Copying Can Be Good: How Students View Imitation as a Tool in Learning to Program

1st Carol Zander
Computing and Software Systems
University of Washington Bothell
Bothell, WA, USA
zander@u.washington.edu

2nd Lynda Thomas
Computer Science
Aberystwyth University
Aberystwyth, Wales, UK
ltt@aber.ac.uk

3rd Jan Erik Moström
Dept. of Computing Science
Umeå University
Umeå, Sweden
jem@cs.umu.se

4th Anna Eckerdal
Dept. of Information Technology
Uppsala University
Uppsala, Sweden
Anna.Eckerdal@it.uu.se

***Abstract*—Student 'copying' is often considered negatively as thoughts of plagiarism come to mind. Previously, we investigated the ways that instructors expect to use copying and imitation positively in their teaching. In this paper, we follow up that study by focusing on the student perspective and explore the ways in which students see copying and imitating as positive tools in learning to program (both at an introductory level and through more advanced learning of algorithms, etc.).**

In a qualitative research study, using semi-structured interviews, students were asked about how they use copying positively - their goals when they copy, how they go about it, how they view the experience of copying, and results beyond simply fulfilling their immediate goals.

When comparing student results with the previous study, it was noted that there is some level of agreement between instructors and students about how copying can be useful for learning to program. There is also some degree of mismatch between instructor and student views. Students did not report imitating instructors' approaches to learning and sometimes were unsure about whether they were supposed to copy the materials that they were given. This leads to some teaching suggestions in terms of instructors being more explicit in their attitude to this type of 'legitimate' copying and imitation.

***Index Terms*—Mimicry; Imitation; Copying; Follow-up study**

I. Introduction and motivation

In some fields, the need to imitate is openly acknowledged. For instance, when learning a foreign language one begins by imitating native speakers. In art, music, and sports, it is common for novices to imitate experts. In subjects like mathematics or history, teachers model things like constructing a good proof or writing a good paper, showing students how to become more expert in their chosen fields.

But in computing education, 'copying' generally has a negative meaning. Conversations between instructors about copying tend to focus on plagiarism - situations in which students copy without understanding for an assignment. But there are positive aspects to copying, and in fact, instructors expect students to copy, understand, and change the materials that they are given all the time. Börstler et al. [1] stress the importance of giving students good code examples precisely because students (consciously or unconsciously) model their code on what they have seen. Further, in a previous study [2], we found that instructors described how they used copying as a technique in their teaching to enhance students' learning. So, copying is not always bad in computing education. In this paper we report on the second step in this investigation by examining how students use copying and imitation when learning to program (both at an introductory level and through more advanced learning of algorithms, etc.)

In this paper, the term 'copy' will generally be used with artifacts and the term 'imitate' with behaviour, process, or attitude; 'mimic' will be used in the sense of Meyer and Land [3] who use the term to be synonymous with copy or imitate and also in relation to identity. When we are speaking generally, we will use the term 'copy'.

In this study, we have the following research question:

RQ How do students experience copying and imitation in learning to program?

When we broke this question down (as described in Section III), it became clear that we needed to examine:

- What goals do students have when they copy?
- How do students go about copying: From where do they copy; what do they copy?
- What are the results from their copying (intended and otherwise)?
- How do students view their (and others') copying?

and

- How do students' uses of copying fit with instructors' expectations as outlined in our previously developed framework?

The structure of the paper is as follows. Section II covers background and some related work. In Section III we describe our project and how we collected and analyzed data. In Sections IV and V we present the results of our analysis. In Section VI we present a deeper consideration of these results. Finally, we conclude in Section VII.

II. Background and Related Work

Several areas of research relate to how imitation is used when learning, and are discussed below.

A. Threshold concepts and liminal space

Our interest in this topic was sparked by the work of Meyer and Land [3], who proposed using *threshold concepts* as a way of characterizing learning new concepts. They define threshold concepts as being

- Transformative: they significantly change the way a student looks at things in the discipline;
- Integrative: they tie together concepts in ways that were previously unknown to the student;
- Potentially troublesome for students: they are conceptually difficult, alien, and/or counter-intuitive;
- Irreversible (probably): they are difficult for the student to unlearn.

Meyer and Land introduce the term *liminal space* for the transitional period from beginning to learn a concept to mastering it. They argue for a positive view of 'mimicry' and write that it can "involve both attempts at understanding and troubled misunderstanding, or limited understanding, and is not merely intention to reproduce information in a given form." [3, p. 377]. This, and the previous paper, are our attempt to investigate this contention further.

Eckerdal et al. investigated the liminal space from a student perspective, doing in-depth interviews of computer science majors nearing graduation [4], and found that "many students state that at some stage during the learning process they mimic what others have done without exactly understanding what they are doing" (p. 131). But "for the interviewed students the 'mimicry' seemed to be just a stepping stone in their further learning" (p. 129).

Meyer and Land connect mimicry to "stuck places" where it can be used as a coping mechanism and help students through the liminal space. They argue that using mimicry can be a useful strategy: "students might well adopt what appears to be a form of mimicry as a serious attempt to come to terms with conceptual difficulty, or to try on certain conceptual novelties for size as it were ... We would not want to belittle or dismiss such responses as they may well prove to be successful routes through to understanding ..." [3, p. 383].

B. Deep and surface learning

In some educational research, mimicry has been related to memorization and rote learning which a number of studies have shown to be less beneficial for learning [5]–[8].

The concepts of deep and surface learning, introduced in the 1970s [9], [10], fall into this area of research. In 1976 Marton and Säljö introduced the concepts of *surface-level* and *deep-level* processing, denoting "differences in what is learned, rather than differences in how much is learned" [9, p. 4]. This view on mimicry is seen differently from non-Western educational traditions and studies show repetition can be used to deepen understanding. [11], [12]

For a more extensive discussion on rote and shallow learning, see the previous instructor study on mimicry [2].

C. Social Learning Theory

Social, or observational learning theory, mainly developed by Bandura and Walters, discusses the learning of behaviour through observation and mimicry [13]. Observational learning describes a process where a student observes a model such as a friend or teacher, and in so doing improves her practical knowledge. Imitation here plays an important role as an efficient technique for behaviour change.

This seems to be in line with our previous work on how instructors want students to learn new processes through imitation [2]. Although we found some evidence of the instructors hoping that students would imitate their behaviour, we found that it was more common to encourage students to copy artifacts while reflecting on their content.

D. Procedural and Conceptual Knowledge

In the area of mathematics education research Sfard [14] discusses two views of knowledge, described in [15]:

- Operational or *process understanding* considers how something can be computed; the concept is regarded as an algorithm for manipulating things.
- Structural or *object understanding* describes a concept by its properties, treating it as a single entity.

Sfard describes three consecutive steps required for concept formation, from process to object understanding. Lister et. al [15] summarizes them:

- Interiorization: the student becomes familiar with applying the process to data.
- Condensation: the student abstracts the process into more manageable chunks, viewing the concept as an algorithmic process.
- Reification: the student understands the concept by its characteristics, and can be manipulated as a primitive object in other concepts. This is the most difficult of these transitions, as it involves a transformation of perspective.

Sfard claims that when students develop their understanding of a mathematical concept, a deeper understanding, i.e., the structural, is preceded by an operational understanding. We can see that many computer science students, in their initial struggle to understand, imitate algorithms given by the teacher for problem solving. For example, Sanders and Thomas [16] describe students copying useless code from a fish-tank program they had seen in lab as a starting point into their own paint program:

> "10 programs (of 14 submitted) included remnants of the code for the fish tank. These students appear to be experiencing code as no more than text that is merely copied and pasted rather than something with real world meaning." (p. 169)

Pegg and Tall [17] describe Dubinsky's APOS theory (Action, Process, Object, and Schema) as similar to Sfard's theory but extended in that it discusses "the transformation of action to mental objects" (p. 180). (p. 180).

This line of research builds on Piaget's work [18], and generally assumes that process precedes the development of object understanding (even though the APOS theory discusses going back and forth), and that object understanding, or schema, are the more advanced. Actions and process seem to be viewed as mere means to reach these goals. Lister et. al [15] contrasts this assumption in mathematics education to computer science education saying:

> "We believe this to be a fundamental difference between the two subject areas. Computer science has both practical and conceptual learning goals. The skills are not merely ways to reach the more sophisticated conceptual learning goals, but are goals in and of themselves." (p. 160)

E. The role of examples

Examples played a substantial role in both instructors and our students' use of copying. Students learn from instructors that examples can be used to learn programming. Growing from simple to complex examples is made explicit by Muller in a description of Pattern Oriented Instruction (POI) [19]: "The main principles ... lie in defining Algorithmic Patterns - solutions to basic algorithmic problems - and in organizing course problem-solving activities around them."

It is common to see examples used for teaching in textbooks. Börstler et al. discuss the importance of examples for learning [1] noting that "By continuously exposing students to exemplary examples, important properties are reinforced. Students will eventually gain enough experience to recognize general patterns which helps them to distinguish between good and bad designs." Morrison et al. [20] explain that examples include a combination of a problem statement and a step-by-step description of how to solve the problem (plus a concrete solution).

In a study on the difficulties for novice programmers, Lahtinen et al. [21] found that "example programs were considered the most useful type of material both by the students and the teachers." Börstler et al. agree [1]: "Students and teachers alike cite examples as the most helpful resource for learning to program."

In a study investigating the troublesome characteristic of threshold concepts, McCartney et al. [22] found that students in their final year of study use examples as a common strategy when stuck while practicing. They also found [23] that one of the strategies used by students doing informal or self-directed learning was to use examples, writing simple, then more advanced code examples.

Bransford [24] reports that the literature on transfer "suggests that the most effective transfer may come from a balance of specific examples and general principles, not from either one alone." Malan and Halland [25, p. 83] agree that "examples form an integral part of learning to program" but also that "the role of examples should go beyond merely illustrating concepts or principles and should 'sell' concepts to new programmers."

One underlying theory of how to efficiently use different examples is Variation Theory. From this perspective, Marton [26] discusses how students become aware of new *critical aspects* of phenomena, or relationships between such aspects when learning by using variation in the examples in specific ways. For example, too much variation between examples may hinder students from discerning the point the teacher wants them to see. Typically phenomena have been concepts, but it has been contended that they could also be skills [27].

III. Methodology

In this study we conducted semi-structured interviews with 10 computer science students, referred to here as S1 - S10. Participants included students at four institutions, in Sweden, the United Kingdom, and the United States. Eight were male, two female; all were at a point in their University's curriculum where they were competent in more than one programming language and at least halfway through their respective programs. The interviews were 30-60 minutes long. For analysis, they were transcribed verbatim; where necessary, they were translated into English by the interviewer.

Students were asked or sent an email invitation to see if they would agree to be interviewed saying that we were looking for examples of how copying or imitating could be used in a positive way in learning to program. And would they be willing to tell us about some programming techniques or code that they were encouraged or expected to copy or imitate?

After asking some basic questions about age/gender etc., the interview script 'imitated' the one used in the instructor study. Students were asked to describe situations in which they were expected to imitate materials or processes while learning to program. Interesting aspects were queried further. For instance, how would students now go about learning to program? What advice would they give others?

(See the appendix for this part of the script.)

Finally, we asked some questions that were prompted by the results of the previous study (see Section V). Had they imitated process as well as an artifact? What was their understanding of their instructors' motivation in expecting them to imitate? What did they find helpful, or otherwise, in their instructors' actions?

To analyze the interviews, we used inductive and deductive content analysis [28]. We read our own interviews and then began by individually reading and re-reading a couple of other interviews to gain a stronger grasp of the data. We then came together as a group and worked on two interviews, tagging all the aspects that related to imitation. We produced more than 30 initial tags. Using inductive content analysis, we then grouped our initial tags around emerging themes. We found that these themes could be represented by the illustration in Figure 1.

This graphical representation of the details of copying focused our thinking in that it articulates that 'copying' is a lot more than simply copying some materials and producing others. Obviously, the *copying activity* is central, but as Figure 1 shows, there is more than that. The student has some *goal* in mind when copying. With this goal, the student then takes some *materials* (supplied by the instructor, the internet, or others) and copies them, producing *resulting materials*. In the course of this activity, in addition to just new materials being produced, a *learning effect* occurs. The student sees this scenario from their own *perspective*.

For example, a student has the goal of trying to understand a concept, say a design pattern. The student copies a simple supplied example and plays around with it in the context of

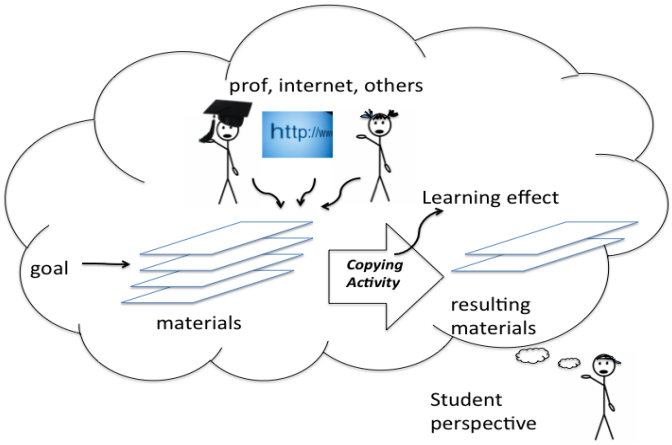

Fig. 1. Elaborating the copying activity

their work, eventually manipulating it into more sophisticated code. During this activity, there is a learning effect - the student acquires an understanding of the concept. The student has a perspective on what has occurred: "I copied in order to understand." They are satisfied that what they have done was both useful and legitimate.

These themes coalesced into the first four research subquestions itemized in Section I: What are student goals in copying; where and what do they copy; what are the results of copying; and how do students view their copying. The fifth research subquestion was analysed using deductive content analysis in order to examine how students' views fit with instructors' views as outlined in the instructor framework.

These subquestions then became the main headings in a table that was filled in for all the interviews. With at least two researchers working on every interview, we discussed the kinds of examples we saw as we examined the remaining interviews. We looked for new themes to emerge, but our initial analysis sufficed for this group of students. (Qualitative studies are not intended to show definitively what a whole population of students believe or do, but rather aim to understand the experience of the subjects, giving a rich description which may be indicative of general trends. Ten informants is normally considered sufficient for such a study. [29])

The majority of the tagging was done while we were together in person. We then finished the process after returning to our respective homes, with communication over Skype.

IV. Results - Student Experiences

A. What goals students have when they copy

Students use copying with two goals in mind: to better understand so that they learn, or to get things, like assignments, done.

Copy to understand and learn. Most of the students discuss copying in positive terms, saying that they use it to reach a better understanding and to learn. Student S2 explains how Java graphics was learned when copying code. Graphics requires a number of existing Java packages:

> *Student S2:* "you can't start from the beginning for it's too much work, unnecessary work. So we start, and then ... they give examples, this one creates a button and this creates a box and stuff like that. And you can copy those generally ..."

The student thinks that copying in this manner, using a "step by step building" is an efficient way to learn. Student S7, in a Software Engineering course, copied "more a design, a way of thinking" rather than code.

In a networking course, student S3 appreciates example code, explaining how "when you first learn how to start up a socket, it's complicated. It's not the easiest of code to read. But then once you kind of do it a few more times and you see it again and again in the programming assignments, it makes perfect sense."

Copy to "get things done". The examples of this kind of goal vary from copying without understanding to instructors helping students with assignments by providing code to copy.

Lack of time and other constraints can pressure students to copy code without understanding it. Student S2 discusses a large assignment (a recursive descent calculator) in the second beginning programming course:

> *Student S2:* "a lot of code was already given. You didn't really copy it, but you didn't always have to understand how things worked ... And then we found some help on ... Stack Overflow. And you didn't really understand how all the pieces worked together you just realized ... they do."

Student S9 explains how the instructor paved the way: "we had specific practicals leading up to the assignment. He [the instructor] said, "You can't work on the assignments until you've finished all these practicals and exercises" which basically laid out how you do it ..." A similar example is provided by student S5 who says "if you went to classes ... then you basically had the entire program."

B. How students go about copying

The central materials mentioned were given by instructors or found through web searches (mostly resulting in Stack Overflow or GitHub). Student S9, however, mentioned both books and 'cheat sheets' initially given by the instructor and then extended by the student. But mainly, students used materials given to them in class or found on the web. Students report copying from instructors' lecture notes and examples, recognising that they were supposed to build on examples given by instructors. Student S5 says: "This professor will actually show us how to do certain ... things in C++ [that] we can actually use in our program and he clearly lets us use it, ... sometimes as is, and sometimes you have to make changes ..." This supports our previous research with instructors, what they are hoping will happen. There may be some confusion, however, in what students think they should be doing, and in what is allowed to be copied:

Student S1: "They didn't say copy the old thing and improve it, but…it was clear that this is the same thing…but a little bit more advanced. …I wouldn't say I was encouraged to do it …The similarities were apparent."

Student S5 also reflects this confusion: "I was never told by instructors 'Hey you can take this and you can use that.' But I did do that and …I wasn't marked down for it." However, some students complained that the material they were given as a basis in this way was not sufficiently well integrated:

Student S8: "I can't remember a single point where he showed malloc in use properly. He told us what it was, what it did, but at least, there wasn't a good example …where he showed, 'Here I use malloc because, and here I use calloc because …'"

When students are trying to understand something, they reported copying and modifying for themselves. For example, Student S4 remarks "I create another file that I can mess around with and I don't care if it breaks." S4 also copies things non-digitally: "I get the basic stuff down and then I force myself to go back through the steps. That's why I like to do most of my stuff on the whiteboard and erase everything, so I can't look back at a previous problem."

C. What are the results of their copying

The students we interviewed believed that copying should result in both fulfilling short-term goals (completing assignments) and learning, They generally did not discuss results beyond fulfilling these immediate goals, but some did consider the process of copying in a more abstract way in the context of their overall development.

Student S4 discusses how "mimicking is, is only good to start with, but an over-reliance is" not a good thing. Moreover,

Student S4: "If you are always going to Stack Overflow or googling, you spend a lot of time copying and pasting and not really knowing why this works …I like to …Not just mimic the code, but go in depth and see why this works …"

Student S6 copies code, but avoids straight "copy and paste," striving to "learn and then try to use parts of it …to know more about how it works." Student S5 uses copying and modifying to try to understand a system:

Student S5: "Usually I try to do it myself first and if I'm hitting some piece where I feel like I don't quite understand it fully, or it's not working right, then I'll check his code. If I need to get something working, …I'll copy his piece and paste it in to get it working so I can debug the entire system and then go back and like, 'Okay, let's analyze why this works the way it does'"

S5 also gains confidence from copying. Admitting being "not that good …but the biggest thing …was simply being aware of that stuff and knowing that I can figure it out if I need to."

Student S10 contends that certain small syntax-like code, like code for opening a file, does not need to be remembered and can be copied when needed. More advanced copying can lead to transfer of knowledge as, for example, S8 recognizes that after learning Pascal, it was easy to learn another language.

Students acknowledged that they often google and use Stack Overflow to find code to mimic. While they appreciate the vast amount of information available, fortunately they also acknowledge the perils of information found on the web. S4 emphasized using course materials before going to the web for information. They wisely noted that "the class examples probably are going to help me more than Stack Overflow because it's more tailored for the assignments."

D. How students view their experience of copying

The students mostly have a positive view of how copying aids learning. Their opinion is that copying helps them understand both the details of the code and how code fits together as exemplified by Student S7:

Student S7: "we are often given material in courses, …solve a small problem …but then the problems we should solve are much bigger so then you have to join the small pieces together and add to your previous knowledge."

Copying also saves students time and effort by being able to use existing code as a starting point for learning. Student S5 says "I don't understand this yet then I'll use that example to help understand it." Moreover, Student S7 thinks it is "fantastic …to get example how it's used …", making "it easier, absolutely."

Student S4 recognizes the dangers of blindly copying without taking the effort of understanding, saying that "it's very good to get you started, but if you want to expand the topic, learn more things, that's something that you can't just do with just mimicking, you have to go out on your own." And while tutoring a beginning programming student, S4 observes a lack of learning:

Student S4: "He just imitated, he didn't pay attention in lectures and he just went and googled everything. …and I'm like but this doesn't apply to your problem. So an over-reliance on imitating can be bad."

However, many students also admit that they themselves are sometimes doing just this. They justify it with the demands of meeting deadlines. Some students say that later, after having met the deadline, they go back and study the code in order to really understand what it does. Notably, one student, S1, says "in my experience understanding always comes after a course."

V. How students' views fit into the instructors' framework

Instructors described situations where they demonstrated code or expected students to copy or change code. But

instructors also hoped that students will imitate their process, their behaviour, and even their attitude (not just to coding, but to learning). For a rich discussion of this, see [2].

From the instructors' interviews, we saw a wide variety of imitation expectations ranging from exact copying to abstraction. The different *kinds* of imitation were organized as artifact vs. process, and the *level* of copying, from exact to abstract. This continuum, with representative examples, is represented in Figure 2.

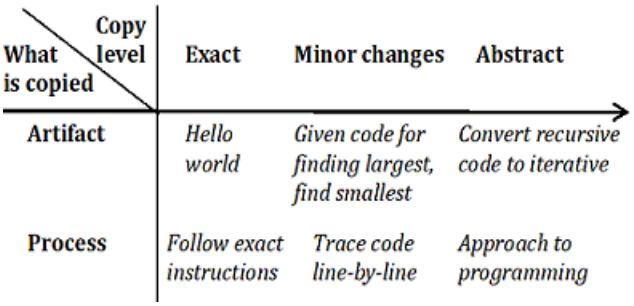

Fig. 2. Instructors' continuum of imitation framework. (Examples are given in italics.) [2]

Although students did not often talk about these situations, they exhibited each of these categories. They talked more about artifacts than process, but there were examples of each type of copying described by instructors.

A. Artifacts

At times, instructors expect students to copy exactly what is given to them. S8 admits to verbatim copying of code:

> *Student S8:* "it's just a matter of getting the code that does the thing you want but not necessarily understanding [it]"

The student rationalizes this type of copying: "I feel that I could understand it if I spend some time on it."

More often, instructors give examples and expect students to copy, yet vary the code. Carefully designed examples help students to grow their problem-solving skills. Student S6 described making minor changes to code by recognizing similarities in algorithms and using bubble sort to help implement insertion sort.

As students mature in the computing discipline, instructors hope they will see abstraction in what is presented and be able to make major changes. S3 exhibits this deeper level of imitation:

> *Student S3:* "Sometimes we have to do additional research to understand the underlying data structure itself ...This will show us an example of it, of how to use a piece of it. But then we'll have to go and research additional parts of it to get it to work with our program."

B. Process

Sometimes students are told to follow precise instructions in doing something. For example, student S9 described how the instructor said to comment their code: "He did say there was a set way ... you should be commenting ... so we copied that."

It is more common, especially beyond beginning programming courses, to have students make changes when applying a process. Students learn how to employ a process in a similar situation to that which has been shown by the instructor. For example, in a networking course, student S3 learned the process that when "you have to rewrite a program to work like it does, but you don't get the code for it" that "you have to mimic what it's doing by using tcpdump."

Instructors discussed situations where they intended that mimicry, more in the sense of Meyer and Land, take place. They hoped that students would imitate an approach such as problem solving, or their attitude to computing or to learning. Student S3 exhibits learning the abstract process of applying software engineering techniques: "after you take [a software engineering course] where you learn about refactoring and some more patterns. Then you can be like 'Hey, here's some code. Write it in a builder pattern' ."

VI. Discussion

A. Threats to Validity

Before the results are discussed, several threats to the validity of our study are acknowledged. In common with all qualitative research, and as discussed in Section III, this project interviewed a small number of students, and we report on what these informants have told us. In other words, it is important to consider what group of students we interviewed (different countries, different places of study, somewhat different places in different curricula, etc.). The students were asked to participate in the study and while it is unknown why students chose to participate, a possible motive might be their own interest in learning. This means our subjects might not represent the whole student population in their view of copying.

Another issue is the way we approached them. Asking for "positive" ways is which imitation is used when learning to program, and the fact that they were talking to instructors, made it unlikely that they would talk about outright plagiarism. Some mentioned other students plagiarising, but all justified their own actions as more or less legitimate. Plagiarism was not the topic of this paper, but students who do outright plagiarise were unlikely to answer our call for interviewees.

Finally, we asked the students to describe how they had experienced various events. In one case, first learning to program happened many years ago in high school. This, of course, meant the student described events in a way that might not reflect what actually happened.

B. Results in context

Despite these validity concerns, we find it encouraging that the students described their experiences in similar ways. This makes us think that the our findings are applicable outside this group of students.

Students are often more thoughtful about imitation than instructors give them credit for. They appear to understand

its uses as well as its limitations. They recognize when they and other students are using imitation inappropriately rather than to actually learn.

So, students express an understanding of copying which is in line with the first steps in Sfard's [14] and the APOS's [17] theories. When copying while learning something new they "become familiar with applying processes to data." But all of the students talked about the need for deeper understanding. Even if students acknowledge that a particular act of copying is shallow, they also recognize that copying can be an important step to reach a deep understanding, both of concepts and skills.

Our initial motivation for this study was Meyer and Land's discussion of 'mimicry' in the context of threshold concepts. We did, indeed, find that students used mimicry when they were 'stuck' and to get them through the liminal space to a place of understanding.

C. Good and bad copying

Previous studies report that copying does not support learning. For example, Dewey [5, p. 41] discusses positive and negative copying saying that "imitation of ends ... is a superficial and transitory affair". However, he makes a distinction between imitation of ends and imitation of means which help to reach ends. Thus, copying can be beneficial for learning. We observe from the data that the students in the present study as well as students in a former study [4] claim that they use copying in ways which seem to be in line with Dewey's description of "imitation of means".

Our students observed when other students are copying without any learning occurring. In a few instances, they recognized when they themselves were copying and not learning. Student S2 admits to not writing much code in one course with "the only thing we did were like change parameters and maybe change some loops for cross validation" confessing that "it didn't really stick with you." Even in this case, it is recognized that "imitation of means" is the way to learn.

Another observation is that few of the examples of copying described by the students involve memorization and rote learning. This sentiment was also seen from the instructors [2]. Copying in computer science seems to have less to do with memorization since most problems students encounter require understanding. Students used it more when they were beginning and less as they progressed to more advanced courses. Students of computer science appear to understand that copying without understanding is useless in the long run.

D. Relation to instruction

The examples of copying and imitation we have seen are at different levels ranging from shallow to deep. This is appropriate for students at different stages of their education and supports the instructor framework of Figure 2. Exact copying or copying making small changes helps beginners to get started, while more advanced students tend to tinker more and make major changes.

Students talked about how some instructors "expected" them to copy materials and change them. There was some confusion as to whether this was legitimate, and instructors should note this - making their intentions transparent can only help students learn.

The students talked almost exclusively about copying code and said little about copying process, whereas the instructors hoped that students would imitate their process, and even their approach to programming and to learning. Perhaps this is absorbed unconsciously, but again instructors might consider making their intentions more obvious.

Students also talked about good and bad materials; materials had to be shown in context, or they were not helpful. Students noted that materials from Stack Overflow, and the like, were less focused and could be hard to use.

VII. Conclusions and Future Work

The major contribution of this paper is to illustrate how students use a wide variety of types of copying to aid with their learning and that students are conscious of this use. They are well aware that copying materials without understanding is not beneficial to them. Although they were prompted to give positive ways in which copying is used, their responses were thoughtful and exhibited depth. Students think that copying in an important tool in their learning process.

There is considerable future work to be done in this area. Some research questions include:

- How can students be encouraged to move from shallow to deeper forms of copying?
- How can *weak* students learn to copy appropriately?
- Do others use copying in the same manner? For example, is there a difference between novice and seasoned programmers? How do practitioners compare?
- What is the best approach to investigate imitation of process, which (unlike code copying) does not leave a 'trace' that can be examined?

VIII. Acknowledgments

Thanks to the students who took the time to meet with us and to our co-authors on the previous paper (Robert McCartney and Kate Sanders). This work was supported by MINT, the Centre for Discipline-Based Education Research at Uppsala University.

References

[1] J. Börstler, M. Nordström, and J. H. Paterson, "On the quality of examples in introductory java textbooks," *Trans. Comput. Educ.*, vol. 11, no. 1, pp. 3:1–3:21, Feb. 2011. [Online]. Available: http://doi.acm.org/10.1145/1921607.1921610

[2] C. Zander, A. Eckerdal, R. McCartney, J. E. Mostrom, K. Sanders, and L. Thomas, "Copying can be good: How instructors use imitation in teaching programming," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 450–456. [Online]. Available: https://doi.org/10.1145/3304221.3319783

[3] J. H. Meyer and R. Land, "Threshold concepts and troublesome knowledge (2): Epistemological considerations and a conceptual framework for teaching and learning," *Higher education*, vol. 49, no. 3, pp. 373–388, 2005.

[4] A. Eckerdal, R. McCartney, J. E. Moström, K. Sanders, L. Thomas, and C. Zander, "From limen to lumen: Computing students in liminal spaces," in *Proceedings of the Third International Workshop on Computing Education Research*, ser. ICER '07. New York, NY, USA: ACM, 2007, pp. 123–132. [Online]. Available: http://doi.acm.org.ezproxy.its.uu.se/10.1145/1288580.1288597

[5] J. Dewey, *Democracy and education*. Courier Corporation, 2004.

[6] J. Lithner, "Principles for designing mathematical tasks that enhance imitative and creative reasoning," *ZDM*, vol. 49, no. 6, pp. 937–949, 2017.

[7] J. Hiebert, "What research says about the nctm standards. i j. kilpatrick, wg martin, d. schifter & national council of teachers of mathematics.(red)," *A research companion to Principles and standards for school mathematics*, 2003.

[8] D. G. Pérez and J. M. Torregrosa, "A model for problem-solving in accordance with scientific methodology," *European Journal of Science Education*, vol. 5, no. 4, pp. 447–455, 1983.

[9] F. Marton and R. Säljö, "On qualitative differences in learning: I–outcome and process," *British journal of educational psychology*, vol. 46, no. 1, pp. 4–11, 1976.

[10] N. J. Entwistle, "Approaches to learning and perceptions of the learning environment," *Higher Education*, vol. 22, no. 3, pp. 201–204, 1991.

[11] B. J. Cooper, "The enigma of the chinese learner," *Accounting Education*, vol. 13, no. 3, pp. 289–310, 2004.

[12] D. Kember, "Misconceptions about the learning approaches, motivation, and study practices of Asian students," *Higher Education*, vol. 40, pp. 99–121, 2000.

[13] J. E. Grusec, "Social learning theory and developmental psychology: The legacies of robert r. sears and albert bandura." 1994.

[14] A. Sfard, "On the dual nature of mathematical conceptions: Reflections on processes and objects as different sides of the same coin," *Educational Studies in Mathematics*, vol. 22, pp. 1–36, 1991. [Online]. Available: http://doi.org/10.1007/BF00302715

[15] R. Lister, T. Clear, D. J. Bouvier, P. Carter, A. Eckerdal, J. Jacková, M. Lopez, R. McCartney, P. Robbins, O. Seppälä *et al.*, "Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer," *ACM SIGCSE Bulletin*, vol. 41, no. 4, pp. 156–173, 2010.

[16] K. Sanders and L. Thomas, "Checklists for grading object-oriented cs1 programs: Concepts and misconceptions," *SIGCSE Bull.*, vol. 39, no. 3, p. 166–170, Jun. 2007. [Online]. Available: https://doi.org/10.1145/1269900.1268834

[17] J. Pegg and D. Tall, "The fundamental cycle of concept construction underlying various theoretical frameworks," in *Theories of Mathematics Education*. Springer, 2010, pp. 173–192.

[18] E. W. Beth and J. Piaget, "Mathematical epistemology and psychology, trans," *W. Mays (Dordrecht: D. Reidel)*, 1966.

[19] O. Muller, "Pattern oriented instruction and the enhancement of analogical reasoning," in *ICER '05*, 2005, pp. 57–67. [Online]. Available: http://doi.acm.org/10.1145/1089786.1089792

[20] B. B. Morrison, L. E. Margulieux, and M. Guzdial, "Subgoals, context, and worked examples in learning computing problem solving," in *ICER '15*, 2015, pp. 21–29. [Online]. Available: http://doi.acm.org/10.1145/2787622.2787733

[21] E. Lahtinen, K. Ala-Mutka, and J. Ärvinen, "A study of the difficulties of novice programmers," in *ITiCSE-05*, 2005, pp. 14–18.

[22] R. McCartney, A. Eckerdal, J. E. Mostrom, K. Sanders, and C. Zander, "Successful students' strategies for getting unstuck," *SIGCSE Bull.*, vol. 39, no. 3, pp. 156–160, Jun. 2007.

[23] R. McCartney, A. Eckerdal, J. E. Moström, K. Sanders, L. Thomas, and C. Zander, "Computing students learning computing informally," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, ser. Koli Calling '10. New York, NY, USA: ACM, 2010, pp. 43–48. [Online]. Available: http://doi.acm.org/10.1145/1930464.1930470

[24] J. Bransford, A. Brown, and R. Cocking, *How People Learn*, expanded edition ed. National Academy Press, 2004.

[25] K. Malan and K. Halland, "Examples that can do harm in learning programming," in *OOPSLA-04*, 2004, pp. 83–87.

[26] F. Marton, *Necessary Conditions of Learning*. New York & London: Routledge, 2015.

[27] K. Sanders, J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, L. Thomas, and C. Zander, "Threshold concepts and threshold skills in computing," in *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ser. ICER '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 23–30. [Online]. Available: https://doi.org/10.1145/2361276.2361283

[28] P. Mayring, "Qualitative content analysis," *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, vol. 1, no. 2, 2000, accessed March 16, 2019. [Online]. Available: http://www.qualitative-research.net/index.php/fqs/article/view/1089

[29] O. Hazzan, Y. Dubinsky, L. Eidelman, V. Sakhnini, and M. Teif, "Qualitative research in computer science education," *Acm Sigcse Bulletin*, vol. 38, no. 1, pp. 408–412, 2006.

Appendix

Semi-structured Interview Script

Introduction:

We are looking for examples of how imitating can be used in a positive way in learning to code. Can you show me some programming techniques or code that you were encouraged or expected to copy or imitate? From here on, I will refer to these as "materials", but they may be in any form: online materials, hard-copy, something you saw in class, etc.

Preliminary information:

1) How old are you?
2) How far along in university are you?
3) What is your gender?
4) When did you first learn to program?

Interview questions:

1) Tell me about these [materials] that you were expected to imitate. Think about both artifacts (like examples in an induction proof) and process (like imitating the structure of an induction proof).
    a) How did you know that you should imitate? Did the instructor say so explicitly? Did you just do it because you were expected to do something similar to the [materials]?
    b) Describe the [materials] briefly.
    c) What class/course do you use these [materials] in? In what context (i.e., in class, in a lab, when working on a programming, etc.)?
    d) What did you learn from this imitation? [A concept? How to go about solving something? Think about artifacts or process.]
    e) Did the instructor go over the [materials] in class? Or provide any added written or verbal explanation?

2) What part of these materials were you expected to imitate?
3) Do you think this imitation is effective in helping you to learn? How/Why?
4) How do you use the thing you imitated in your programming assignments? (How close of a relationship? How close of a transfer? How much variation?)
5) What other online materials, hard-copy, something presented in class, etc. are you expected to imitate?
6) Do you use imitation in other classes? In the same way or differently?
7) Is there anything else you imitate when learning to program, inside or outside the university? Describe the whole experience. [Something that your instructor didn't provide, e.g., from the Internet, friends...]