Mastery Learning and Productive Failure: Examining Constructivist Approaches to teach CS1

Cruz Izu, Daniel Ng, Amali Weerasinghe

School of Computer Science, The University of Adelaide
(cruz.izu,amali.weerasinghe)@adelaide.edu.au

Abstract. The struggles of novices taking introductory computer science courses to master basic constructs and develop an understanding of the notional machine continues to drive computer science education in the search of new pedagogical approaches. This work examines in depth two recent proposals: mastery learning and productive failure. Both approaches are grounded by constructivism, which should reduce the challenges that CS1 students face when learning to code. By exploring the concepts that drive these pedagogical approaches, this study aims to make constructivism more accessible to CS0/CS1 teachers. The two approaches illustrate and highlight key concepts that support constructive learning.
The main outcomes from this work are the concept maps generated for each pedagogical approach, along with descriptive tables of their concepts. Both approaches support constructive learning by utilising (1) adaptive instruction that aligns with the current constructed knowledge of students, and (2) the use of student's failure as key to identify knowledge gaps and improve learning.

Keywords: constructivism, active learning, failure, prior knowledge

1 Introduction

Core pedagogical concepts, such as scaffolding and learning trajectories, are used by tertiary teachers to structure their instruction (Anderson & Gegg-Harrison, 2013; Rich, Strickland, Binkowski, Moran, & Franklin, 2017; Wass & Golding, 2014). These approaches are not sufficient to teach computer science effectively, as shown by lower pass rates (mean pass rate of 67.7%) identified in the systematic review of 161 CS1 courses taught in 15 different countries Watson and Li (2014).

More importantly, many students that passed a CS1 course still struggle to "see the forest from the trees" (Lister, Simon, Thompson, Whalley, & Prasad, 2006) or even understand their own code (Lehtinen, Lukkarinen, & Haaranen, 2021). These documented struggles may explain the high attrition rates which in the past have been attributed to the idea that computing ability is innate (Hoda & Andreae, 2014; Robins, 2010), but may also be caused by poor pedagogy. Hence, further steps are needed to improve computer science education (CSE) at introductory levels. (Robins, 2010; Watson & Li, 2014).

A classic theory of learning, such as objectivism, might suggest that students passively gain knowledge from textbooks or lectures (Jonassen, 1991). Constructivism, a more modern theory of learning, suggests that this is not the case. Instead, it implies, that students learn by actively constructing and connecting knowledge recursively (Brooks & Brooks, 1993; Waite-Stupiansky, 1995).

Ben-Ari (2001) presented a theoretical analysis of computer science education (CSE) from a constructivist paradigm, focusing mostly on novice programmers (Ben-Ari, 2001). He suggests that some struggles students face in CS, such as building mental models and connecting concepts coherently, can be explained by constructivism. Robins (2010) describes in a similar vein the tightly integrated nature of the concepts comprising a programming language that creates an inherent structural bias in CS1 which drives students towards success, if they managed to quickly grasp early concepts, or failure if they don't.

Ben-Ari lists 3 pedagogical principles that support constructive learning (Ben-Ari, 2001):

1. A preference for **active learning** to enable the student to construct mental models.
2. Recognition of the importance of **pre-existing knowledge**.
3. The employment of the inevitable errors and misconceptions as a pedagogical device rather than as a symptom of **failure**.

Thus we want to explore in more depth recent approaches that are aligned with the three above principles so that teachers can support students in building correct mental models of programming plans and the notional machine. This work aims to examine and explore some pedagogical approaches that seemingly support constructive learning. After an initial review of the literature, we choose depth versus breath and focused only on two approaches: Mastery learning (ML) have already been applied to CS1 courses and have demonstrated positive

results (McCane, Ott, Meek, & Robins, 2017; Gorp & Grissom, 2001); Productive failure (PF) is another interesting approach which has shown positive results in high school maths (Kapur, 2008) and seems to be a good fit in terms of constructive principles.

This paper revises these two pedagogical approaches, highlighting the core underlying concepts of each and utilizing concept maps to clearly convey the concepts and how they relate to each other. Whilst there are already a number of papers covering PF and ML, there is value in collating and revising their contributions, so we can draw out the core concepts of each approach. This can help teachers to better understand the underlying mechanics of these pedagogical approaches, making it easier for adoption in their classrooms. The analysis could also help to drive further research by suggesting different pedagogical mechanics for investigation.

The rest of the paper is organised as follows: after a literature review in section 2, a methodology section follows; section 4 presents the concept maps derived from the analysis and its associated tables. Section 5 discusses how each pedagogy supports constructive learning and section 6 provides the conclusions of this study.

2 Literature review

This section starts with a brief review of Constructivism before moving to the two selected approaches, ML and PF that apply constructivist principles. Then we will provide a brief description of each approach, summarising some of its applications, results recorded and why they are of interest to CS educators.

2.1 Constructivism

As the educational resources are ubiquitous and freely available online, the challenge for the current generation of students is to receive guidance to develop coherent mental structures that facilitates their application. Thus, active learning has been changed from being passive to active with many approaches being tried and evaluated in recent years, including problem-based learning (Greening, Kay, Kingston, & Crawford, 1996), team-based learning (Michaelsen, 2014), and peer instruction (Zingaro & Porter, 2014).

Constructivism provides explicit support to build coherent mental models by focusing on prior knowledge and also utilizing failure as a learning opportunity. However, there is no *explicit* mechanism in the active learning approaches cited above to build on pre-existing knowledge and using failure as an opportunity to learn. A large body of work in CS education has focused on identifying issues in prior knowledge by detecting and reporting misconceptions (Caceffo, Wolfman, Booth, & Azevedo, 2016; Kaczmarczyk, Petrick, East, & Herman, 2010). However, it is left to CS1 instructors the task of developing activities that correct or cross-examine those misconceptions.

As successful learning appears to require that the student reach an impasse (VanLehn, Siler, Murray, Yamauchi, & Baggett, 2003), failure provides that impasse that increases learning. Educators needs to help students to cultivate a growth mindset (O'Rourke, Haimovitz, Ballweber, Dweck, & Popović, 2014), enabling students to understand that not being able to complete tasks or failing at tasks is a part of learning. Once students have identified the reason to fail, usually a knowledge gap, they are more receptive to instruction to reduce or eliminate that gap as it is more relevant to them. However, as failure has the potential to discourage some students, we should be aiming for small failures that provide a sense of progress or "being nearly there".

Some works support the idea of learning from errors and used a constructivist approach to teach a particular difficult topic. For example, Moreno, Sutinen, and Joy (2014) used conflictive animations of code to help students reflect and learn about operators, expressions and statements. Ginat and Shmalo (2013) used tasks that contains errors to explore OOP concepts. Although inspiring, such approaches are not transferable to teach other topics, limiting their use in the classrooms. However, both productive failure and mastery learning appear more suitable for wider use, as explained in the rest of this study.

2.2 Mastery Learning (ML)

The underlying premise of ML is simple, course content is broken down into sequential parts, where each part builds on from the prior part (Mercer, 1986). Students are required to "master" a part (e.g achieve 90% on a diagnostic task) before moving on to the next (Mercer, 1986). Figure 1 illustrate this process: students must master *core 1* before starting to work on *core 2* . ML allows students to work at their own pace, helping to ensure that they have adequately learned each part before moving on to the next (Mercer, 1986). Extension work is provided for top students that master the topic first so that the whole cohort is able to move at the same time to the next topic.

The sequential structure of ML seemingly supports students in connecting concepts together and building effective mental models. This makes ML a particularly compelling candidate to use in CS1 as each program statement works in combination and relies on the knowledge of other statements. by providing either remedial work or extension work, all students benefit regardless of their performance.

ML has been applied to a variety of contexts, including high school mathematics (Zimmerman & K. Dibenedetto, 2008) and introductory programming at university (McCane et al., 2017). Both ML approaches were shown to

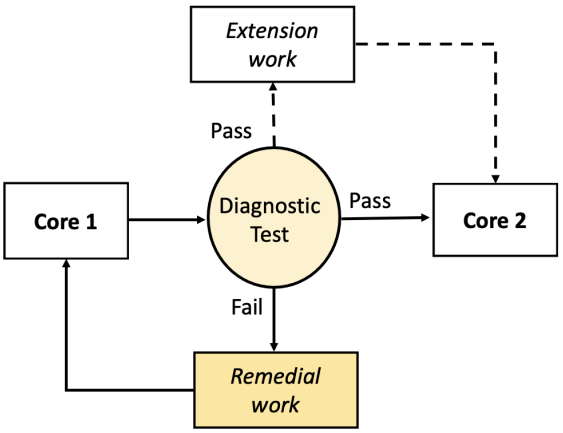

Fig. 1: Mastery learning visualised process from(Mercer, 1986)

increase student performance (McCane et al., 2017; Zimmerman & K. Dibenedetto, 2008). Research has also been shown mastery learning leads to stronger engagement from students and a greater desire to learn (McCane et al., 2017; Mercer, 1986; Zimmerman & K. Dibenedetto, 2008).

Mastery learning requires rethinking the whole course, thus it does not allow for small trials. However, ML has shown promising results and in particular it seems to be more effective in poorer performing students, when compared to high-achieving students (McCane et al., 2017). These traits make ML strong candidate for helping to improve CS1 education.

2.3 Productive Failure (PF)

Traditional instruction start with describing and demostrating the application of a concept to students and then providing problems for students to work on.

PF **flips** the order by getting students to start with the problems first, before receiving instruction regarding the concept from the teacher. This is described in a PF alternative PS+I (problem solving plus instructions) as delayed instruction (Loibl, Roll, & Rummel, 2016). The essence of PF is that students will fail the complete the task given, as they haven't learnt the concept required to do so yet. That failure, caused by delayed instruction (Loibl et al., 2016), together with a reflective teaching of the concept afterwards is key to improved the learning of that new concept by students, as well as developing their cognitive skills (Kapur, 2008; Kapur & Bielaczyc, 2012)

PF has been applied to a range of educational levels, but primarily in the domain of high school mathematics and it demonstrated to improve the performance of students (Kapur, 2017; Kapur & Bielaczyc, 2012; Loibl et al., 2016; Westermann & Rummel, 2012). Figure 2 illustrate one of the main reasons behind PF's effectiveness: the attempt to problem-solve helps students to be aware of their knowledge gaps as described in Figure 2, as well as reflecting on the task at hand and its key features. Besides, PF helps students to construct their conceptual knowledge (Loibl et al., 2016) by providing just-in-time instruction after the gaps are identified, making that instruction more relevant to students. This increases student's engagement and awareness of the relationship between that new concept and the knowledge they have used in their failed attempt.

Similarly to ML, PF has been reported to be more effective in poorer performing students (Kapur & Bielaczyc, 2012), hence it appear to be a good candidate for teaching in a computer science context. Note PF is more flexible than ML in that we can choose a topic or a workshop to trial it out and gradually build up its application and our expertise using it. We are only aware of one work that reports the use of PF in an IT-related topic.

3 Methodology

The previous section has presented a brief literature review of ML and PF. Our task now is to examine them from the constructivist perspective. For each pedagogical approach explored in this work, a concept map and table of concepts is created. Each concept map redraws the figure from the literature (figs 1 and 2) in order to reflect the links of each approach to constructivism. The associated table aid in communicating how each concept applies or is shaped in each context.

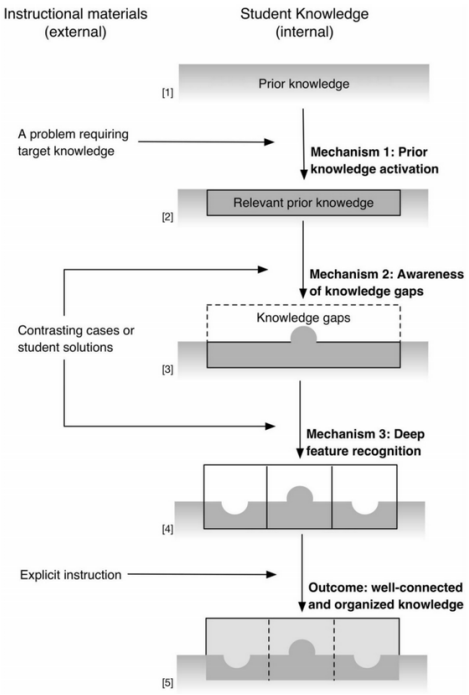

Fig. 2: Productive failure visualised process from(Westermann & Rummel, 2012)

3.1 Concept maps

Concept maps are great tools for expressing the key components of a topic and how they relate to each other. They comprise of concepts, regularities seen in the topic, and propositions, relationships between concepts (Novak & Cañas, 2006). Though a seemingly simple tool, the theoretical foundations of concept maps are rooted in psychology and they have proven to be effective (Novak & Cañas, 2006).

This work uses concept maps to explore the pedagogical approaches and help communicate their underlying concepts. Concepts maps are not only a good graphical representation of information, but the process of creating them is an effective research tool (Novak & Cañas, 2006). The use of concept maps helps to explore the underlying mechanics of the pedagogical approaches and relay the findings effectively. A concept map was made for each pedagogical approach, exploring the concepts of its application and how they are interrelated. The mapping process was broken down into three main steps:

Literature review We decided to prioritise depth versus breadth by focusing only on core or seminal papers. Three to four papers for each pedagogical approach were selected and annotated. Key concepts were extracted by observing the applications between papers and noting the concepts/practices which were consistent between them. Once concepts were listed, they were reexamined to consider which would be better represented as a relationship between concepts.

Individual concept maps Once concepts and relationships were identified, preliminary concept maps were sketched. How concepts fitted into the map, how concepts related to each other and even the concepts themselves were reviewed in an iterative process by the first two authors until they captured the essence of ML(PF).

Concept map revision: Finally, the two maps were compared to find commonalities: looking at the concepts across both maps and using the same terminologies and structures where possible. At this stage, it was also recognized that the cross-links between concepts, rather than the hierarchical links, were more important to demonstrate the mechanics of each pedagogical approach. Because of this, the final maps were redesigned to be non-hierarchical.

3.2 Tables of concepts

In the last step, we identified the need to provide detailed definitions to complement the concept maps. Thus, the definitions were collated in tables to accompany each concept map. Tables are an efficient way of indexing information and relaying it to readers. They could help readers to quickly understand the contexts where each pedagogical approach can be applied and the concepts included in its implementation.

To keep them consistent, during the research a generalised structure for the tables was created. Each table has a header depicting the context for which the pedagogical approach can be applied and concepts relating to the overall structure of the approach. In our context, the readers would be educators that may consider integrating an approach into their classroom. Thus, we aim for a practical description that groups concepts by their practical roles such as teacher input or student tasks.
The concepts in the concept maps can be considered to be in one of four areas:

Knowledge Concepts related to the knowledge students have, require or lack.
Teacher input Concepts that describe teacher actions to aid students to complete or learn a task.
Student tasks Concepts that describe hands-on tasks given to students.
Response to student results Concepts that pertain to the results of students.

4 Results

4.1 Mastery Learning

Mastery learning considers that students need to connect existing knowledge with the new knowledge being taught in order to adequately learn a concept. To support this model, content is broken into subsequent components. The sequence of components makes it so that the new knowledge acquired in one component becomes part of the existing knowledge required to complete the next component.

Instruction for a component teaches new content that builds on from the last component. As the overarching structure for content is sequential, the instruction implicitly implies the need for students to utilize the existing knowledge gained from the last component.

The prime goal of the task/diagnostic in mastery learning, is to verify that students have adequately learned the component. Test failure serves as an opportunity for students to see their knowledge gaps and take additional adaptive instruction. The instruction is specifically catered to the areas where the student struggled in the diagnostic test. Students are able to re-attempt the diagnostic indefinitely until they pass. Once they succeed in the diagnostic, they are deemed to have "mastered" the component and are able to move to the next component.

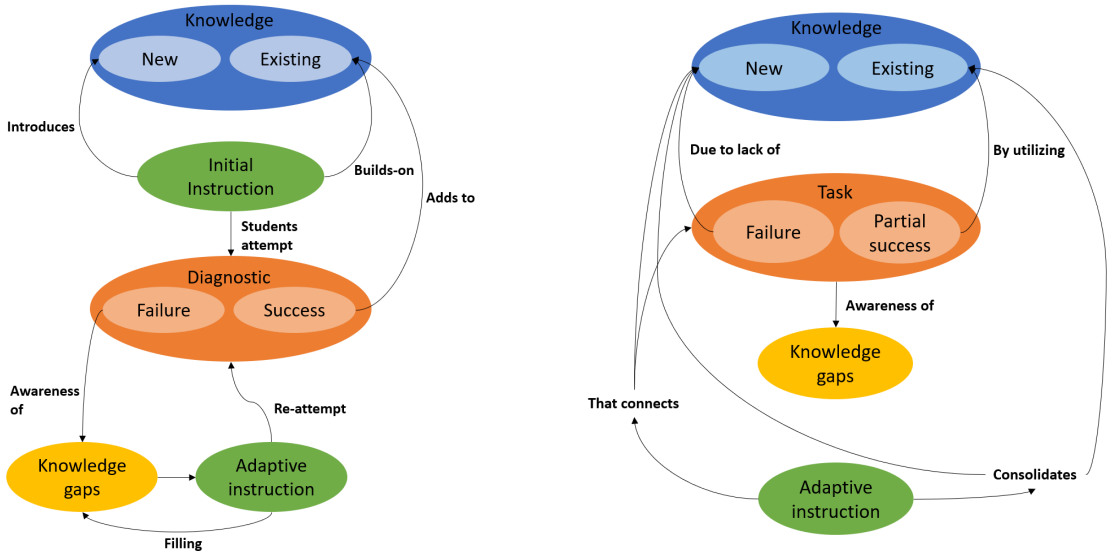

Fig. 3: Concept maps of (a) mastery learning and (b) productive failure

4.2 Productive Failure

Productive failure suggests that students failing, by attempting a task before receiving instruction, can improve student learning. To complete that first task, students require using existing knowledge in tandem with new knowledge. When they attempt the task, the absence of new knowledge encourages students to exhaust their

Table 1: Concepts - Mastery Learning (ML)

Structure: Course content needs to be broken down into subsequent components. Each component taught requires students to have an adequate understanding of the component taught before it.		
Area	**Concept**	**Description**
Knowledge	New knowledge	The new content that is being taught to students.
	Existing knowledge	Knowledge that students are assumed to already possess. This is knowledge acquired in previous components.
	Knowledge gaps	Knowledge linked to observed failure.
Teacher input	Initial instruction	Instruction phase focuses on teaching new content and links it (implicitly or explicitly) with the content taught in the previous component.
	Adaptive instruction	Students are individually given further work that focuses on the areas they struggled with in the diagnostic. This is self-administered, but materials may be selected by the teacher.
Student tasks	Diagnostics	Students are required to pass a diagnostic or assessment task for a component before being able to move onto the next component. If students fail the diagnostic, they are able reattempt the diagnostic an indefinite amount of times until they pass.
Student results	Success	Completion of the diagnostic task is considered to reflect that a student has "mastered" that component. Mastery means that students have adequately understood the concepts taught in a component and they have the required knowledge to move to the next component.
	Failure	Failing a diagnostic stop progression but this is not seen as a negative outcome, but rather an opportunity for students to better learn the component being taught. Students are able to become aware of their knowledge gaps (reflected by the errors made in the diagnostic), and the teachers can adapt their instruction to fill those gaps.

Table 2: Concepts - Productive Failure

Structure: productive failure applies to a single session or lesson linking two concepts. A key aspect of productive failure is that the task phase comes *before* the instruction phase.		
Area	**Concept**	**Description**
Knowledge	New knowledge	The new content that is being taught to students and they expect to acquire.
	Existing knowledge	Knowledge students are assumed to already possess, from previous instruction or experience.
	Knowledge gaps	Conceptual knowledge linked to observed failure. It may indicate missing knowledge (new or existing) or misconceptions
Teacher input	Adaptive instruction	Teachers should connect the partial successes of students to the canonical solution being taught. Thus, instruction is related to the students' results. This is also an opportunity for teachers to fill in any other knowledge gaps identified during the task phase.
Student tasks	Task before instruction	The task given to students covers in some degree the new content being taught. Students attempt the task before receiving instruction so they are expected to make some progress or at least explore some ideas.
Student results	Failure	Most students will fail the task given, due to not possessing the new knowledge required. Failure helps students to become aware of this gap in their knowledge and prepares them to learn the content that maps to it.
	Partial success	Students are expected to only partly succeed in completing the task. A partial success reflects that students have been able to utilize some of their existing knowledge in order to get part of the solution. How close students get to a full solution helps teachers to see what gaps they have in their existing knowledge.

existing knowledge as much as they can to produce partial solutions. Ultimately, they are expected to fail. The failure reveals, to students, gaps in their overall knowledge (the lack of new knowledge) and the partial success reflects that they have utilized their existing knowledge. The level of partial success reveals to the teacher any knowledge gaps students might have in their existing knowledge.

Instruction follows the task and is adaptive to the results of students. A key mechanic, within productive failure, is for the teacher to connect the new knowledge being taught, with the partial solutions students have generated. This connection is critical for two reasons: (1) it is the opportunity for the teacher to go over what existing knowledge students should be using, addressing any gaps students might have and (2) to help students connect their existing knowledge with the new knowledge, consolidating the two. Having students be self-aware of their gaps helps this consolidation.

5 Discussion

The discussion of this report is guided by three the pedagogical concepts listed by Ben-Ari (Ben-Ari, 2001), that support constructive learning. Each section discusses one of the pedagogical concepts and how the concepts of each pedagogical approach exhibit it.

5.1 Active learning to facilitate mental model construction

Both pedagogical approaches are amenable to active learning as we are to explore next.

ML leaves the instructor more room to choose the way the initial instruction is given to students, so it could be passive or active to start with.

After the diagnostic task is completed, adaptive instruction (in the way of teaching materials such as online content, podcast, quizzes etc) is individually selected for each student based on the knowledge gaps identified in that diagnostic. Whilst teachers may select the learning resources, students are expected to go over them by themselves, so it can be considered as a self-learning process. Mastery learning acknowledges that every student progress at their own pace, hence it make sense to be self paced. Teachers support students in that process, but they do not provide traditional active learning activities to the class.

Productive failure promotes active learning by beginning with a task phase that immediately engages students with the learning process, rather than starting with a passive instruction phase. Student engagement continues into the instruction phase. The adaptive instruction given by the teacher in the instruction phase is collaborative, connecting the canonical answer with the solutions of students. In other words, instead of showing them the correct solution or fixing their errors, the teacher guides students through a discussion of the part required to make each students solution complete. The collaborative approach allows to support the diversity of student solutions by exploring them in the class and building from them. This is more scalable compared to one-to-one tuition.

5.2 Utilizing pre-existing knowledge

Most instruction given by teachers builds on previous knowledge that students are *implicitly* assumed to possess. For example, when we teach to manipulate arrays we expect students to have learnt about iteration, because we have covered iteration the previous week. Approaches that support constructive learning, *explicitly* recognize how concepts build on top of previous knowledge and support students to connect the two.

Pre-existing knowledge is embedded into the structure of mastery learning. Fundamentally, mastery learning recognises that students need to learn some concepts before being able to adequately learn others. Moving on to the next related concept without a correct grasp of the current concept can be detrimental to their learning.

Hence, the sequential structure of ML ensures that concepts are learnt in an order that satisfy their required knowledge. ML also utilizes diagnostic tasks to ensure that students have properly learnt a given concept before being able to move on to the next. The sequential structure encourages students to apply what they've just learnt in previous components to the new component. There is also some guarantee they have mastered the required knowledge, although they may have forgotten some.

Note ML does not explicitly link the particular prior knowledge with the concepts currently being taught. This explicit connection may be indicated by the teacher under direct instruction.

In productive failure, the task phase is designed for students to both identify and apply their required prior knowledge to the task. Student should be encouraged to apply prior knowledge in order to come up with naive answers and partial solutions. As students have not been taught yet the additional concept or insight required to complete the task in full, they may identify the knowledge gaps. This approach would only work when students have adopted a growth mindset. More importantly, applying their prior knowledge to the task before receiving instruction helps students to discover how their existing knowledge fits with the new concepts being taught.

This connection between prior and new knowledge is reinforced when the teacher connect the partial solutions of students to the canonical answer.

The collaborative phase allows peers to share the ways they applied prior knowledge. As teachers connect the canonical answers to the partial solutions of students, they can highlight examples of students correctly applying their existing knowledge. Some students may have a sound knowledge of a prior concept but they were not aware it was relevant to the task. This is what Perkins called *inert* knowledge (Perkins, Martin, & Educational Technology Center, 1985). By looking at peers' examples of using that knowledge, it stops being inert as it gets grounded into the canonical answer.

This process helps to ensure that students correctly construct knowledge by connecting the two, as opposed to constructing an incorrect knowledge model. By highlighting what was correct about student solutions, or alternatively demonstrating how students should have applied their prior knowledge, the teacher can address any problems students have. In other words, this phase may identify novices' misconceptions that can be cleared to facilitate learning.

5.3 Utilizing failure

Failure is many times seems as a negative outcome because it reduces motivation. Thus, it is common to provide scaffold activities which minimize the chance of failure. However, Bjork remind us that "learning is different to performance" (Bjork & Bjork, 2011), and that in reducing failure we may not increase learning.

Failure is not always negative provided we can recover and learn from it. In constructivism, failure is seeing as an opportunity. In both ML and PF failure is utilized to reveal knowledge gaps in students, which are then used to guide adaptive instruction or are utilized to help students construct new knowledge. Regardless of whether a student succeeds or fails a task, traditional pedagogical approaches simply moves all students on to the next topic. Mastery learning doesn't expect that all students will immediately "master" each concept, and plans for it accordingly. Students failure is provides the teacher with data that indicates any misconceptions or knowledge gaps that students have. Once identified, the appropriate adaptive instruction can be provided to students to strengthen their learning.

Utilizing the failure of students is central to productive failure. Students are intentionally given a task which they are expected to fail (or at least to only make partial progress). The failure helps students to self-identify gaps in their overall knowledge, these gaps reflecting the new content that they need to learn. Teachers might already aware of these gaps, but enabling students to become aware helps them to construct conceptual knowledge. Awareness of knowledge gaps helps students to realise a need to learn new concepts, where the new concepts connect with their existing knowledge and prepares them to learn the new concepts. The extent to which students fail the initial task, and how thorough their partial solutions are, can reveal to teachers what gaps *this* group of students have in their existing knowledge. Teachers should supplement those gaps in the subsequent instruction. Teachers are further able to reinforce the connections by explicitly linking the partial solutions of students (reflecting that previous knowledge) to the canonical answer. Thus students can see students how their existing knowledge connects to the new concepts being taught.

In short, productive failure front-loads failure to happen early on in the class room, so that it can be utilized to focus the instruction and make it relevant. Experienced teachers will try to cover knowledge gaps they have seen in previous cohorts. Front-loading failure will aid teachers to better manage the gaps students have in their prior knowledge.

Other approaches are also using errors for students to learn. For example, Griffin has provided code with carefully designed bugs as a learning task (Griffin, 2019); this recent study shows that student can learn as much from debugging that code as from writing similar code from scratch. Ginat used task that contained key conceptual errors to teach OOP concepts such as the role of constructs and the scope of variables(Ginat & Shmalo, 2013). He followed constructivist principles and implement a collaborative approach in the classroom to discuss each error. In contrast to those strategies, in either ML or PF, the errors are produced by the students instead of inserted by the teacher. Thus, they support better the constructivist principles by forcing students to examine and correct their own faulty mental models. Additionally, the collaborative phase of PF allows students to share their mistakes and for the teacher to explain how to recover from a range of mistakes. This can be useful for students to be aware of potential pitfalls in similar problems.

6 Conclusion

We are looking for new ways for students to learn, particularly in an era of information overload, in which online content is ubiquitous and in the view of naive students any problem is only one google query away from being resolved. Undergraduate learning has moved in the last 10 years from being passive to becoming active with many new approaches tried and evaluated. In this context, the role of the teacher is to help students not to memorise or locate that content but to develop a coherent mental structure that facilitates its application.

From the constructivist perspective, active learning is enhanced by both building on prior knowledge and using failure as an opportunity to learn. In this study we have reviewed two approaches that support this view. Both pedagogical approaches relied on prior knowledge and use failure as an opportunity to learn. Mastery learning supports constructive learning by sequencing concepts, forcing mastery before moving to the next one, and

providing additional learning resources to help reaching that mastery level at their own pace. As sequential and self-paced progress is a core concept of mastery learning, the approach can't be applied to a single session/topic. Instead, mastery learning is suited for structuring a course, or teaching a series of connected topics.

Productive failure utilizes failure to help students connect their existing knowledge with the concepts being taught. The approach can be applied to an individual module and is especially applicable to teaching concepts which are interrelated as is the case with programming constructs. Productive failure key strength is to help students connect their existing knowledge with the new concepts being taught. Specifically, productive failure is best suited towards single sessions that introduce concepts which build on from previous instruction. In contrast, the approach would be less effective when teaching a new concept which has weak links to previous concepts. For students that have a fixed mindset, PF may be challenging to start with, and using this approach multiple times will help them to understand the benefits of desirable difficulties (Bjork & Bjork, 2011) and increase their resilience. Note that we could embed a PF session inside a ML module, for example to replace the initial instruction, so the two approaches could be used in combination.

The analysis and creation of concept maps and associated tables for each approach has highlighted the concepts that drive mastery learning and productive failure. Being aware of the key concepts of each approach, and the contexts that they can be used in, can help CS instructors apply constructivism to improve learning at CS1 level. Both ML and PF provide strategies to scale constructive learning from one-to-one or pairs to small and medium class sizes. It is a further challenge to make these approaches scale to large classes. Further real-life experience and analysis is required for both ML and PF to explore mechanisms that could facilitate their usage under different teaching environments.

There are two open lines of research from this study: (1) extend the use of productive failure to our CS0/CS1 with emphasis on implementing and providing guidelines for its collaborative approach, (2) use the concepts highlighted in this work to understand in depth how each pedagogy helps students construct knowledge.

References

Anderson, N., & Gegg-Harrison, T. (2013). Learning computer science in the "comfort zone of proximal development". In *Proceeding of the 44th acm technical symposium on computer science education* (pp. 495–500). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2445196.2445344` doi: 10.1145/2445196.2445344

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, *20*(1), 45–73. Retrieved from `https://www.learntechlib.org/p/8505`

Bjork, E., & Bjork, R. (2011, 01). Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning. *Psychology and the Real World: Essays Illustrating Fundamental Contributions to Society*, *2*, 56-64.

Brooks, J. G., & Brooks, M. G. (1993). *In search of understanding: The case for constructivist classrooms*. Alexandria: Association for Supervision and Curriculum Development.

Caceffo, R., Wolfman, S., Booth, K. S., & Azevedo, R. (2016). Developing a computer science concept inventory for introductory programming. In *Proceedings of the 47th acm technical symposium on computing science education* (pp. 364–369). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2839509.2844559` doi: 10.1145/2839509.2844559

Ginat, D., & Shmalo, R. (2013). Constructive use of errors in teaching cs1. In *Proceeding of the 44th acm technical symposium on computer science education* (pp. 353–358). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2445196.2445300` doi: 10.1145/2445196.2445300

Gorp, M. J. V., & Grissom, S. (2001). An empirical evaluation of using constructive classroom activities to teach introductory programming. *Computer Science Education*, *11*(3), 247-260. Retrieved from `https://doi.org/10.1076/csed.11.3.247.3837` doi: 10.1076/csed.11.3.247.3837

Greening, T., Kay, J., Kingston, J. H., & Crawford, K. (1996). Problem-based learning of first year computer science. In *Proceedings of the 1st australasian conference on computer science education* (pp. 13–18). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/369585.369588` doi: 10.1145/369585.369588

Griffin, J. M. (2019). Designing intentional bugs for learning. In *Proceedings of the 1st uk & ireland computing education research conference* (pp. 5:1–5:7). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/3351287.3351289 doi: 10.1145/3351287.3351289

Hoda, R., & Andreae, P. (2014). It's not them, it's us! why computer science fails to impress many first years. In *Proceedings of the sixteenth australasian computing education conference - volume 148* (pp. 159–162). Darlinghurst, Australia, Australia: Australian Computer Society, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2667490.2667509

Jonassen, D. H. (1991, Sep 01). Objectivism versus constructivism: Do we need a new philosophical paradigm? *Educational Technology Research and Development*, *39*(3), 5–14. Retrieved from https://doi.org/10.1007/BF02296434 doi: 10.1007/BF02296434

Kaczmarczyk, L. C., Petrick, E. R., East, J. P., & Herman, G. L. (2010). Identifying student misconceptions of programming. In *Proceedings of the 41st acm technical symposium on computer science education* (pp. 107–111). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1734263.1734299 doi: 10.1145/1734263.1734299

Kapur, M. (2008). Productive failure. *Cognition and Instruction*, *26*(3), 379-424. Retrieved from https://doi.org/10.1080/07370000802212669 doi: 10.1080/07370000802212669

Kapur, M. (2017, 11). Examining the preparatory effects of problem generation and solution generation on learning from instruction. *Instructional Science*, *46*. doi: 10.1007/s11251-017-9435-z

Kapur, M., & Bielaczyc, K. (2012). Designing for productive failure. *Journal of the Learning Sciences*, *21*(1), 45-83. Retrieved from https://doi.org/10.1080/10508406.2011.591717 doi: 10.1080/10508406.2011.591717

Lehtinen, T., Lukkarinen, A., & Haaranen, L. (2021). Students struggle to explain their own program code. In *Proceedings of the 26th acm conference on innovation and technology in computer science education v. 1* (p. 206–212). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3430665.3456322 doi: 10.1145/3430665.3456322

Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: Novice programmers and the solo taxonomy. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 118–122). New York, USA: ACM. doi: 10.1145/1140124.1140157

Loibl, K., Roll, I., & Rummel, N. (2016, 07). Towards a theory of when and how problem solving followed by instruction supports learning. *Educational Psychology Review*, *29*(4), 693-715. doi: 10.1007/s10648-016-9379-x

McCane, B., Ott, C., Meek, N., & Robins, A. (2017). Mastery learning in introductory programming. In *Proceedings of the nineteenth australasian computing education conference* (pp. 1–10). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/3013499.3013501 doi: 10.1145/3013499.3013501

Mercer, D. (1986). Mastery learning. *British Journal of In-Service Education*, *12*(2), 115-118. Retrieved from https://doi.org/10.1080/0305763860120212 doi: 10.1080/0305763860120212

Michaelsen, N. M. C. H., Larry K.; Davidson. (2014). Team-based learning practices and principles in comparison with cooperative learning and problem-based learning. *Journal on Excellence in College Teaching*, *25*(3&4), 57-84.

Moreno, A., Sutinen, E., & Joy, M. (2014). Defining and evaluating conflictive animations for programming education: The case of jeliot conan. In *Proceedings of the 45th acm technical symposium on computer science education* (pp. 629–634). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2538862.2538888 doi: 10.1145/2538862.2538888

Novak, J. D., & Cañas, A. J. (2006). *The theory underlying concept maps and how to con-*

struct and use them (research report No. 2006-01 Rev 2008-01). Florida Institute for Human and Machine Cognition. Retrieved from http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.htm

O'Rourke, E., Haimovitz, K., Ballweber, C., Dweck, C., & Popović, Z. (2014). Brain points: A growth mindset incentive structure boosts persistence in an educational game. In *Proceedings of the 32nd annual acm conference on human factors in computing systems* (pp. 3339–3348). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2556288.2557157 doi: 10.1145/2556288.2557157

Perkins, D., Martin, F., & Educational Technology Center, M., Cambridge. (1985). *Fragile knowledge and neglected strategies in novice programmers. ir85-22 [microform] / david perkins and fay martin* [Book, Microform, Online]. Distributed by ERIC Clearinghouse [Washington, D.C.]. Retrieved from http://www.eric.ed.gov/contentdelivery/servlet/ERICServlet?accno=ED295618

Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2017). K-8 learning trajectories derived from research literature: Sequence, repetition, conditionals. In *Proceedings of the 2017 acm conference on international computing education research* (pp. 182–190). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/3105726.3106166 doi: 10.1145/3105726.3106166

Robins, A. (2010, 04). Learning edge momentum: A new account of outcomes in cs1. *Computer Science Education*, *20*, 37-71. doi: 10.1080/08993401003612167

VanLehn, K., Siler, S., Murray, C., Yamauchi, T., & Baggett, W. B. (2003). Why do only some events cause learning during human tutoring? *Cognition and Instruction*, *21*(3), 209-249. Retrieved from https://doi.org/10.1207/S1532690XCI2103_01 doi: 10.1207/S1532690XCI2103_01

Waite-Stupiansky, S. (1995). The road to constructivist classrooms. *The Educational Forum*, *59*(1), 98-100. Retrieved from https://doi.org/10.1080/00131729409336369 doi: 10.1080/00131729409336369

Wass, R., & Golding, C. (2014). Sharpening a tool for teaching: the zone of proximal development. *Teaching in Higher Education*, *19*(6), 671-684. Retrieved from https://doi.org/10.1080/13562517.2014.901958 doi: 10.1080/13562517.2014.901958

Watson, C., & Li, F. W. (2014). Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on innovation & technology in computer science education* (pp. 39–44). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2591708.2591749 doi: 10.1145/2591708.2591749

Westermann, K., & Rummel, N. (2012, 07). Delaying instruction: Evidence from a study in a university relearning setting. *Instructional Science*, *40*. doi: 10.1007/s11251-012-9207-8

Zimmerman, B., & K. Dibenedetto, M. (2008, 03). Mastery learning and assessment: Implications for students and teachers in an era of high-stakes testing. *Psychology in the Schools*, *45*, 206 - 216. doi: 10.1002/pits.20291

Zingaro, D., & Porter, L. (2014). Peer instruction in computing: The value of instructor intervention. *Computers & Education*, *71*, 87 - 96. Retrieved from http://www.sciencedirect.com/science/article/pii/S0360131513002777 doi: https://doi.org/10.1016/j.compedu.2013.09.015