Bull.* 57 (April 1976), 4–9.

[18] Craig S Miller. 2016. Human Language and Its Role in Reference-point Errors. In *PPIG; 27th Ann Wksp*. Psychology of Programming Interest Group, 10 pages.

[19] Donald A Norman. 2014. Some Observations on Mental Models. In *Mental Models*, Albert L. Stevens Dedre Gentner (Ed.). Psychology Press, New York, NY, USA, Chapter 1, 8 pages.

[20] Roy D Pea. 1986. Language-independent Conceptual "Bugs" in Novice Programming. *J Educ Comput Res* 2, 1 (1986), 25–36.

[21] Frederick Reif. 1987. Interpretation of Scientific or Mathematical Concepts: Cognitive Issues and Instructional Implications. *Cogn Sci* 11, 4 (Oct. 1987), 395–416.

[22] Alan Henry Schoenfeld. 1986. On Having and Using Geometric Knowledge. In *Conceptual and Procedural Knowledge: The Case of Mathematics*, James Hiebert (Ed.). Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, USA, Chapter 9, 225–264.

[23] Juha Sorva. 2007. Notional Machines and Introductory Programming Education. *Trans Comput Educ* 13, 2 (2007), 1–31.

[24] Roger Säljö. 1994. Adult Practices and Children's Learning. Communication and the Appropriation of Cultural Tools. *Eur J Psychol Educ* 9, 87 (June 1994), 87–91.

[25] Carl Winsløw. 2000. Between Platonism and Constructivism: Is There a Mathematics Acquisition Device? *For Learn Math* 20, 3 (Nov. 2000), 12–22.

[26] Inger Wistedt. 1994. Everyday Common Sense and School Mathematics. *Eur J Psychol Educ* 9, 2 (June 1994), 139–147.