Notional Machines for Inclusive Learning

Dr. David Wilson *, Saquib Sarwar †, Dr. Nadia Najjar‡
* david.wilson@charlotte.edu, †ssarwar@charlotte.edu, ‡nadia.najjar@charlotte.edu
College of Computing and Informatics, University of North Carolina at Charlotte

***Abstract*—Understanding computing concepts is one of the foundational elements of computational thinking, which helps students formulate logic and algorithms for effectively developing and designing codes. However, novice learners often struggle to understand basic computing concepts, such as variables, loops, arrays, conditionals, and functions. This is often caused by the invisibility of the computing mechanism and the complexity of language syntax and conventions. Moreover, challenges in learning computing concepts can be compounded for learners with atypical conditions (e.g. visual, cognitive, and learning impairment, attention deficit, and neurodivergence). To tackle this, teachers of typical ability learners frequently use notional machines (NMs) — simplified representations (e.g. diagrams, flow charts, and analogies) created to assist novice programmers in understanding abstract concepts, program execution, and dynamics. In addition, traditional learning activities may need assistive learning interventions (ALIs) and learner-oriented practices like physical tactile models, computing artifacts, and interactive learning tools. NM approaches tend to focus on the explanation of concepts from instructor to learner, while ALIs tend to focus on learner support in activities. Both methods can provide advantages for learning, and we expect that a combined approach would provide even greater synergy for all learners. However, little consideration has been given to how NM and ALI approaches could be effectively combined. This research investigates how NM and ALI can be effectively brought together more holistically as an inclusive pedagogical tool — *notional machines for inclusive learning* (NMIL). In this full paper, we develop the conceptualization of NMIL, grounded in previous NM and ALI research. We also propose a structured method for creating NMIL. Finally, we also develop a physical NMIL model, based on the structured NMIL creation method, to teach *lists* in *Python*, as part of an introductory computer science course.**

***Index Terms*—Computational thinking, Pedagogy, Instructional design**

I. Introduction

Computational thinking is a problem-solving method that involves formulating problems and solutions for effective execution by computational devices [1], [2]. According to Brennan and Resnick, there are 3 key dimensions of computational thinking — computational concepts, practices, and perspectives [3]. Specifically, a clear understanding of computing concepts is the first step towards understanding programs, which can be quite challenging for novice learners [4]–[6]. For this reason, novice learners often face challenges while initiating, updating, and testing variables [2], writing recursive functions, and implementing loops and conditionals [2]. They have also shown misconceptions about nested and while loops [2], associations between range conceptions and loop boundaries, and the increase of control variable value in each step of the loop [4], [7]. Many misconceptions are also associated with specific programming languages, their conventions, and syntax [5].

According to duBoulay, one of the major issues behind these challenges and misconceptions is comprehending the underlying systems of computing concepts, which involves the program and the multi-layered interaction between the source code and the machine [8]–[10]. Novice learners have limited experience with these interactions as they primarily engage with the IDE and observe the outcome. Another key issue is the lack of specific support systems for novice learners. Resnick & Silverman used the terms "low floors, high ceilings, and wide walls" [11] to explain the properties of beginner-friendly tools. According to them, computing learning should be easy to get started (low floors), allow gradual exploration of complex concepts (high ceilings), and support creativity (wide walls). However, such interventions are not always adequate as individuals' learning abilities are diverse, and general support does not work for everyone. Alper, Hourcade & Gilutz [12] suggested providing *ramps* for low floors, *ladders* to reach the high ceilings, *frames of interest* to explore wide walls, and *reinforced corners* to develop confidence in program use [13].

Focusing on these issues, researchers and educators have explored ways to support novice learners. Educators frequently use analogies, metaphors, visualizations, and representations to help learners understand complex concepts in simpler forms [14]. These intuitive forms are often termed notional machines (NMs), a signature pedagogical tool in computing education [14]. NMs are widely adopted by educators to provide simple representations, developed based on teachers' conceptual knowledge, that highlight the complex mechanism and hidden state changes [14]. For example, variables are often referred to as boxes, and assigning integer values can be like putting a number block inside the box. Utilizing the power of simple analogies and representations, NMs lower the entry threshold of learning complex concepts and provide *low floors* to novice learners. However, educators' heavy reliance on visual and verbal representations [15], [16] (i.e. analogies, metaphors, diagrams, and flow charts) and generalization of students' learning abilities can restrict NM's adoption across learners with diverse learning abilities. With a complementary goal of supporting learners, accessibility researchers have also explored various assistive learning interventions (ALIs) that can provide engagement and hands-on learning opportunities to learners with atypical abilities. Many of these ALIs have also proven to be effective for learners across diverse abilities by providing *ramps* and *low floors*. For example, block-based

programming tools simplify learning CS concepts by removing language-oriented barriers and representing algorithms as construction blocks [17], [18]. Scratch [19] and Blockly [20] provide digital block-based programming environments. Roblock [21], e-block [22], and DBug [23] are tangible block-based interfaces developed for blind and visually impaired (BVI) learners, but help to support diverse ability learners through their physical nature. Other ALIs also include physical computing toolkits [21], [24]–[26], robot-based learning environments [21], [24], [27], and interactive systems [28].

While the perspective is different, NMs and ALIs both aim to support novice learners. NMs tend to focus on the explanation of concepts from instructor to learner, while ALIs provide practical affordances for learner support in activities. Both approaches have advantages, and we expect that a combined approach would provide even greater synergy for all learners. However, comparatively little consideration has been given to the possibility of combining the practices of NM and ALI as an inclusive intervention that can support assisted low-threshold entry in computing education.

Our research investigates how NM and ALI can be effectively brought together more holistically as an inclusive pedagogical tool — *notional machines for inclusive learning* (NMIL). In this full paper, we develop the conceptualization of NMIL as a pedagogical tool incorporating the practices of NM and ALI to support novice learners. We also develope a structured method to create NMILs and demonstrate the method by creating a physical NMIL model to teach lists in Python as part of a CS1 course. The primary research questions this work explores are:

1) How can notional machines be made more inclusive to support novices with learning challenges?
2) How to develop NMILs to ensure effective mapping between representations and the abstract concept?

This paper is organized as follows. We begin by discussing foundational research in NMs and ALIs, considering their focus and comparative advantages for computing education. Based on the review across the complementary areas, we propose a combined NMIL approach, highlighting its characteristics and connections with foundational NM and ALI approaches. To support NMIL in practice, we propose a framework for designing NMILs. Finally, we illustrate the NMIL approach through a specific NMIL physical model development process to teach *List* in *python* for an introduction to computing course. The paper concludes with a summary discussion and future work.

II. Background

Our work is built on two different approaches to teaching computing to novice learners, NMs and ALIs. They share a similar high-level goal for supporting learners, with research to explore teaching methods and interventions for learners from diverse communities, age groups, and abilities. However, they approach the goals with fairly different, but complementary, focus and perspectives.

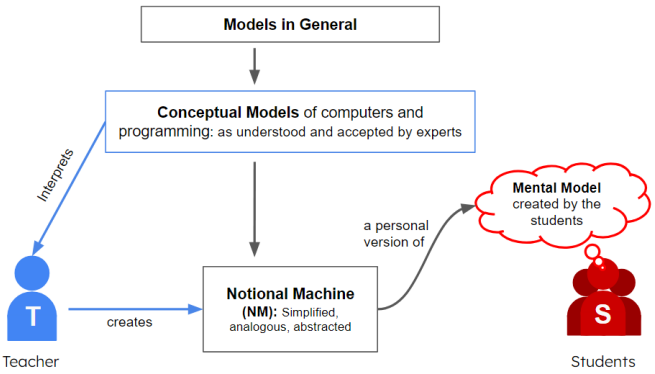

Fig. 1. The interplay between notional machine, conceptual model, and mental model developed by Fincher [14], where notional machines are a version of conceptual models developed by teachers for the students

A. Notional Machines in Computing Education

Notional machine (NM) is a signature pedagogical approach in computing education that explores teaching complex concepts through simplified representations [14], [29]. Du Bouley coined the term notional machine in the 1980s while explaining LOGO to children and teachers, emphasizing the use of a computer, its properties, and its relationship to programs. In the past four decades, this concept has been adapted in many ways, with researchers like Robins, Rountree, and Rountree [30] (in 2003) and Sorva [31] (in 2013) significantly contributing to its popularization, along with more recent work by Fincher, Jeuring, Miller, and others [14] (in 2020). Recent definitions of NMs have put program execution as central to distinguishing its characteristics.

NMs are developed based on a teacher's conceptual model of a specific topic [14], [16] to simplify the inner works of complex programs [32], [33] by the use of analogies and clarifying layers of abstraction, referred to as approximation or partial truth of the actual process [10], [14]. Hence, the term *notion* stands for the simplified concept, and *machine* represents the inner works, program behavior, and actions [14]. Du Bouley defined it as idealized and conceptual computers with properties directed by the conceptual element of the programming language. Fincher, Jeuring, Miller, and others [14] positioned NMs as a teacher's tool, designed for and delivered to the students [34] and used to shape the mental model of the learners [14], [16], [32] (see Figure 1).

Fincher and colleagues [14] conducted an empirical study to explore NMs in practice by researchers and educators. Digital tools are used to represent variables and objects [14] and to visualize execution steps [14] to assist with debugging. But Teachers are more accustomed to using hand-drawn representations (e.g. drawn on a whiteboard and digital space), and hand-made representations (e.g. paper cutouts of code segments and arrows, sticky notes, clothespins, and labels). For example, drawing arrows to show control flow for program execution, memory stack diagrams to indicate changes in variable values, and stings used to connect variables with

their reference values. Teachers also utilize different forms of analogies to connect computing concepts verbally with real-world examples, like referencing variables with parking spaces and using activities like dancing and roleplay to engage students with the representation.

As a pedagogical tool, NMs provide easier access to computing concepts using diverse representations as teachers reported using multiple NMs while teaching [14], [32]. However, there is limited research on the challenges and complexities of employing more diverse (e.g., physical / tangible) modalities for representation and their effective utilization, which is important to support diverse ability learners. Moreover, there are largely open research questions regarding students' interaction with NMs and the use of NMs by the users, as many argue that students should not only passively engage with NMs, but they should actively interact, engage with them, and adapt them according to their needs [16], [34].

B. Assistive Learning Interventions in Computing Education

Interventions in the field of computing education are developed to minimize the entry threshold and make learning interesting and engaging by providing *low floors*, *high ceilings*, and *wide walls* [11]. Assistive learning interventions accommodate the needs of learners with disabilities by incorporating assistive features, similar to adding *ramps* [12]. For example, graphical block-based interfaces like scratch support learning without language-related distractions (low floors), whereas tangible block-based interventions support blind and visually impaired (BVI) students to learn using multimodal interactions (ramps).

Block-based languages provide low floors by removing the need for complex characters, syntax, and white spaces. Physical blocks provide added accessibility by allowing interaction and manipulation of the physical object to explore constructionism and embodiment [35]. Added features (e.g., audio-haptic feedback) with physical blocks can also support learners with diverse abilities (e.g. visual impairment, motor impairment). For example, Storyblocks and Tip-toy support concept exploration through audio feedback support [13], [35], [36]. Modular accessible toolkits like Tern [37], [38] and Turino [28] allow users to create code, connect physical instruction beads, and use their parameters to create music [28]. These ALIs also provide frames of interest that enable users with different visual abilities and expertise to connect and collaborate. Extending the interactivity and feedback, Kazakoff et al. [39] used graphic and concrete blocks on the computer screen to learn program execution with LEGO®WeDo robots.

Low-cost DIY (do-it-yourself) solutions can also provide scaleable and personalized ALIs with hands-on exploration and haptic interaction opportunities. For example, Stefik et al. incorporated manipulative physical three-dimensional representations (e.g. box as a variable, die as a numeric value, and toggle light switch as a boolean value) [40] (Figure 3), Capovilla et al. employed Lego models in the teaching of sorting and searching algorithms, and [41] Jašková & Kaliaková used a tactile grid to teach simple algorithm execution. The children were given the task of writing a sequence of commands in a text editor that guided a bee-bot to follow a pre-set path through the tactile grid. The learners would simulate the execution of the program by moving the bee-bot with their hands [36]. Similarly, Papazafiropulos et al. used 3D printed models to teach concepts such as data structures and algorithms using cylinders of varying heights, with the height representing the value of the element. [42].

Multimodal, tangible, and physical representations, haptic-audio feedback, and hands-on learning opportunities are some of the ALI features that proved to be supportive and engaging for diverse ability learners [28], [35], [43]. In order to design and develop features to provide low floors and ramps, accessibility researchers adopt different human-centered approaches. These approaches are used to design, prototype, and develop solutions with features that accommodate learners' accessibility needs while fulfilling the learning outcomes [44]. Their approach often involves experts from the field of accessibility and special education; learners with atypical abilities and their peers; and teachers and teaching support staff. The learner-centered approach helps researchers understand their diverse needs and develop support accommodations that can often support collaboration in schooled settings [13], [27], [45] between learners with and without disabilities [13], [35] and develop inclusive ALIs that provide low floors and ramps.

C. The fundamental difference between NM and ALI: "Who" are the learners?

In the early examples of NMs, du Boulay developed NMs for students and teachers, positioning him as an expert [32] and making simplified representations for students and teachers. Similarly, Fincher positioned NMs as an intervention developed based on the conceptual model of the teacher (i.e. expert with the clean conceptual model of the concept) to support the mental model of the student (i.e. learner) [14]. Focusing on these two pieces of literature from 1981 and 2020, we find the researchers talking about simplifying the teaching process from teachers' point of view. However, these do not specifically explore learners' abilities to understand the concepts and the adoptibility of NMs. Moreover, learner communities and teaching methods are increasingly diverse, which makes it important to consider the socio-cultural background of the learners as well as course delivery approaches (i.e. online, face-to-face, synchronous, and asynchronous).

On the other hand, ALIs are developed by paying significant attention to the end users, focusing on usability, adaptability, function, and engagement. Following a learner-centered approach, ALIs are often designed and developed with the collaboration of special education experts, individuals with atypical learning abilities and their peers, and accessibility researchers [44], which is missing in NMs. Moreover, the greater adaptability and acceptance of ALIs across diverse-ability learners is associated with multimodal and playful interactivity. However, ALIs are often developed for children, which might not be suitable for high school and college students. Despite this, ALI's learner-centered approach creates a major difference in developing inclusive learning intervention,

which was not typically explored in NM literature with similar depth.

III. Inclusive Teaching Practices

Inclusion in education is a fundamental approach that helps ensure that all students, regardless of their physical, behavioral, or learning disabilities, are integrated into general education classrooms as much as possible. The concept of inclusivity in education covers a broad range of practices and principles that aim to provide equitable access to learning opportunities. However, achieving inclusivity in the context of education is not a trivial task and there is no one-time solution. Burgstahler provided a detailed checklist of practices, based on the universal design of instruction (UDL) framework. In the context of the teaching process, inclusive learning should support multiple ways to learn and interact with the course contents, with the goal of wider accessibility to incorporate learners of atypical abilities [46].

In the context of teaching computing, researchers have explored diverse ways to make the learning process inclusive, most of which are in line with Burgstahler's checklist. Some CS-specific recommendations involve supporting collaborative learning [13], [47], and embodied debugging [47], [48] while providing instructions in multiple representations [47]. However, researchers have specifically explored tangible interfaces, physical toolkits, and block-based interventions as physical interaction supports collaboration among diverse ability individuals [48], affords embodied engagement [49], and provides concrete examples [50].

IV. NM Perspective Analysis on ALI

Notional machines use diverse forms of analogies, representations, models, and activities to simplify CS concepts, focusing on the complex and invisible parts [14]. Assistive learning interventions often also involve some implicit level of abstraction as part of mapping to different modalities for hands-on interactive activities and multimodal feedback. To better understand the relationship between NM and ALI approaches for novice learner support, we analyzed several previously reported ALIs to characterize design aspects that could also be considered as primary elements of an NM framing. This process helps us to understand how "ramps" are created to support learners as a basis for NMIL development guidance.

To analyze and understand instructor NM practices toward effective NM design, Jayathirtha proposed a framework to capture NM in use with three basic questions [16], which we adopt as part of our analysis.

1) "What" concept is being taught?
2) "At what level of granularity" is the concept being explored?
3) "How" is the concept being represented?

Jayathirtha used this framework to capture the use of NMs in a physical computing classroom by qualitatively exploring the recorded video lectures [16]. Utilizing this approach, we will explore the use of ALIs to teach concepts, focusing on specific parts, supported by a diverse range of representations. For example, representing codes as blocks and connecting code development with a construction process, both virtually [51] and physically [28], [37], [38].

A. "What" concept is being taught:

ALIs are used to teach a wide range of concepts using a diverse range of representations. For example, 3D manipulatives are used to represent different concepts at a basic level, such as variables and numeric values through a physical box and dice [40], program execution process through hands-on exploration on tactile grid [36], sorting and searching algorithm through organizing physical cylinders of different heights [42] and data visualization through tactile 3D maps [52]. ALIs use multi-layer simplifications and physical representations (*ramps*) to make the learning process non-visual, tangible, and engaging for diverse-ability students.

B. "At what level of granularity" the concept is being explored

ALIs are generally developed with a specific goal, which explores different concepts with multiple levels of focus. We incorporated four parameters to communicate program dynamics, introduced using ALIs, at a particular level of granularity [16], [33] — atoms, blocks, relations, and entire programs. For example, representing very basic concepts like variables, numeric values, and strings through physical items fits the *atom* parameter [40] whereas exploring array structure using physical cylinders of varying heights resembles the *block* parameter [42]. We can also see examples of *relations* parameters when learners explore cylinders of different heights to sort them [42]. Lastly, an example of the *entire program* parameter can be exploring loops and sequences through audio story and commanding robots to move (execution) [28], [35]–[37].

C. "How" is the concept being represented

ALIs are developed for students with diverse abilities, which leads to the incorporation of multi-modal interaction, hands-on exploration, and non-visual feedback. For example, *multimodal representations* (i.e. physical artifacts with audio, tactile, or haptic feedback) were used to provide access to the concepts that afforded interaction and exploration. Similar to the analogy that empty parking spots are like variables [14], accessibility researchers used the physical box analogy [40] to represent variables. *Similar concepts can be represented differently* based on expert design decisions. For example, numerical values were represented with dice [40] and 3D printed cylinders of varying heights [42]. In the first example, dice were used with a physical box representing variables. In the second case, cylinders were used to introduce arrays, and their physical appearance supported non-visual sorting activity. *Learning goal also determines representation modes*, like block-based coding tools were complemented with audio feedback [28], [37], [38] to allow non-visual exploration [28], [35]. Learner-centered materials for representations were also

common. For example, ATs for kids incorporated Legos [41], learning through music making [35], and gamified activities with toy robots [36] to grab their attention.

V. Notional Machines for Inclusive Learning

Based on the NM perspective analysis on ALI, we are proposing an inclusive-first approach that incorporates ALI practices as an explicit element of NM. We will refer to the approach as notional machines for inclusive learning (NMIL). NMIL is an inclusive pedagogical tool, intended to make computing concept learning more student-oriented, while incorporating the expertise of the educators. To properly understand NMIL, its use, and its representations, we explore its characteristics built on the characteristics of NM [14] and the practices of ALIs.

A. Pedagogical purpose

An NMIL should focus on a specific pedagogical purpose with simplified representations (notions) that depict programming mechanisms (machines) with an inclusive first approach. This means any learning intervention can be considered as NMIL if it has a pedagogical purpose and its representations are inclusive. NMIL can serve a pedagogical purpose in many ways — simplifying a concept, drawing attention to hidden mechanisms and state changes, focusing on program behaviors, or demonstrating system dynamics.

An NMIL is a teacher's tool developed based on their conceptual model to reinforce students' mental models, demonstrate dynamic program behavior, or allow them to explore the properties of computing concepts. While developing NMILs, teachers should follow a student-oriented approach to develop NMILs that enable exploration and active interaction. NM examples in literature use analogies (e.g. a variable is a box with a label) [14] to simplify concepts, explanative diagrams [53] to highlight program state changes, and even physical models [54] to teach references and objects. While these approaches support learning, they are often teacher-focused, being used primarily for explanation, so students don't really interact with them directly. NMILs should go one step further to ensure hands-on learning. For example, using physical manipulatives to represent variables [40], cylinders in a tray for sorting arrays through haptic exploration [42], and physical blocks with auditory feedback to teach sequence and loops [35] employ specific representations to support inclusive considerations.

As a simple demonstration of NMIL, we will look at a similar concept of teaching variables (Figure 2). Teachers can use verbal analogies (a variable is a box with a label) and diagrams (sketching a box on the board and labeling it) to introduce the concept. However, they can further support the student's mental model with physical artifacts (a labeled empty physical box) which students can interact with. This can be reinforced by engaging them in assigning values to the variable (putting a block with a value inside the box). Such activities have proven to be effective with learners with limited vision, who develop cognitive models of abstract concepts through hands-on exploration [40].

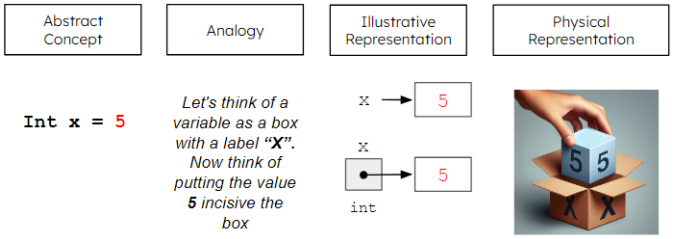

Fig. 2. A demonstration of the gradual incorporation of NMIL for learning computing concepts. Starting from abstract concepts (int x = 5), teachers often use verbal analogies and diagrams in the classroom. The details on the diagrams can differ based on the concepts being taught. These diagrams were developed based on [53]. teachers can introduce physical models to better understand and reinforce students' conceptions. Teachers can reinforce learning by engaging students in interactive activities like assigning values to the variable. These physical models were developed by [40])

B. Drawing Attention

NMIs should draw attention to the invisible parts, which might not be apparent through the coding environment or need support for better conceptualization. For example, NMIL can draw attention to program execution processes, state change, memory allocation, or program dynamics by using multi-dimensional representations and tactile-audio-haptic feedback to highlight the abstractions. During a field study at the Washington State School for the Blind, Stefik and his team found that teachers introduce concepts using manipulative objects that students can touch and explore [40], like using a light switch to explain a boolean variable or a string with beads as a sting variable (Figure 3). Such tangible and haptic representations can help uncover the concept while making it relevant to everyday objects.

Similarly, audio-tactile feedback can also be vital in highlighting the hidden process of the programs. StoryBlocks [35] is an assistive system for computing concept understanding through multimodal interaction. It utilizes physical block, which is often used by researchers [13], [35], [55], to highlight the similarity with the "real-life construction" process. Moreover, students could explore concepts like *sequence* by connecting blocks like *Cat* $\rightarrow$ *Run* $\rightarrow$ *Mouse* to create an audio story like "the cat runs after the mouse" [35]. Similarly, **loops** can be visualized with repeated audio stories. Such multimodal representation can reinforce learning through visualizing the hidden process which can be cognitively challenging for the learners to understand.

Fig. 3. Physical representations of numeric, boolean, and string variables from [40]

C. NMIL with Focus

This focus of NMIL can be very broad, (e.g. program execution and behavior) or very atomic (e.g. name / label of

a variable or numeric values assigned to variables), based on the pedagogical goal. The focus can effectively be achieved by utilizing inclusive practices like block-based tools (e.g. StoryBlocks [35] and Tip-toy [13]) and learning interventions (e.g. Torino [28] and Tern [37]), where the primary focus is conceptual development. Moreover, the NMIL of a topic / concept can be different based on the focus and specific purposes. For example, Papazafiropulos et al used 3D-printed cylinders of different heights, held in a custom tray, as physical models of the array [42]. Although arrays can be represented with myriad variations, Papazafiropulos used this specific kind of representation because the focus was to teach sorting algorithms. Cylinders of different heights allowed learners to explore the numeric differences. Similarly, educators can use various properties and dimensions of NMIL to focus on processes like- the mechanism of array creation, updating values of indices, adding / removing certain indices, or state changes of the system when indices are injected or removed. Similarly, affordances of physical objects can be used to focus on relations, like- using strings to show reference, using empty pockets or boxes to show unassigned variable [40], using light switches to show boolean [40], or using sticky notes to show the ability remove and reassign names / labels.

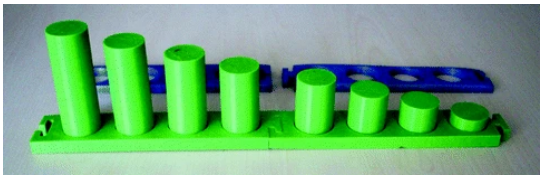

Fig. 4. 3D-printed cylinders of varying heights, held in a custom tray, as physical models of the array [42]

D. Representation through NMIL

An NMIL can represent the complicated inner workings of a system, program behavior, or computing concept by using multi-dimensional representations (e.g. physical models, audio-haptic feedback, manipulatives, activities, and roleplay). According to Fincher and colleagues [14], their view in NM is more concerned with *what* is being represented than with *how* it is being represented. But NMIL should equally emphasize the "how" with the "what". Because "what" explores the requirements of the educators, while "how" explores the support for the learners. For example, when teachers use the "a variable is a box with a label" analogy [14], the capacity of the box is not explicitly mentioned and different students might "conceptualize" the process differently- is the box only the size of one value or can it hold multiple values? On the other hand, using a physical box to represent the variable and a cube [40] (with the same volume as the box) to represent the value will support the students in conceptualizing the accurate simple representation of a variable.

Inclusive learning interventions and guidelines also encourage multimodality, where the physicality of education materials is one of the basic requirements. According to Burgstahler's checklist for inclusive teaching [46], the physical nature of teaching materials can deliver an inclusive teaching environment, cognitive support, and multiple entry points. Moreover, physical interaction can create a collaborative learning experience [48], afford embodied and engaging learning [49], and make affiliation with abstract concepts easier [50]. Because of the inclusive nature of multimodal representations, ALIs developed for learners with mixed visual abilities were found to be effective with learners with typical vision.

Based on the proposed NMIL pedagogical tool and its characteristics, we have devised a structured method for creating physical models such as NMIL. In the next section, we will first explore the design method for developing physical models.

VI. A structured approach to developing NMIL

NMs are generally used by teachers and developed based on their conceptual models. Previous research has developed only limited general guidance on NM construction. For example, Dickson, Richards, and Becker [34] describe dividing the process into three conceptual categories — state, rules, and knowledge. According to them, NM is generally a set of rules that govern the state of the machine, and knowledge helps to understand those rules [34]. Across previously reported NM development, we found that identifying the representation medium and developing rules to use the representation were common. ***However, NMs are almost always created by the teachers and there is no explicit description of the process, which makes it difficult to replicate and recreate NMs.***

Contrasting this, ALIs typically follow a specific design process, which is moderated by the experts (e.g. teachers, researchers, and special education experts) and often involves the learners in designing and evaluating the intervention. Exploring the practices of NMs and ALIs and focusing on our second research question (RQ2), we are proposing a simple structure to develop NMIL, connected with concepts and learning goals while bringing NMIL characteristics into practice. Our approach consists of 5 specific steps: (1) Context, (2) Concept Identification, (3) Learning Goals, (4) Simplified representation, and (5) NMIL.

Context starts with identifying the course structure, delivery method (i.e. online or face-to-face delivery, synchronous or asynchronous structure), the programming language (python, java, C++), and the student's level of education (i.e. high school, undergrad, or grad students) to understand the existing constraints, opportunities, and target learners. All this information is crucial for the effectiveness of NMIL as some features might not be suitable for others. The multimodal interactivity features (e.g. music making, audio-story development) explored by ALIs are often targeted towards children [13] which might not be suitable for undergrad-level students [54].

Concept Identification involves selecting the concept (e.g. array, loop, conditions) that instructors want to draw attention to and focus on. For example, instructors might want to focus on a specific topic, the program behavior [16], or the entire course [34], [54]. The concept can also be based

on students' performance, teachers' perspectives, or research-based observation.

Learning Goals are the foundation of the design choices of the NMIL, developed based on the target concept, pedagogical focus, course materials, and even their students' needs. An example can be drawn from the different representations of *Arrays* by Mazumder [53], Lewis [54], and Papazafiropulos [42] with the same target concept but different learning goals.

Simplified representations (e.g. analogies, metaphors, or diagrams) can work as a starting point to develop NMILs and work as a connection between a concrete representation and an abstract concept [56] while drawing attention to that physical form. For example, Referring to a variable as a box is a simple analogy, that can lead to the use of a physical box to learn a variable.

NMIL is the final step which is build on the earlier steps — context, concept, learning goals, and simplified representation. Following these, NMIL should reflect the teaching requirements and pedagogical purpoer. To support inclusive interaction, NMILs should also incorporate visual and tactile aids, use large characters, symbols, and contrast colors, or allow flexible multimodal exploration [46].

VII. Physical NMIL models Development and Demonstration

In this section, we will demonstrate the development of an NMIL model, based on our proposed structure, to expain *List* in the *Python* language for an introductory computing course. This course has been previously designed and instructed by the third author. Here, we selected the physical model as an NMIL for several reasons.

A. Initial Considerations

The process started with the second author meeting with the third author to understand the needs and requirements of the instructor, identify the initial considerations, and select the CS concept for NMIL development. The course instructor expressed the need for a physical representation of *Lists*. For the representation, she preferred physical models that could be readily available, so that the instructor was not burdened to design the models from scratch, and portable, so that those could be carried to the classrooms. The second author then explored the course lecture and slides to identify the topics that were focused for *Lists*. For example, indices, types of elements (e.g. integer or string) and their properties, list functions and methods, and mutability. Based on these, the learning goals were developed.

- LG1 Representations should clearly show indices
- LG2 Representations should clearly convey different types of elements
- LG3 Representations should clearly demonstrate mutability
- LG4 Representations should clearly demonstrate the reassignment of name / label of a list

Next, we developed a simplified representation of a list example, based on Mazumder's approach of developing explanative diagrams of Array in Java [53]. This explanation diagram was particularly chosen because the topics are comparable and support the representation and explanation of different components.

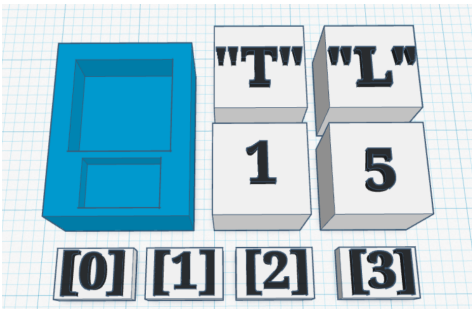

Fig. 5. Design of different element cubes and modules with separated index indicators in a 3D modeled view

B. Design Considerations of Physical Model

Taking the diagrams as a starting point, we started developing a physical NMIL model with some specific design considerations. First, to address the concerns of the course instructor, the models need to be easily accessible, developed, and portable. Next, following the inclusive teaching materials checklist by Burgstahler [46], models should support hands-on exploration. Lastly, we wanted to explore the physical properties of objects and connect them to abstract concepts, like a switch with a boolean variable, or using string to connect references [40].

Based on these design considerations, we started prototyping physical models. Our initial prototypes included paper cups as element holders with paper balls as elements. Following the explanative diagrams of the list, we understood that the element holders (i.e. paper cups) need to be in a connected sequence while showing their indices (LG1). However, the elements need to be interchangeable with other elements to show mutability (LG3) while representing its type (LG2). Lastly, the list model should also show its name, which also should be modifiable (LG4). With several iterations and prototypes, we developed our final form of the physical NMIL model.

C. Physical NMIL Model

Our final physical NMIL models consist of three types of components — the element cubes, the modules, and the list frame. Elements are represented through white *element cubes* with the string / integer value on them. The *modules* have the index value with space to place *element cubes*. The string / integer values on *element cubes* and the index value on *modules* are slightly offset from the top surface, to support better haptic exploration. Lastly, the *list frame* can hold multiple index *modules*, side by side, to properly represent a list. It also has a specific place to add the name / label of the list, with sticky notes (Figure 6).

To make the physical NMIL model inclusive, we adopted a modular block-based design pattern, following existing block-based computational tools [13], which allows learning in a

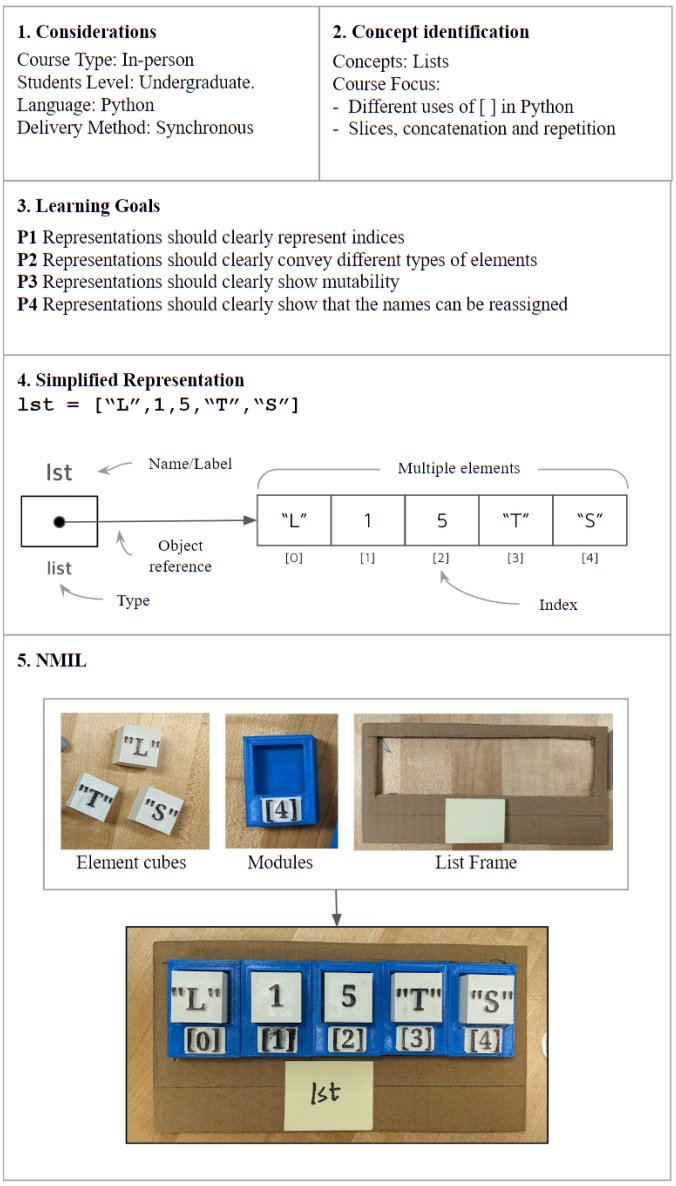

Fig. 6. Demonstrating the 5 steps of structured NMIL development method

tabletop setting. We used contrasting colors to differentiate the components of the model. Moreover, to support easy access to these physical models, we developed these as 3D models (figure 5) which can be fabricated using desktop 3D printers and used with minimum post-processing. Lastly, the rectangular design elements allow stackable components for easier portability. Figure 6 shows each step of developing the physical NMIL model in brief.

VIII. Conclusion

This paper introduces the concept of notional machines for inclusive learning (NMIL), which combines instructor-centered approaches of notional machines (NMs) with student-oriented practices of assistive learning interventions (ALIs). NMIL was developed to provide a more inclusive pedagogical tool to support novice students who it difficult to understand computing concepts(RQ1). We further developed a structural approach for NMIL development to map the relationship between computing concepts and representations (RQ2). Additionally, physical NMIL models were created to teach Lists in *Python*, serving as a demonstration of our approach. Our future work involves evaluating the physical NMIL models by implementing them in classroom settings, in an introductory computing course.

References

[1] J. M. Wing, "Computational thinking and thinking about computing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, pp. 3717–3725, 2008.

[2] B. Cetin, "Academic motivation and self-regulated learning in predicting academic achievement in college." *Journal of International Education Research*, vol. 11, no. 2, pp. 95–106, 2015.

[3] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, vol. 1, 2012, p. 25.

[4] B. Du Boulay, "Some difficulties of learning to program," in *Studying the novice programmer*. Psychology Press, 2013, pp. 283–299.

[5] J. R. Anthony Robins and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, 2003.

[6] S. Wiedenbeck, V. Ramalingam, S. T. Sarasamma, and C. L. Corritore, "A comparison of the comprehension of object-oriented and procedural programs by novice programmers," *Interact. Comput.*, vol. 11, 1999.

[7] D. Ginat, "On novice loop boundaries and range conceptions," *Computer Science Education*, vol. 14, no. 3, pp. 165–181, 2004.

[8] M. Berry, *The Design and Implementation of a Notional Machine for teaching Introductory Programming*. University of Kent, 2015.

[9] M. Guzdial, *Learner-centered design of computing education: Research on computing for everyone*. Morgan & Claypool Publishers, 2015.

[10] M.Guzdial, "Balancing teaching cs efficiently with motivating students," *Commun. ACM*, vol. 60, no. 6, p. 10–11, may 2017. [Online]. Available: https://doi.org/10.1145/3077227

[11] M. Resnick and B. Silverman, "Some reflections on designing construction kits for kids," in *Proceedings of the 2005 Conference on Interaction Design and Children*, ser. IDC '05. ACM, 2005, p. 117–122.

[12] M. Alper, J. P. Hourcade, and S. Gilutz, "Adding reinforced corners: designing interactive technologies for children with disabilities," *Interactions*, vol. 19, no. 6, p. 72–75, nov 2012.

[13] G. Barbareschi, E. Costanza, and C. Holloway, "Tip-toy: a tactile, open-source computational toolkit to support learning across visual abilities," in *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, ser. ASSETS '20. ACM, 2020.

[14] S. Fincher, J. Jeuring, C. S. Miller, P. Donaldson, B. du Boulay, M. Hauswirth, A. Hellas, F. Hermans, C. Lewis, A. Mühling, J. L. Pearce, and A. Petersen, "Notional machines in computing education: The education of attention," in *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR '20. New York, NY, USA: ACM, 2020, p. 21–50.

[15] G. Jayathirtha and Y. B. Kafai, "Program comprehension with physical computing: A structure, function, and behavior analysis of think-alouds with high school students," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE '21. ACM, 2021, p. 143–149.

[16] G. Jayathirtha, "Video analysis of a teacher's use of notional machines in an introductory high school electronic textile unit: A three-tier framework to capture notional machines in practice," in *Proceedings of the 17th Workshop in Primary and Secondary Computing Education*, ser. WiPSCE '22. ACM, 2022.

[17] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: students' perceptions of blocks-based programming," in *Proceedings of the 14th international conference on interaction design and children*, 2015, pp. 199–208.

[18] M. Kölling, N. C. Brown, and A. Altadmri, "Frame-based editing." *J. Vis. Lang. Sentient Syst.*, vol. 3, no. 1, p. 1, 2017.

[19] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[20] N. Fraser, "Ten things we've learned from blockly," in *2015 IEEE blocks and beyond workshop (blocks and beyond)*. IEEE, 2015, pp. 49–50.

[21] E. Schweikardt and M. D. Gross, "The robot is the program: interacting with roblocks," in *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*, ser. TEI '08. ACM, 2008.

[22] D. Wang, Y. Zhang, T. Gu, L. He, and H. Wang, "E-block: a tangible programming tool for children," in *Adjunct proceedings of the 25th annual ACM symposium on User interface software and technology*, 2012, pp. 71–72.

[23] M. Bodén, B. Pretorius, B. Matthews, and S. Viller, "Dbugs: large-scale artefacts for collaborative computer programming," in *Proceedings of the 17th ACM Conference on Interaction Design and Children*, 2018.

[24] P. Wyeth, "How young children learn to program with sensor, action, and logic blocks," *The Journal of the learning sciences*, vol. 17, no. 4, pp. 517–550, 2008.

[25] A. Sullivan, M. Elkin, and M. U. Bers, "Kibo robot demo: engaging young children in programming and engineering," in *Proceedings of the 14th International Conference on Interaction Design and Children*, ser. IDC '15. ACM, 2015, p. 418–421.

[26] M. Gordon, E. Rivera, E. Ackermann, and C. Breazeal, "Designing a relational social robot toolkit for preschool children to explore computational concepts," in *Proceedings of the 14th International Conference on Interaction Design and Children*, ser. IDC '15. ACM, 2015.

[27] I. Neto, H. Nicolau, and A. Paiva, "Community based robot design for classrooms with mixed visual abilities children," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. ACM, 2021.

[28] C. Morrison, N. Villar, A. Thieme, Z. Ashktorab, E. Taysom, O. Salandin, D. Cletheroe, G. Saul, A. F. Blackwell, and D. Edge, "Torino: A tangible programming language inclusive of children with visual disabilities," *Human–Computer Interaction*, vol. 35, no. 3, 2020.

[29] L. Shulman, "Knowledge and teaching: Foundations of the new reform," *Harvard educational review*, vol. 57, no. 1, pp. 1–23, 1987.

[30] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer science education*, vol. 13, no. 2, pp. 137–172, 2003.

[31] J. Sorva, "Notional machines and introductory programming education," *ACM Trans. Comput. Educ.*, vol. 13, no. 2, 2013.

[32] B. Du Boulay, T. O'Shea, and J. Monk, "The black box inside the glass box: presenting computing concepts to novices," *International Journal of man-machine studies*, vol. 14, no. 3, pp. 237–249, 1981.

[33] C. Schulte, "Block model: an educational model of program comprehension as a tool for a scholarly approach to teaching," in *Proceedings of the Fourth International Workshop on Computing Education Research*, ser. ICER '08. ACM, 2008.

[34] P. E. Dickson, T. Richards, and B. A. Becker, "Experiences implementing and utilizing a notional machine in the classroom," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, ser. SIGCSE 2022. ACM, 2022.

[35] V. Koushik, D. Guinness, and S. K. Kane, "Storyblocks: A tangible programming game to create accessible audio stories," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. ACM, 2019.

[36] L. Jašková and M. Kaliaková, "Programming microworlds for visually impaired pupils," in *Proceedings of the 3rd international constructionism conference.* OCG, 2014.

[37] M. S. Horn and R. J. K. Jacob, "Tangible programming in the classroom with tern," in *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '07. ACM, 2007.

[38] ——, "Designing tangible programming languages for classroom use," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, ser. TEI '07. ACM, 2007.

[39] E. R. Kazakoff, A. Sullivan, and M. U. Bers, "The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood," *Early Childhood Education Journal*, vol. 41, no. 4, pp. 245–255, 2013.

[40] A. M. Stefik, C. Hundhausen, and D. Smith, "On the design of an educational infrastructure for the blind and visually impaired in computer science," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. ACM, 2011.

[41] D. Capovilla, J. Krugel, and P. Hubwieser, "Teaching algorithmic thinking using haptic models for visually impaired students," in *Proceedings of the 2013 Learning and Teaching in Computing and Engineering*, ser. LATICE '13. IEEE Computer Society, 2013.

[42] N. Papazafiropulos, L. Fanucci, B. Leporini, S. Pelagatti, and R. Roncella, "Haptic models of arrays through 3d printing for computer science education," in *Computers Helping People with Special Needs*. Springer, 2016.

[43] A. Thieme, C. Morrison, N. Villar, M. Grayson, and S. Lindley, "Enabling collaboration in learning computer programing inclusive of children with vision impairments," in *Proceedings of the 2017 Conference on Designing Interactive Systems*, 2017, pp. 739–752.

[44] S. Sarwar and D. Wilson, "Systematic literature review on making and accessibility," in *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 2022.

[45] G. Regal, D. Sellitsch, S. Kriglstein, S. Kollienz, and M. Tscheligi, "Be active! participatory design of accessible movement-based games," in *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, 2020.

[46] S. Burgstahler, *Equal access: Universal design of instruction*. DO-IT, University of Washington, 2008.

[47] Z. Lechelt, Y. Rogers, N. Yuill, L. Nagl, G. Ragone, and N. Marquardt, "Inclusive computing in special needs classrooms: Designing for all," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[48] M. S. Horn, E. T. Solovey, R. J. Crouser, and R. J. Jacob, "Comparing the use of tangible and graphical programming languages for informal science education," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 975–984.

[49] O. Zuckerman and A. Gal-Oz, "To tui or not to tui: Evaluating performance and preference in tangible vs. graphical user interfaces," *International Journal of Human-Computer Studies*, vol. 71, no. 7-8, 2013.

[50] S. Sentance, J. Waite, S. Hodges, E. MacLeod, and L. Yeomans, ""Creating Cool Stuff": Pupils' experience of the bbc micro:bit," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 2017.

[51] S. Ludi, "Position paper: Towards making block-based programming accessible for blind users," in *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, 2015, pp. 67–69.

[52] S. K. Kane and J. P. Bigham, "Tracking @stemxcomet: teaching programming to blind students via 3d printing, crisis management, and twitter," in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. ACM, 2014.

[53] S. F. Mazumder, C. Latulipe, and M. A. Pérez-Quiñones, "Are variable, array and object diagrams in java textbooks explanative?" in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '20. ACM, 2020.

[54] C. M. Lewis, "Physical java memory models: A notional machine," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM, 2021.

[55] V. Gadiraju, A. Muehlbradt, and S. K. Kane, "Brailleblocks: Computational braille toys for collaborative learning," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, 2020.

[56] S. Suh, M. Lee, and E. Law, "How do we design for concreteness fading? survey, general framework, and design dimensions," in *Proceedings of the Interaction Design and Children Conference*. ACM, 2020.