Models of Mastery Learning for Computing Education

Claudia Szabo
The University of Adelaide
Adelaide, Australia
claudia.szabo@adelaide.edu.au

Miranda C. Parker
mcparker@sdsu.edu
San Diego State University
San Diego, CA, USA

Michelle Friend
mefriend@unomaha.edu
University of Nebraska Omaha
Omaha, NE, USA

Johan Jeuring
j.t.jeuring@uu.nl
Utrecht University
Utrecht, The Netherlands

Tobias Kohn
kohnt@tobiaskohn.ch
Karlsruhe Institute of Technology
Karlsruhe, Germany

Lauri Malmi
Lauri.Malmi@aalto.fi
Aalto University
Espoo, Finland

Judithe Sheard
judy.sheard@monash.edu
Monash University
Clayton, Australia

Abstract

The application of mastery learning, where students progress through their learning in a self-paced manner until they have mastered specific concepts, is considered appealing for teaching introductory programming courses. Despite its growing popularity in computing and its extensive use in other disciplines, there is no overview of the design of courses that use mastery learning. In this position paper, we present an overview of five mastery learning models and discuss examples of how these can be applied in practice, both in foundational programming as well as more advanced courses. Our analysis focuses on the student progression through the course, the assessment structure, and the support for self-paced learning, including for struggling students. This work provides a greater understanding of mastery learning and its application in a computing education context.

CCS Concepts

• **Social and professional topics** → **Computing education**; **Computer science education**.

Keywords

mastery learning, models of instruction, competency-based learning

ACM Reference Format:
Claudia Szabo, Miranda C. Parker, Michelle Friend, Johan Jeuring, Tobias Kohn, Lauri Malmi, and Judithe Sheard. 2025. Models of Mastery Learning for Computing Education. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3641554.3701868

SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0531-1/25/02
https://doi.org/10.1145/3641554.3701868

1 Introduction

Mastery learning is an approach to teaching where students progress through a curriculum in a self-paced manner, demonstrating competency in specific, usually ordered topics [15]. Mastery learning can help to ensure that foundational concepts are well understood and applied before other concepts are introduced or attempted [12]. The application of a mastery learning pedagogy within a course as well as across a series of courses can facilitate scaffolded, gradual, and self-paced progression through a curriculum, making it a valuable approach for teaching. Mastery learning has been extensively applied in areas such as healthcare [7, 13], legal [10], and English literature [5], and has seen recent popularity in computing education [1, 12, 24], in particular for teaching introductory programming courses (i.e., CS1 and CS2).

Mastery learning establishes a direct correspondence between assessment and individual learning outcomes or thematic units of the curriculum. The broad brush approach of assessing student performance at the end of a course, which has been the traditional way of assessing in computing education, gives way to fine-grained control of which topics, concepts, or learning objectives are effectively mastered by the student. By moving the focus from performance to mastery, the learning of individual students is permitted to progress at different paces, i.e., students are usually given the time needed to learn and master a unit rather than evaluating how much they were able to learn within a fixed time frame. Pedagogically, mastery learning is based on the belief that every student can master a subject when given enough time [3].

Computing education is a prime target for the application of mastery learning due to the inherent complexity and (inter)dependence of concepts [26]. CS1/CS2 courses often have a wide distribution, with some students performing very well and others failing completely [8], with a large number of possible explanations for the cause of this distribution [26]. One explanation given is that the strong dependence between concepts can cause students with an insufficient grasp of earlier concepts to fail when faced with the complexity of more advanced concepts and ideas [26]. For example, someone who struggles with the basic concepts of variables and looping would be hopelessly lost when facing nested loops with dependencies between the inner and outer loop. Mastery learning could provide a possible way to deal with this issue through its emphasis on mastering earlier concepts before building on them. It can also help to make dependencies between core concepts more visible to students, encouraging a deeper understanding of concepts before progressing.

When designing a mastery learning approach, it is important to consider that the order of concepts and/or curriculum items might not be straightforward to define. For example, in a CS1/CS2 setting, where concepts are tightly coupled [26], it is possible to define a strict dependency order between specific topics. However, this strict dependency might not hold in higher-level courses, where a collection of topics or principles might only be loosely related. This suggests that mastery learning models need to vary to adapt to different curriculum requirements.

Garner et al. note that, in reviewing the use of mastery learning in computer science, there is no single way that mastery learning is typically implemented [12]. In this position paper, we expand on existing work by constructing and visualising models of mastery learning. In a recent Dagstuhl seminar [9], the authors spent a week refining these models based on existing literature and the practical experience of seminar attendees. Based on this iterative process, we propose a series of templates of various sequences of curricular activities to help practitioners and researchers in implementing mastery learning in their courses. Our models focus on the sequence of curricular activities, the progression of students through the course, assessment structure, and student support. To the best of our knowledge, no such visualisations exist in the literature. This paper seeks to add to the analysis done by Garner et al. and provide templates to compare implementations and guide instructors and researchers interested in trying mastery learning in their classrooms. By rendering explicitly the diversity in approaches to and understandings of mastery learning, we provide the groundwork for comparison of studies of the effectiveness of mastery learning in computing education settings. The contributions of this position paper are twofold:

- A taxonomy and visualisation of possible models of mastery learning.
- An analysis of how these models could be applied in computing courses, both in introductory and more advanced courses.

2 Background and Related Work

In conventional learning settings, time is a constant factor and achievement varies among students within that time. Carroll postulated that learning is a function of the time a student spends learning compared to the time the student needs to learn. Bloom extended this notion to a belief that, if given enough time, almost every student can learn a concept [3]. As a result, mastery learning, or competency-based learning, as conceptualised by Bloom [3], holds achievement constant as much as possible, while varying the time required of each student to reach that level of mastery of the material: all students can learn a concept if given enough time.

Mastery learning hinges on the importance of fully understanding basic concepts before building upon them [11]. However, given the challenges in supporting each student in mastering the material, there are different approaches to mastery learning. Two such approaches include Bloom's Learning for Mastery system [3], which is group-based and teacher-paced, and Keller's Personalized System of Instruction (PSI) [15], which is independent and student-paced. Learning for Mastery is characterised by clear learning objectives, a course that is broken up into smaller, atomic components that can be evaluated periodically, teachers who use those formative assessments to provide feedback to students about their specific issues, and the encouragement of students to work together in small groups to review their results and progress, perhaps using additional alternative learning resources to continue their learning in that unit [21]. Meanwhile, the Personalized System of Instruction has the key features of being self-paced, requiring students to demonstrate mastery of a unit before progressing, using lectures for motivation of progression in the material, emphasis on written materials, and the use of proctors to provide a one-on-one evaluation of students [11, 15]. These two approaches to mastery learning are not entirely in contrast to one another, though they do begin to indicate the different priorities that instructors can have when implementing mastery learning in the classroom.

Depending on the mode of implementation, mastery learning has been found to positively impact student achievement [4], with larger impacts for weaker students [17]. Mastery learning efforts generally elicit positive reactions from students, though they may reduce course completion rates [17]. Further, an analysis of 17 meta-analyses on mastery learning indicates an overall effect size of 0.67, which is interpreted as having the potential to considerably accelerate student achievement [14].

Despite the benefits of mastery learning, there are real logistical challenges that arise when trying to apply the philosophy of every student being able to achieve if given enough time. One challenge is quantifying what it means to achieve within the context of the learning goals of the course. Scholars have debated as long as mastery learning has existed about how much a student should be expected to learn before the law of diminishing returns starts to apply [2]. Further, there are design decisions to be made between whether mastery is defined as a percentage of units learned or the percentage learned within each unit, and how this all fits within modern grading systems [29]. And, these metrics all depend on how the learning goals are composed for the course, and whether they define a level of mastery within them [20]. There are also concerns around time, given the lack of holding time constant in a mastery learning classroom. As a result, some students may learn the material quickly, but others may require significant time to master the same material. This also represents a time investment on the part of the teacher, who is designing the learning goals, facilitating the assessments and feedback, and creating alternative learning materials for students to further their understanding of the material [28]. There are also institutional constraints that require a grade to be assigned to students by a given date; however, this may not be an adequate amount of time for the students to learn the material.

2.1 Mastery Learning in Computing

As computing education has grown and borrowed from other fields, the learning theories that are implemented within our computing classrooms have evolved in turn. Mastery learning, though, has been shown to be in the top-ten most popular learning theories in computing education [27]. Many of the implementations of mastery learning in computing use the Learning Edge Momentum theory to motivate their design decisions [26]. In these cases, the belief that learning one concept will aid in the learning of closely related concepts feeds easily into a mastery learning model that requires students to demonstrate mastery of a concept before progressing further into the course. There have been efforts to break down introductory computing courses into their atomic concepts [19], which would further aid mastery learning efforts.

A review of computing courses using mastery learning found that these efforts were primarily concentrated in introductory computing courses [12]. Mastery learning in computing has replicated findings of positive reception by students, though issues of scaling, procrastination, and effective design of assessments have also been found [12]. These benefits and risks can be seen in a six-year implementation of mastery learning in a computing classroom [24]. With interactive interventions within the mastery learning model, more students ultimately received an A in the course [24].

3 Models of Mastery Learning in Computing Courses

In the following sections, we describe several different models of how courses can present content and assessment. We first describe how traditional computing courses are typically organised, then present several different models of mastery learning.

In each section, we provide a diagram; a description; how instruction/content is organised; how students progress through the model, particularly in the case of failure; how assessment fits into the model; and how support is provided for struggling students. We also provide an application of the model in the context of computing education. Each model contains a representation of content in the form of a *module*. Each content module is an abstraction of some curricular unit. The "size" of each unit is necessarily left as an exercise for the instructor. Thus, a module may be as small as a simple idea such as programming statements, or may be more complex such as object-oriented programming as a paradigm. We thus emphasise that in this paper, *module* should not be associated with a traditional fixed part of studying in curriculum descriptions and degree requirements, typically using a term such as *course or unit.* In addition, it should be noted that the models presented henceforth can be applied at different levels of course granularity - for a whole course or parts thereof, or for a sequence of courses or programs, as suits the curriculum, instructor, and institution.

3.1 Traditional

The predominant teaching model in computing courses is one or more weekly lectures and lab sessions, the content of which is determined by the lecturer. Learning is assessed by final and potentially mid-term exams, problem sets, quizzes, team-based project work, and/or lab assignments, as shown in Figure 1.

Progression: The speed of progression through the course is set by the teacher.

Assessment: Courses are assessed in various ways. At the authors' current and previous institutions, courses are predominantly assessed through a final assessment at the end of the course, zero, one

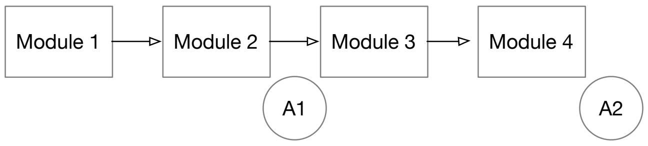

Figure 1: Traditional

or more mid-term assessments on a fixed date during the course, and assessments of lab assignments, often with various deadlines during the course. Usually, a student can take an assessment even if they did not take or failed previous components, or did not go to the classes. If a student fails one or more of the assessment components, there may be possibilities to retake them. Figure 1 shows the flexibility of the approach: the assessments are not connected to the modules, nor to each other.

Support: Students are supported by teaching assistants (TAs) in lab sessions, and can make use of office hours of the teacher. Unless exams and labs are automatically graded, they get (sometimes substantially) delayed feedback on their assessments, in the form of grades or more elaborate feedback.

Application: The ACM Curriculum 2013 [23] contains the description of 84 computing courses taught in the US, Europe, Asia and Australia, at 50 different universities. All course descriptions include a (sometimes brief) description of how students are assessed in the course. There is a wide variety of combinations of assessment components, but almost all courses make use of a mid-term and a final exam, besides problem sets, quizzes, labs, or team-based projects. As far as we could determine, all assessment deadlines were set by the teacher or the institution of the course.

3.2 Instruction Locked (Keller Plan / PSI)

Keller [15] proposed a model of mastery learning that emphasised self-paced learning and unit mastery ("unit-perfection requirement for advance" in Keller's words), which became known as 'Keller plan' and 'Personalised System of Instruction (PSI)'. Students are provided with the learning materials for one unit or module of the curriculum at a time and have to demonstrate mastery of this module before they are given access to the next module (Figure 2). As a consequence of this model, learning materials and instructions must be provided in "written" (i.e. time-independent) rather than "oral" (i.e. fixed time) form, although technology has clearly blurred these lines.

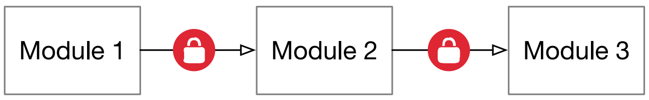

Figure 2: Instruction Locked

Progression: In the "instruction locked" model students are required to show mastery of a module before being allowed to progress to the next unit. This strict barrier for each individual student is entirely at odds with traditional course delivery structures and thus often renders its implementation problematic at scale.

Assessment: Students undergo assessment of module mastery at virtually any time, demonstrating mastery of the module according

to the learning objectives. In case of not passing the assessment, students return to the module and retake the assessment as often as needed to pass.
Support: A corner stone of Keller's model are the proctors and instructors who, by assessing and watching over unit mastery, give immediate feedback to students on elements of the assessment they have failed, leading to "almost unavoidable tutoring" [15].
Application: The "instruction locked" model has repeatedly found application in computing education [16, 18, 25]. Purao et al. [25] report on a multi-year evolution of their implementation of the PSI for an introductory programming course. Modifications to the original model by Keller were necessary to counteract procrastination and help students find a good pace with which to progress. Simply setting a minimal requirement for passing the course led to students orienting themselves towards that minimum with few students going beyond. Proposing different "tracks" (paces at which to progress) with clear indication what grade each of these tracks will most likely lead to had the best success in guiding students.

3.3 Assessment Locked

The "assessment locked" model shown in Figure 3 embodies the progression aspect of the mastery learning approach, where the curriculum is split into a series of modules, each with an associated assessment. The sequence of assessment items follows a dependency structure through a set timeline, and students progress through each test in order, moving on to the next test once they have shown competency in the previous one.

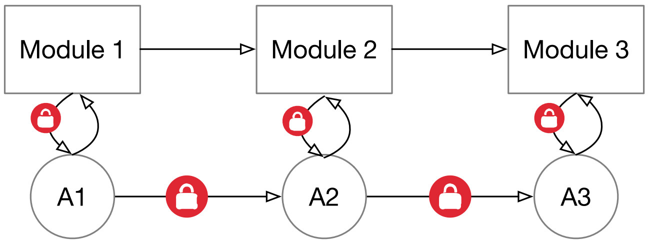

Figure 3: Assessment Locked

Progression: In this model the learner progresses through the content modules sequentially, taking assessment items only after the module has been completed. Students can progress to learning content from subsequent modules even if they have not passed the assessment items.
Assessment: Each module has an associated assessment that the learner takes upon the completion of the module. The modules are delivered according to the course timeline, which implies that the assessments might also be run during specific set time periods. There is a dependency connection between assessments, in that they have to be taken in order. If students fail an assessment, they return to learning the associated content module.
Support: Formative assessment supports students in this model by providing a series of practice exercises in support for each module and in preparation for each assessment item. Support for struggling students could come in the form of supporting them to retake past assessments, in a pipeline manner. This would imply both the implementation of tests from past assessments, but also support sessions through the in-between period. For example, A1 tests could be scheduled at the same time as A2 tests, and A1 and A2 tests could be scheduled at the same time as A3 tests, allowing for support sessions for students who need them.
Application: An example of the assessment locked model can be found in Ott et al.'s [24] implementation of mastery learning, where students advance through a series of 'progression level' assessments. Once students achieve passing marks in a progression-level test, a series of other progression levels are unlocked, and students can proceed to taking specific tests. Students can attempt Progression Levels at any point, however, to increase engagement they are required to attempt at least one PL level per week.

In one of the authors' implementations of mastery learning at their institution, the foundations programming course is split into four three-week modules, and competency in each module is demonstrated by passing a test at the end of each module. Students who fail one module test have to undertake it again before progressing to subsequent tests. Assessments for all modules are given at three-week intervals. Students can participate in instruction and practice for the subsequent modules, as well as specific support sessions relevant to the modules they have failed to show competency in.

3.4 Unlocked

The "unlocked" model strongly embodies the self-paced aspect of mastery learning. The most prominent aspect of this model is that students may progress through assessments at any speed. Unlike the "assessment locked" model, students are not required to engage with learning materials before taking assessments (see Figure 4). Assessments may be ordered or unordered. That is, students may be required to take assessments in a pre-determined order or they may be allowed to take assessments in any order they choose without needing mastery in a prior assessment to unlock the current one.

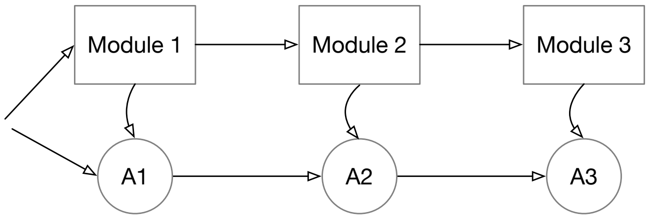

Figure 4: Unlocked

Progression: In courses adopting this approach, learning materials would be provided, particularly as a form of support for students who have not yet mastered content. These materials may be provided asynchronously, in a fully self-paced course. Alternatively, learning materials such as lectures may be provided at pre-determined intervals. Assessment is completely independent of the delivery of content, and students may choose to engage with some learning materials and not others.
Assessment: An assumption of the "unlocked" model is that content is still modularised. While a student may be able to pass all the assessments at the beginning of a term, the model is not one of a comprehensive exam which is pass/fail. Instead, a student would

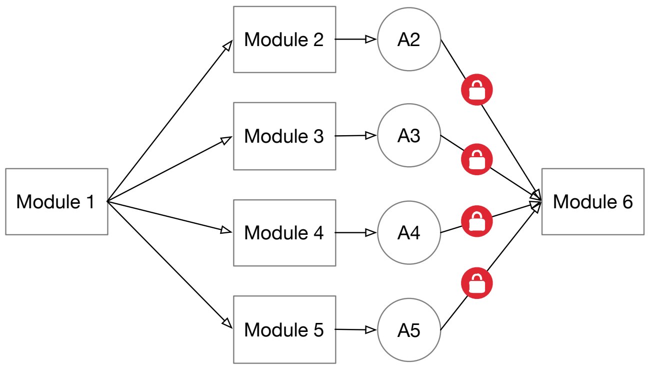

Figure 5: Parallel Locked

be able to demonstrate mastery of each concept individually and revisit any that they had not mastered. For example, a student might pass assessments on statements, variables, conditionals, and lists at the beginning of a CS1 course, but fail an assessment of loops. The student can then work on learning loops before retaking a loops assessment again, possibly repeatedly, until mastery is demonstrated. In a software development course, a student may quickly demonstrate mastery of software design and construction but require more time to practice testing. Because software development courses are generally project-based and collaborative, it may be that teams of students work together to demonstrate mastery of a skill, or that students are assessed individually, possibly concurrent to working on a group project.
Support: Support for students, particularly struggling students, is assumed but is not a strong feature of this model. Thus, student support may be implemented in different ways. For example, all support may take place as just-in-time individual tutoring (either with an instructor, with a generative AI tutor, or with static learning materials such as book sections or explanatory videos). A course may provide a series of lectures or labs that reinforce certain topics, particularly if they are topics that students struggle to master.
Application: This model lends itself to settings with varying levels of prerequisite knowledge that students bring into the classroom. As a result, this model is well-suited for online learning environments such as Massive Open Online Courses (MOOCs), where large numbers of students are enrolled. This model is implicitly described in [22]. With the use of this model, the instructor can relatively quickly understand where the cohort is in terms of prior knowledge.

3.5 Parallel Locked

The "parallel locked" model addresses cases where there is no obvious ordering in which concepts should be learned. A student can learn two or more concepts separately in any order, but it is not possible to proceed further until all branches have been successfully passed.
Progression: Especially in CS1, the basic concepts are tightly integrated implying that it is difficult to separate them totally to support learning them one after another. [26]. For example, basic syntax, variables, assignments and sequential execution are all needed to create even a simple working program. To be able to proceed with learning conditional statements and loops, one needs to understand relational operators and logical expressions. However, it is not clear in which order conditional statements and simple loops should be learned, i.e., either one could be learned first. On the other hand, both are needed, as well as assignments and sequential execution before one can start learning any non-trivial algorithms. Typically learning about functions, objects, and classes takes place after the former basics are mastered, as these more advanced concepts are not needed for their understanding.
Assessment: Each branch has its own assessment module. The student can proceed further only when assessments in *all branches* in the model have been successfully passed. This needs to be checked holistically.
Support: Each branch in the model can have its own supporting learning resources, which could be examples, worked examples, and formative assessments. The logic of the whole model should be explained, i.e., students need to understand that the covered topics are all necessary to learn before one can proceed to further, more complex topics.
Application: In CS1, several basic concepts could be learned using this model, as explained above. In CS2 and intermediate courses, for example, when learning about data structures and algorithms, there is no obvious ordering for learning stacks, queues, and linked lists. But they are relevant prerequisites for graph algorithms. One could learn tree structures and hashing methods in any order, but both are relevant in further studies on algorithms.

3.6 Spiral

The "spiral" model shown in Figure 6 is the most complex of all. In this model the learning modules are ordered to reflect a progression of learning; however, the modules may contain topics that revisit and build on previously introduced topics. This model has flexibility in the assessment whereby learners may demonstrate mastery of a topic in a module through assessment in future modules.

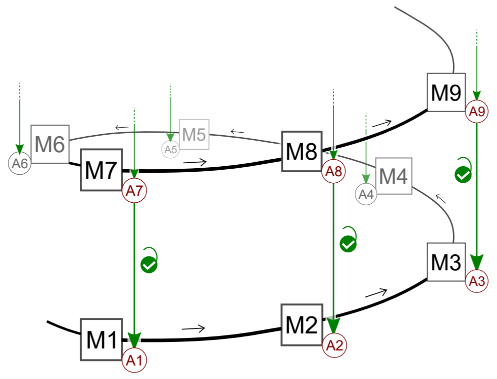

Figure 6: Spiral

Progression: In this model, the learner progresses through modules sequentially. However, in contrast to the other models, the modules spiral back to build on previously covered concepts. For example, in Java one module may introduce the concept of loops with a simple while loop construct. A later module may introduce variations on this with a for loop and introduce the concept of post-test loops.

Table 1: Mastery Learning Models: Summary of key features

Model	Modules	Assessment
Instruction locked	ordered	completed before progressing to next module
Assessment locked	ordered	completed before progressing to next assessment
Unlocked	unordered	completed at any time
Parallel locked	partially unordered	parallel modules completed at any time but all must be completed before progressing
Spiral	ordered with topics revisited in later modules	assessment after each module with possible reassessment at a later module
Traditional (non-mastery)	ordered	assessments not connected to specific modules or each other

A later module may cover loops specific to data structures, for example *for...each* and the *Iterator* object.
Assessment: Each module will have an associated assessment that the learner may take on completion of the module. However, there is some flexibility in the way the learner can demonstrate mastery of the topics through the assessment. If a module has built on a concept covered previously, then passing the assessment associated with this module demonstrates mastery of the current module and there is an implicit assumption that the learner has demonstrated mastery of the topics in the previous module. For example, if module 2 introduces loops and module 8 covers more advanced loops then if a student passes the assessment for module 8 then they do not need to pass the assessment for module 2.
*Support:*Associated resources for each module may be accessed to support their development of mastery in the topic.
Application: In a typical CS1 course there are many concepts to introduce each building on others and this can be overwhelming for novice. This model revisits topics throughout a course and allows the student to develop a deep understanding. While Lister [18] hints at using a spiral curriculum as well as mastery learning (in the form of a Keller plan), he does not really follow this model. Still, it might be a possible example for how a spiral curriculum could look.

4 Discussion

In this paper, we present several models of mastery learning, which are summarised in Table 1. These models serve to demonstrate key elements of mastery learning and the variety of ways it can be implemented. It is important to highlight that these models can be adapted, changed, and combined, providing a rich array of possibilities for a mastery learning course. In addition, a course may adopt some principles of mastery learning without being entirely centred around this approach. For example, a course may use a mastery learning approach for formative assessment, allowing students to revisit assignments to master each topic, but also present one or more summative exams that cannot be re-taken [24]. In addition, the scope of implementation may vary as well, with mastery learning being implemented within a weekly learning activity, during a semester long course, or even at a degree program level.

Many classroom formats and instructional approaches lend themselves well to a mastery learning pedagogy, and its application as discussed above is not restrictive, nor is the type of assessment used prescriptive. Kirchgraber et al. [16], who applied mastery learning for teaching mathematics and computing at high school, stress that mastery learning requires more effort from the students and should therefore be used judiciously for particularly difficult topics or where the student population exhibits large heterogeneity.

The application of mastery learning faces a number of challenges, in particular: (i) students tend to procrastinate in a self-paced learning environment, (ii) the high number of assessments and feedback required strains resources, (iii) the self-paced learning aspect makes it difficult to adopt it in the usual fixed-time structures of our educational institutions. The actual implementation of mastery learning in any specific setting will therefore vary greatly, address the challenges in different ways and end up as a mixture of the models presented in Section 3. For instance, Purao et al. [25] point out that "[Keller's PSI] is an ideal that requires adaptation and evolution that result from nuances of implementation". Moreover, they state that all their adaptations can ultimately be understood as addressing the issue of procrastination and guiding students in terms of the recommended pace of learning. Lastly, another challenge in implementing mastery learning relates to the tension of implementing mastery learning goals in the university reality, where unlimited time and support for students are not available due to external pressures. We believe that the visualisation and discussion of the above models of mastery learning will facilitate their full, partial, or combined implementation within a university context.

5 Conclusion

We present in this position paper a number of models of mastery learning, highlighting their adaptability and the diverse ways they can be implemented in a computing education setting. The aim of this collection of mastery learning models, unique to the literature to the best of our knowledge, is to provide practitioners and researchers alike with a framework for the potential implementation and analysis of mastery learning. The models we presented are not rigid frameworks but provide flexible guidelines that can be tailored to fit specific educational contexts. The potential benefits of mastery learning for computing education are significant, however the practical challenges such as resource constraints, the need to define the computing curriculum in a modular way, the need for extensive assessment items to ensure practice need to be considered. We believe that the analysis and visualisation of mastery learning models presented in this paper will allow educators to better explore these challenges to ensure that learners reap the extensive benefits of mastery learning.

Acknowledgments

This work was derived from Dagstuhl Seminar 24272 - A Game of Shadows: Effective Mastery Learning in the Age of Ubiquitous AI. We thank our collaborators and the Schloss Dagstuhl.

References

[1] Claudio Alvarez, Maira Marques Samary, and Alyssa Friend Wise. 2024. Modularization for mastery learning in CS1: a 4-year action research study. *Journal of Computing in Higher Education* 36, 2 (2024), 546–589.

[2] James Harold Block. 1970. *The effects of various levels of performance on selected cognitive, affective, and time variables*. Ph. D. Dissertation. The University of Chicago.

[3] Benjamin S Bloom. 1968. Learning for Mastery. Instruction and Curriculum. Regional Education Laboratory for the Carolinas and Virginia, Topical Papers and Reprints, Number 1. *Evaluation comment* 1, 2 (1968), n2.

[4] Benjamin S Bloom. 1984. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher* 13, 6 (1984), 4–16.

[5] Matthew Bloom. 2019. Assessing the Impact of" Open Pedagogy" on Student Skills Mastery in First-Year Composition. *Open Praxis* 11, 4 (2019), 343–353.

[6] John B Carroll. 1963. A model of school learning. *Teachers college record* 64, 8 (1963), 1–9.

[7] David A Cook, Ryan Brydges, Benjamin Zendejas, Stanley J Hamstra, and Rose Hatala. 2013. Mastery learning for health professionals using technology-enhanced simulation: a systematic review and meta-analysis. *Academic medicine* 88, 8 (2013), 1178–1186.

[8] Saeed Dehnadi, Richard Bornat, et al. 2006. The camel has two humps (working title). *Middlesex University, UK* (2006), 1–21.

[9] Nick Falkner, Juho Leinonen, Miranda Parker, Andrew Petersen, and Claudia Szabo. 2024. A Game of Shadows: Effective Mastery Learning in the Age of Ubiquitous AI. https://www.dagstuhl.de/en/seminars/seminar-calendar/seminar-details/24272.

[10] Jay M Feinman and Marc Feldman. 1985. Achieving Excellence: Mastery Learning in Legal Education. *Journal of Legal Education* 35, 4 (1985), 528–551.

[11] Eric J Fox. 2004. The personalized system of instruction: A flexible and effective approach to mastery learning. In *Evidence-based educational methods*. Elsevier, 201–221.

[12] James Garner, Paul Denny, and Andrew Luxton-Reilly. 2019. Mastery learning in computer science education. In *Proceedings of the Twenty-First Australasian Computing Education Conference*. 37–46.

[13] Laura Gonzalez and Suzan Kardong-Edgren. 2017. Deliberate practice for mastery learning in nursing. *Clinical Simulation in Nursing* 13, 1 (2017), 10–14.

[14] John Hattie and Gregory CR Yates. 2013. *Visible learning and the science of how we learn*. Routledge.

[15] Fred S Keller. 1968. Good-bye, teacher... *Journal of applied behavior analysis* 1, 1 (1968), 79.

[16] Urs Kirchgraber, Werner Hartmann, Marco Bettinaglio, Moritz Adelmeyer, Jean-Paul David, and Karl Frey. 1997. Und dann und wann ein Leitprogramm! *ZDM* 29 (1997), 199–206.

[17] Chen-Lin C Kulik, James A Kulik, and Robert L Bangert-Drowns. 1990. Effectiveness of mastery learning programs: A meta-analysis. *Review of educational research* 60, 2 (1990), 265–299.

[18] Raymond Lister. 2020. On the cognitive development of the novice programmer: and the development of a computing education researcher. In *Proceedings of the 9th computer science education research conference*. 1–15.

[19] Andrew Luxton-Reilly, Brett A Becker, Yingjun Cao, Roger McDermott, Claudio Mirolo, Andreas Mühling, Andrew Petersen, Kate Sanders, Simon, and Jacqueline Whalley. 2018. Developing assessments to determine mastery of programming fundamentals. In *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*. 47–69.

[20] Robert Frank Mager and Nan Peatt. 1962. *Preparing instructional objectives*. Vol. 62. Fearon Publishers Palo Alto, California.

[21] John D McNeil. 1969. Forces Influencing Curriculum. *Review of Educational Research* 39, 3 (1969), 293–318.

[22] Jeff Offutt, Paul Ammann, Kinga Dobolyi, Chris Kauffmann, Jaime Lester, Upsorn Praphamontripong, Huzefa Rangwala, Sanjeev Setia, Pearl Wang, and Liz White. 2017. A Novel Self-Paced Model for Teaching Programming. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*. ACM, Cambridge Massachusetts USA, 177–180. https://doi.org/10.1145/3051457.3053978

[23] The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. 2013. Computer science curricula 2013. (2013). http://dx.doi.org/10.1145/2534860.

[24] Claudia Ott, Brendan McCane, and Nick Meek. 2021. Mastery learning in cs1-an invitation to procrastinate?: Reflecting on six years of mastery learning. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. 18–24.

[25] Sandeep Purao, Maung Sein, Hallgeir Nilsen, and Even Åby Larsen. 2016. Setting the pace: Experiments with Keller's PSI. *IEEE Transactions on Education* 60, 2 (2016), 97–104.

[26] Anthony Robins. 2010. Learning edge momentum: A new account of outcomes in CS1. *Computer Science Education* 20, 1 (2010), 37–71.

[27] Claudia Szabo and Judy Sheard. 2022. Learning theories use and relationships in computing education research. *ACM Transactions on Computing Education* 23, 1 (2022), 1–34.

[28] Bryan Whiting and Gary F Render. 1987. Cognitive and affective outcomes of mastery learning: A review of sixteen semesters. *The Clearing House* 60, 6 (1987), 276–280.

[29] Marshall Winget and Adam M Persky. 2022. A practical review of mastery learning. *American journal of pharmaceutical education* 86, 10 (2022), ajpe8906.