Construction of a Taxonomy for Tools, Languages, and Environments across Computing Education

Monica M. McGill
Knox College & CSEdResearch.org
Galesburg, IL, USA
monica@csedresearch.org

Adrienne Decker
University at Buffalo
Buffalo, NY, USA
adrienne@buffalo.edu

ABSTRACT

The sheer number of tools, languages, and environments (TLEs) used in computing education has proliferated in the last few years as more tools are developed to meet new demands of the growing amount of K-12 computing education that has been undertaken. However, there is little formalized language at either the K-12 or post-secondary level that provides for a way to classify these TLEs for discussing research and for classifying in databases.

In this research study, we step through a formal process for building a taxonomy for TLEs. As part of the supporting research, we first discuss the importance of taxonomies and classification systems in computing education, provide a formal method for building a taxonomy, and provide working definitions of TLEs based on previous literature. This is followed by a systematic literature review using a widely-accepted methodology for finding articles that have examined TLEs in primary, secondary, and post-secondary computing education. This literature review focuses on studies that looked at multiple TLEs and specifically attempted to classify or categorize them. We then propose a new taxonomy for classifying TLEs and provide definitions and samples for each category. This is followed by a discussion of the next steps in vetting the taxonomy and the challenges and issues that need to be considered when evaluating it for classifying TLEs in computing education.

CCS CONCEPTS

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

KEYWORDS

Languages, tools, environments, K-12, primary, secondary, post-secondary, computing, education, literature review, taxonomy, ontology, classification

ACM Reference Format:
Monica M. McGill and Adrienne Decker. 2020. Construction of a Taxonomy for Tools, Languages, and Environments across Computing Education. In *Proceedings of the 2020 International Computing Education Research Conference (ICER '20), August 10–12, 2020, Virtual Event, New Zealand*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3372782.3406258

ICER '20, August 10–12, 2020, Virtual Event, New Zealand

ACM ISBN 978-1-4503-7092-9/20/08.
https://doi.org/10.1145/3372782.3406258

1 INTRODUCTION

As defined by Whittaker and Breininger, a taxonomy is "... a controlled vocabulary with each term having hierarchical (broader and narrower) and equivalent (synonymous) relationships. Because of its hierarchical nature, a taxonomy imposes a topical structure on information." [73, p. 2] One of the earliest, well-known and widely-used taxonomies relates to the sciences. Linnaeus's 18th century taxonomy names, defines and classifies groups of organisms across the biological life sciences [15], which remains as the basis for the U.S. National Institutes of Health Taxonomy database [51]. Likewise, Dewey (1876) decimal classification for data also represents a taxonomy that is still embedded in libraries today [20]. Ontologies, closely related to taxonomies, provide "...a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base" [67]. They can be used to define "...the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary" [52, p. 2].

More recently, taxonomies have moved well beyond their initial usage in biology and have been integrated across many disciplines, such as user experience, e-commerce, marketing, and interactive media [44, 69, 71]–essentially, digital data that are stored in a database or represented in an interface and that requires a logical grouping. Taxonomies and other types of classifications of computing education data have been studied over the last several decades [31, 37, 39]. A subset of these studies have focused on tools, languages, and environments (TLEs), such as Singh (1990), Price et al. (1993), Wright and Cockburn (2003), Kelleher and Pausch (2005), and Gómez-Albarrán (2005) [28, 35, 58, 64, 74].

The sheer number of TLEs used in K-12 computing education has proliferated in the last few years as more tools are developed to meet new demand [45]. However, as more are created, how we refer to TLEs with similar characteristics have not been defined in a systematic way nor have the few classifiers in use (e.g., block-based, tangible computing, etc.) been examined closely to know if they consider future expansion and adaptability as the field changes, two key issues raised by Price et al. [58]. To explore this further, for this study we adopted the following research questions:

- What taxonomies exist for TLES?
- What would a modern taxonomy of TLEs contain?

For the first phase of this study, we examine taxonomies that have been created for TLEs in the past few decades through a systematic literature review, followed by a proposed taxonomy of TLEs, replete with definitions and examples. We provide here a summary of the importance of taxonomies, a description of the methodology we used to form the taxonomy, a working definition of TLEs for this

study, the Systematic Literature Review (SLR) for identifying past literature for related taxonomies, and a summary of how we built the taxonomy. This is followed by the proposed taxonomy and a discussion. We then provide an overview of the next phase of this study, reflecting on the need to gather evidence of the proposed taxonomy's validity throughout the broader computing education research community.

2 TAXONOMY IMPORTANCE AND USAGE

In general, taxonomies are now heavily used in information management, due to their importance and usefulness for adding to databases and keyword tagging to make information storage and retrieval manageable and to present data hierarchically [72]. In research, lack of common terminology can result in fragmentation of a field with researchers using differing sets of terminology [22]. This has the potential of leading to a duplication of efforts, since researchers searching by keywords using their own terminology may be quite different than those in published articles. Taxonomies can provide a way to mitigate these problems by mapping the field so research studies share common vocabulary that is consistent and persistent in the field, thereby preventing a proliferation of various synonyms [5, 22]. Likewise, [16] (2003) believed classifications are valuable "...to achieve preciseness and economy in making generalizations," [16] while they can also be used to guide future research [46].

Closer to education research and practice, several notable taxonomies exist, including Bloom's and SOLO taxonomies for describing processes involved in cognition and learning [8, 9]. Bloom believed that "...his taxonomy could serve, among other things, to provide a common language of reference, defining educational goals, and provide a panorama of educational possibilities" [37, p. 8]. These taxonomies have been widely researched and used within computing education and, in part, they illustrate their importance as Bloom anticipated [23, 27, 34, 38, 39, 70].

Taxonomies can provide a way to identify affordances, as well as advantages and disadvantages of TLEs, lending themselves as a framework for assessing whether a particular TLE is suitable to a specific teaching scenario as well as evaluating and comparing TLEs [10, 11, 49, p. 1222]. They can also be used to direct the design of future digital systems by pinpointing gaps that could be potentially met by new or enhanced systems [46].

Kelleher and Pausch's (2005) reasons for building a taxonomy of programming environments and languages are rooted in their desire to identify the sociological factors that prevent students from learning programming [35]. Their efforts were made in order to identify which TLEs addressed barriers to learning, including mechanical barriers that place unnecessary cognitive load on learners, as well as the gaps in these systems. Malmi et al. discuss several classifications of tools, noting the need for structure in this area and providing their own groupings and definitions of tools [41].

Finally, taxonomies "...enable the development of IS research databases with high levels of user performance characteristics" [5, p. 299]. In various ways, the taxonomies mentioned reflect on Linnaeus's efforts of naming, defining, and then classifying groups of items that have similar characteristics. Though all of these reasons provided above are important, our primary motivations for undertaking this research study is to enable the structuring of data using agreed upon keywords within a large dataset for computer science education researchers [5] and to update older taxonomies to reflect modern TLEs currently infiltrating primary and secondary computing education. Our work is part of larger work similar to Kelleher and Pausch [35] and reflects our efforts to determine where the gaps are in meeting the needs of the CS for All community working to promote effective CS learning across primary and secondary education.

3 METHODOLOGY

Gavrilova et al. (2005) proposed an algorithm for the development of an ontology, which are closely related to taxonomies. These steps included developing a glossary, laddering, desintegration, categorizing, and refining the ontology [24, p. 2]. Various forms of taxonomies can be dynamically created from databases or sets of data using tools that leverage natural language processing and statistical clustering [72]. Datasets that lend themselves well to auto-generated taxonomies are keyword focused and typically well-defined, such as corporate documents that already have topics and subtopics from which hierarchies can be formed. There are also examples of taxonomies being created from keywords, such as in Engineering Education Research [22]. Turning to steps for creating a taxonomy for our study, we follow steps provided by Whittaker and Breininger [73], who specify seven steps to creating a taxonomy for managing knowledge:

(1) Determine requirements
(2) Identify concepts
(3) Develop draft taxonomy
(4) Review with users and subject matter experts
(5) Refine taxonomy
(6) Apply taxonomy to content
(7) Manage and maintain taxonomy

For the purposes of this phase of our research study, we explored steps 1 through 3 of the Whittaker and Breininger framework to develop a taxonomy for TLEs. These are described below with their corresponding sections in this paper:

- Section 4: Identify the concepts by providing a broad definition of tools, languages, and environments
- Section 5: Develop requirements for this study via a systematic literature review (SLR) of existing taxonomies for TLEs
- Section 6: Identify and define the concepts of the taxonomy based on the results of the SLR
- Sections 7: Develop a draft taxonomy, including names, definitions, and examples of the targeted TLEs based on the requirements and concepts
- Sections 8 & 9: Define next steps in gathering evidence of validity and reliability of the taxonomy (including steps 4-7 above)

This methodology provided us with the necessary steps to execute in order to answer our research questions. Since the first two steps are closely related, we chose to provide a working definition for TLEs first, then examine requirements.

4 WORKING DEFINITION OF TLES

Providing formal definitions of TLEs is a challenging task that could warrant a separate SLR with many months of work and, at its conclusion, may not prove useful to this task. In this section, we loosely cover several definitions and then provide a working definition for TLEs that we refer to throughout the rest of this study.

Al-Abri et al. (2017) define a tool as "...the instrument which is necessary to complete the purpose of the task based on the method applied" [2, p. 884]. In the context of computing education, Malmi et al. focus on digital tools, forming them as software applications, web applications providing some service, software frameworks, or definition languages [41]. While Malmi et al. restrict their overview of software tools to those that do not include physical computing or other unplugged activities (like CS Unplugged), others include these as tools per Al-Abri et al.. Horn and Bers (2019) explore the term tangible computing as including digital manipulatives, representation of computer code through the use of physical objects, those that blend movement, action, and physical space through programming, robots, and crafted materials like E-textiles [32]. They also provide a broad classification, including smart block languages, tangible demonstration languages, and externally compiled languages.

Though not limited to computing education, Deek and McHugh (1999) defines programming environments, or what we often refer to now as interactive development environments, as "...systems used by programmers to develop and test programs" [19, p 133] These environments are a collection of "...tools that can be used in program construction, compilation, testing, and debugging: editors, language compilers, pre-coded function libraries, linking loaders, parsers, tracers, and debuggers" [19, p. 133].

With many tools, programming languages, and development environments in computing education, it is easy to identify and separate each. Python, for example, is a language. NotePad++, Sublime, and Emacs are tools for editing code. However, an integrated development environment (IDE) could support just one language, with the language being inseparable from the IDE, such as Scratch, making it impossible to break apart languages and environments [30]. In computational thinking, unplugged activities and physical manipulatives may be used and have shown to be effective [32].

Since there has not yet been a comprehensive definition that captures all of the learning aids we have discovered that have been included in research studies, we have chosen to use the phrase "tools, languages,and environments" to "...broadly capture those formal materials that have been studied in research studies or mentioned in experience reports as part of the overall student experience" which we have collected in ongoing research [45].

Materials is a term used to capture many forms of general media (e.g. textbooks, wikis, and forums). These are sometimes the subject of computer science education research or experience reports and as such could be classified as a TLE. We are aware that identifying products designed for primary and secondary education requires 1) a broad definition of tools, languages, and environments that reflect the rich set of past and current research and 2) be broad enough to capture near-future aids used for learning computer science. We also recognize that digital textbooks can incorporate editing and run-time environments, which can further conflate any process that attempts to separate these terms.

5 SYSTEMATIC LITERATURE REVIEW

To develop the requirements and answer our first research question, it was necessary to consider the research and taxonomies that have been previously published. We chose to conduct a systematic literature review (SLR) in accordance with the Khan et al. framework [36]. This five-step process includes:

- Step 1: Framing questions for a review
- Step 2: Identifying relevant work
- Step 3: Assessing the quality of studies
- Step 4: Summarizing the evidence
- Step 5: Interpreting the findings

For the remainder of this section, we provide an in-depth review of our steps used in this SLR.

5.1 SLR Step 1: Framing questions for a review

We first established a free-form question based on the goals for this research. The free-form question directly aligns with our first research question, but with more clarity: *What are the previously established taxonomies and classifications of tools, languages, and environments used in computing education research?*

We then derived the structured questions from the free-form question, breaking them apart into Khan et al.'s categories. Since the framework is primarily for interventions rather than theories, we modified that category to accommodate our needed analysis work:

- Population: Computer science education researchers, with a focus of K-12 CS education researchers
- Interventions or exposures: Reclassified as investigation into tools, languages, and environments (TLEs) used in computing education
- Outcomes: a taxonomy, hierarchy, or classification of TLEs
- Study design: review articles that propose a grouping, classification, or hierarchy of TLEs, and/or an in-depth review of available TLEs that produce (as a by-product) some grouping

For the study design, we were aware of articles that propose taxonomies of TLEs. We were also aware through our previous work that groupings and classifications may appear in articles, though they were not the original intent of the research being reported. Further, though we were particularly focused on TLEs for K-12 CS education, there are relatively fewer TLEs and taxonomies at this level. In addition, there is considerable overlap between TLEs designed for late secondary and early post-secondary learners. For these reasons, we included taxonomy work borne from all levels.

5.2 SLR Step 2: Identifying relevant work

For step 2 of the methodology, we established our selection criteria *a priori* from our resesarch questions. We chose to search the following publication venue libraries to make the search extensive: ACM Digital Library (limited to the ACM Full-Text Collection), IEEE Explore, and Google Scholar.

We established four search criteria to be executed across the three libraries:

- Search 1: Education AND “Programming language” AND (hierarchy OR classification OR taxonomy)

- Search 2: Education AND (Tools or languages or environments) AND (hierarchy OR classification OR taxonomy)
- Search 3: “computer science” AND education AND (hierarchy OR classification OR taxonomy)
- Search 4: Education AND (“Integrated Development Environment” OR IDE) AND (hierarchy OR classification OR taxonomy)

Our goal was to include the first 50 articles found in each search, which would result in 600 articles to review (minus duplicates). Upon conducting the four searches in the ACM Digital Library, we discovered through analysis that the AND Boolean logic operator did not appear to be working as is expected for a logic statement. For example, a search of the first returned article in the fourth search criteria, stripped down to simply Education AND "Integrated Development Environment" AND Classification resulted in 268 results. However, when the first result returned was examined, the word “Classification” did not appear in the article, inferring that the AND operator is treated as the Boolean operator OR.

Since we were skeptical of these results, we performed the searches on Google Scholar. After seeing that the results of the first search criteria included articles from both ACM and IEEE publication venues, and seeing familiar and relevant articles (e.g. Kelleher and Pausch), we changed our strategy to only search Google Scholar, but expanded the number of articles to the top 100 results returned, which would then give us a total of 400 articles (minus duplicates) to review.

We placed all articles in a Google spreadsheet so we could identify and remove duplicates. The titles and authors of the results provided some assurance that they were falling into the scope of our desired literature. The searches were conducted in March, 2020. Of the 400 articles, 33 duplicates were found, leaving 367 articles. Of these duplicates, none appeared in the results of all four searches, but the following three appeared across three of the searches:

- Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys (CSUR), 37(2), 83-137.
- Rongas, T., Kaarna, A., & Kalviainen, H. (2004, August). Classification of computerized learning tools for introductory programming courses: learning approach. In IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings. (pp. 678-680). IEEE.
- Tomei, L. A. (2005). Taxonomy for the technology domain. In *Taxonomy for the Technology Domain* (pp. 89-108). IGI Global.

This gave us additional assurance that we were collecting the data that we intended.

After consideration, we determined that the inclusion criteria for articles would be one or more of the following:

- Contained any discussion on tools, languages, or environments in the context of computing education
- Contained any discussion on the relevance or importance of taxonomies in the context of computing education

Next, we went through each of the remaining 367 articles to determine whether or not they met our inclusion criteria. As we reviewed articles, it became clear that our second criteria was too limiting. We then changed the criterion to "Contained any discussion of taxonomies in the context of computing education" and re-reviewed articles where necessary and applied the new criterion to articles that were still under review. This broader criterion would enable us to capture additional articles that might serve as models of creating taxonomies while at the same time encompass any articles that also created taxonomies of TLEs within computer science education.

Since inter-rater and intra-rater reliability can change across time [7], we adopted the following methodology to mitigate these issues:

- We examined the first 75 articles together over a three-day period to see if they met our search criteria, discussing any discrepancies.
- We then examined a unique set of 25 articles independently, for a total of 50 articles.
- After reviewing the articles independently, we met daily to discuss and address any issues that arose during our independent reviews.
- We then examined the next 10 articles together.
- We repeated the last three steps until all articles were reviewed.

Thus, we examined 115 together and 126 each independently. By reviewing 10 articles together at regular daily intervals and discussing any discrepancies in our work performed individually, we were able to provide greater assurance that our method of evaluating the articles against our inclusion criteria remained the same. Of the 367 articles, we found 60 met the first criteria, 65 met the second criteria, and 24 of these met both criteria, for a total of 77 articles remaining to be assessed in Step 3.

5.3 SLR Step 3: Assessing the quality of studies

For assessing the quality of the studies, we examined all remaining 77 articles more deeply to determine if their content contributed to the building of a modern taxonomy and whether or not the study was of high-quality. For heterogeneity and suitability, we determined whether each article discussed taxonomies in relation to TLEs. For quality, we examined the publication venue to determine if it was of acceptable quality (e.g. ACM digital library, IEEE digital library, or other journals or publication venues that have a formal review process).

We examined 10 of the 77 articles together, then divided the remaining to assess independently. After assessing them independently, together we discussed any discrepancies or questions that we had regarding our own set. Once the discrepancies were resolved, we identified 19 articles that qualified for inclusion in the next step [6, 28, 31, 33, 35, 40, 42, 43, 50, 54–56, 60–63, 65, 74, 75].

5.3.1 Step3A: Ancestry Searches. Before proceeding with Step 4, we reflected on the 19 articles and questioned whether or not this represented a complete set. For example, missing from these 19 was the recent work by Malmi et al. [41], which is an important work to consider. Likewise, Horn and Bers (2019) [32] report on Tangible Computing was also missing from the list. In addition, the authors were aware of two evaluation instruments measuring the K-12 computing education enacted curriculum that use classifiers for TLEs [4, 21] that were relevant to our study.

We then performed an ancestry search to review the references of all 19 articles plus Malmi et al.'s and Horn and Bers' work, and determine if there were additional articles from these that could be relevant to our study. Upon review, we found an additional 13 articles that were added to our list of qualifying articles [3, 12–14, 18, 30, 47, 48, 57–59, 64, 66].[1] Thus, we added an additional 17 articles to be reviewed in Step 4, for a total of 36 articles.

5.4 SLR Step 4: Summarizing the evidence

Of the 36 articles we found through this SLR, we examined each to determine what each article offered with respect to taxonomy, classification, or ontology of TLEs for computing education. The original categories as defined by the authors of these articles are reported in Tables 1 and 2 for ease of reading and referencing, and we discuss the findings in Section 5.5.

During this step, we eliminated Louca and Zacharia [40] because their classification of languages was in only two categories: textual and graphical, without further categorizations. We felt that those categorizations of languages were well represented by many of the other taxonomies and we did not include their work in the tables. Myers has two papers in the SLR results [47, 48] that both present the same taxonomy. The Myers taxonomy is only listed once in the summary table. The same situation occurs for Rongas and Kalviainen [60, 61], where again there were two papers uncovered in the SLR that present the same taxonomy, leaving 33 distinct taxonomies summarized by the tables.

5.5 SLR Step 5: Interpreting the findings

Step 5 of the systematic literature review is named "Interpreting the findings", however, it actually calls for the SLR researchers to determine whether or not the overall summary (provided in Step 4) can be trusted. For our SLR, each step was completed in a thorough manner, with reflection and discussion during each step. Heterogeneity was achieved throughout proper implementation of each of the steps, and we have noted when and why we have included (as well as excluded) articles. Through regular discussions throughout the entire process and careful adherence to procedures of the SLR, we conclude that any biases regarding the publications are limited in scope. We explored limitations of our initial assessment of the quality of the studies, and added an additional substep to perform an ancestry search. We also added additional recent materials that we were familiar with that were not on the list and performed an ancestry on these materials as well. We further ensured that each article or other material reviewed was of high quality, with many of the articles published in ACM and IEEE journals and conferences.

Therefore, we have a strong level of assurance that the literature referenced in Section 5.4 contains a thorough, heterogeneous set of articles that are ready for further analysis for building a taxonomy.

6 BUILDING THE TAXONOMY

To build the taxonomy, we worked together and took the results of the 136 categories defined in the 33 distinct taxonomies from the SLR (those shown in Tables 1 and 2) and placed each of the categories in separate rows in a spreadsheet. We then examined each individually to determine if there was overlap among the categories. In many cases, we needed to search for the context of the taxonomy from the original article, considering its age and the terminology used which varied across studies. In other cases, we had to determine if they were trying to define the same category as another article's, but just named differently.

Upon completion, we had defined 66 new categories that captured all 136 original categories, and we added these 66 to a new worksheet. Studying these, we saw that some categories were system features (e.g., cost, ease of use, supports collaborative learning), while others were oriented toward common descriptors of programming environments (e.g., block-based programming, text-based programming). To further group these categories and eliminate those that were only tangentially related to our goals, we borrowed from Burnett and Baker's arrangement of five categories of visual programming languages [14]: Taxonomy, Features, Implementation Issues, Purpose, and Theory. This gave us a way to clarify whether or not each of 66 categories could be represented in a new taxonomy or perhaps another category (e.g., implementation issues such as which operating system it needed to operate). This grouping resulted in 27 categories classified as actual TLE Taxonomy (see Table 3), while others were better suited for one of the other Burnett and Baker categories.

Our next step was to examine the 27 categories identified as Taxonomy and determine similarities and differences. Our goal was to see if there were natural groups among the categories. Working together through a lengthy process of discussion and debate, our work culminated in two major components: the *Engagement & Empowerment Medium* and the *Digital Programming Environment Dimensions*. We immediately saw that the groupings derived from these 33 articles in part reflected work performed by Kelleher and Pausch [35], though interpreted differently. The taxonomy is presented in the next section (7) and further discussion of these differences follows in Section 8.

7 PROPOSED TAXONOMY

As mentioned in section 6, we found two major groups through our analysis, *Engagement & Empowerment Medium* (hereafter referred to as *E&E Medium*) and *Digital Programming Environment Dimensions* (hereafter referred to as *Dimensions*). The taxonomy for the *E&E Medium* group contains Non-programming Environments and Programming Environments (see Figure 1). Non-programming environments are defined as CS/Computational Thinking (CT)-Specific or Non-CS/CT Specific, and each of these can have subcategories of Digital or Non-digital (see Table 4). Programming environments can be defined as General Purpose, Device, Simulation, Digital Media, Data Manipulation, Smart Blocks, Visualization of Computing Processes and Finite State Machine (see Table 5).

Dimensions consist of the dimensions of the environments and are represented orthogonal to the *E&E Medium*. *Dimensions* consist of Automated Assessment and Submission Tools, Gamified Systems, Integrated Development Environments, Teaching Systems, and Sandbox Systems (see Table 6).

We present these two groupings across Tables 4, 5, and 6, including definitions and examples. To ensure the taxonomy is relevant

[1] Three books that appeared to contain information on visual programming languages [25, 26, 29] were also cited by other studies, but we were unable to secure copies for this study and did not include these in our analysis.

Table 1: Summarizing the evidence, presented by ascending years, 1986-2005.

Author (Year) [Ref]	Subject of Taxonomy	Main categories defined
Myers (1986,1990) [47, 48]	Visual Programming and Program Visualization	(1) Taxonomy of programming systems: Visual programming or not, Example-based programming or not, Interpretive or compiled (orthogonal criteria) (2) Classification of Specification Technique: Textual languages, Flowcharts, Flowchart derivatives, Petri nets, Data flow graphs, Directed graphs, Graph derivatives, Matrices, Jigsaw puzzle pieces, Forms, Iconic sentences, Spreadsheets, Demonstrational, None (3) Taxonomy of Program Visualization Systems: Code, Data, Algorithm
Dart et al. (1987) [18]	Software Development Environments	Language-centered; Structure-oriented; Toolkit; Method-based
Brown (1988) [12]	Algorithm Animation Displays	Content; Persistence; Transformation
Perry et al. (1988) [57]	Software Development Environments	Individual; Family; City; State
Singh (1990) [64]	Graphical Support for Programming	Programming; Visualization
Stasko et al. (1992) [66]	Software Visualizations	Aspect; Abstractness; Animation; Automation
Mancoridis (1993) [42]	Software Development Environments	Scale; Genericity; Integration; History
Price et al. (1993) [58]	Software Visualization	Scope; Content; Form; Interaction; Effectiveness
Roman et al. (1993) [59]	Program Visualization Systems	Scope; Abstraction; Specification Method; Interface; Presentation
Burnett (1994) [14]	Visual Programming Languages	Environments and Tools for VPLs; Language Classifications; Language Features; Language Implementation Issues; Language Purpose; Theory of VPLs
Green et al. (1996) [30]	Cognitive Dimensions of Visual Programming Environments	Abstraction gradient; Closeness of mapping; Consistency; Diffuseness; Error-proneness; Hard mental operations; Hidden dependencies; Premature commitment; Progressive evaluation; Role-expressiveness; Secondary notation; Viscosity; Visibility
Naps et al. (2002) [50]	Learner Engagement with Visualization Technology	No viewing; Viewing; Responding; Changing; Constructing; Presenting
Yehezkel (2002) [75]	Computer Architecture Visualizations	Scope of Operation; Content Modeling; Presentation Methods; Activity Styles
Wright et al. (2003) [74]	Programming Environments	Level one: How many languages exist in environment; Level Two: How user manipulates the languages: Reading programs, Writing programs, Watching programs run; Also identified gulfs where the three tasks are supported differently: expression, representation, visualization
Beckwith et al. (2004) [6]	Gender Difference Relevant to End User Programming	Confidence; Support; Motivation
Rongas et al. (2004) [60, 61]	Computerized Learning Tools for Introductory Programming Courses	Support for Problem Solving; Syntax; Semantics; Pragmatics; Support for Self-Study; Support for Teacher; Suitablity for a Newcomer
Gómez-Albarrán (2005) [28]	Tools to Support the Teaching and Learning of Programming	Tools with a Reduced Development Environment; Example-based Environments; Tools Based on Visualization; Simulation Environments
Ihantola et al. (2005) [33]	Ability of Systems to Create Algorithm Visualizations Effortlessly	Scope; Integrability; Interaction: producer-system interaction, visualization-consumer interaction
Kelleher et al. (2005) [35]	Programming Environments and Languages for Novice Programmers	Teaching Systems: Mechanics of Programming, Make Programming Concrete, Learning Support; Empowering Systems: Mechanics of Programming, Activities Enhanced by Programming

Table 2: Summarizing the evidence, presented by ascending years, 2006-2020.

Author (Year) [Ref]	Subject of Taxonomy	Main categories defined
Hernán-Losada et al. (2006) [31]	Creating Tools Mapped to Bloom's Taxonomy	Knowledge-quizzes; Comprehension-translate, predict, but no methods; Application-problems posed can be solved directly by applying given methods; Analysis-decompose a program into its constituent parts, examine its relations, studying different possibilities; Synthesis-tools to solve problems by means of any solution; Evaluation-tools must be able to compare different alternatives
Parker et al. (2006) [55]	Selecting a Language for Introductory Programming Courses	Software Cost; Programming Language Acceptance in Academia; Programming Language Industry Penetration; Software Characteristics; Student-Friendly Features; Language Pedagogical Features; Language Intent; Language Design; Language Paradigm; Language Support and Required Training; Student Experience
Pears et al. (2007) [56]	Research on Tools in Introductory Programming Literature	Visualization Tools; Automated Assessment Tools; Programming Environments; Programming Support Tools; Microworlds; Other Tools
Mason et al. (2012) [43]	Programming Languages	Procedural; Functional; Object-oriented; Mixture
Orehovački et al. (2012) [54]	Web 2.0 Applications with Educational Potential	Function of Application; Cognitive Processes from Revised Bloom's Taxonomy
Sorva et al. (2013) [65]	Program Visualization Systems for Introductory Programming Education	Extends Naps et. al taxonomy [50] to include another dimension: Content Ownership: Given Content, Own Cases, Modified Content, Own Content
Brusilovsky et al. (2014) [13]	Smart Learning Content	Dimensions; Pedagogical Content
Saito et al. (2017) [62]	Programming Learning Tools for Children	Style of Programming; Programming Constructs; Representation of Code; Construction of Programs; Support to understand programs; Designed accessible languages; Game elements; Supporting language; Operating Environments; Interface; Experience
Angel-Fernandez et al. (2018) [3]	Educational Robotics	Robotics as Learning Object; Robotics as Learning Tool; Robots as Learning Aids
Falkner et. al. (2019) [21]	Programming Environments	"Unplugged"/non-digital programming; Symbolic (text-free) visual programming; Block-based visual programming; Hybrid visual to junior text programming; General Purpose/Textbased Programming;
Horn et al. (2019) [32]	Tangible Computing	Smart Block Languages; Tangible Demonstration Languages; Externally Compiled Languages
Malmi et al. (2019) [41]	Tools	Software Applications; Web Application Providing some Service; Software Frameworks; Definition Language
Scaradozzi et al. (2019) [63]	Tools, Experiences, Assessments for Educational Robotics	(1) Experiences: Learning Evironment; Impact on Curriculum; Integration of the Robotic Tool; Evaluation of Activities (2) Educational Robotics Tools: Age; Programming Language; Assembly feature; Robot's Environment
Anwar et al. (2020) [4]	Programming Environments	"Unplugged"/non-digital Programming Using Human or Tangible Objects or Manipulatives; General Purpose/Text-based Programming without Computers; General Purpose/Text-based Programming with Computers; Visual Programming;

Table 3: Our 66 refined categories grouped according to Burnett and Baker's classification [14]. Only 27 were part of an actual taxonomy, while others fell into the definitions for system features, implementation specifications, purpose, or theories.

Category	N	Examples
Taxonomy	27	Unplugged activities, Text-based environments, Robotics
Feature	31	Ease of use, Step-wise control of visualization, Languages supported
Implementation	5	Operating systems, Cost
Purpose	2	Language purpose, Scope
Theory	4	Supports learner, Supports problem-solving

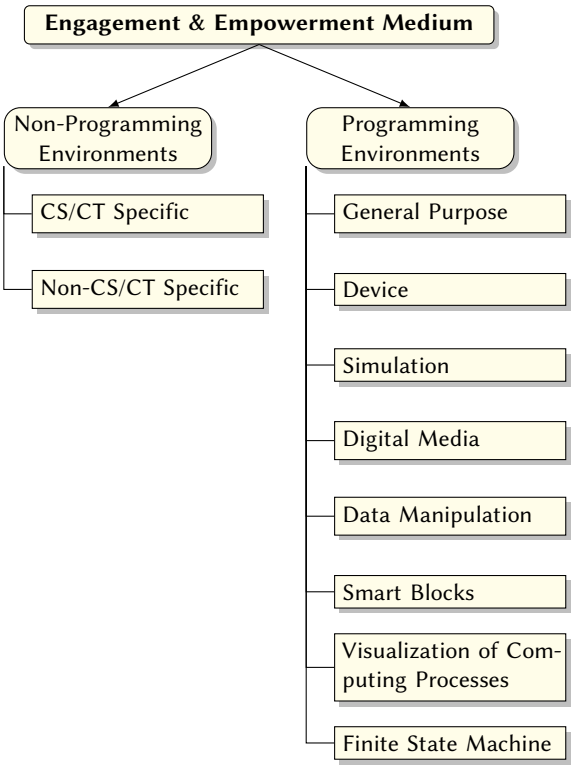

Figure 1: Engagement & Empowerment Medium Hierarchy. Non-programming environments are described in Table 4 while programming environments are described in Table 5. CT refers to Computational Thinking.

for K-12 CS education, the examples primarily come from the publicly available dataset at https://csedresearch.org, supplemented where needed. These groups are further discussed in section 8.

8 DISCUSSION

While we also found two main groupings (*E&E Medium* and *Dimensions*) similar to the work of Kelleher and Pausch [35], there are some distinct differences in our eventual taxonomy categorizations. First, teaching systems are a characteristic that programming environments can have. That is, the environment could have been designed explicitly for teaching. However, there are many tools that are used in the teaching of computing that were not designed for this purpose and it was important to reflect these in our taxonomy. The name and themes represented in *E&E Medium* was influenced by this earlier work [35], but has a key difference in that it includes the possibility that all systems can be engaging and empowering within their own domains and context and that much of the empowerment actually comes externally from the curriculum and classroom in which the system is presented. Lastly, we do not dictate, nor require, that each system fit squarely into one and only one categorization in the taxonomy. It is possible for a system to fit into multiple categories.

Several items in Programming Environments (Table 5) warrant further elaboration. When looking at the subcategories (Block-based with symbols, Block-based with text, Hybrid, and Text-based), there are fewer examples of Block-based with symbols or the Hybrid medium. The Block-based with symbol examples have often been designed for pre-reading age students, which, prior to a few years ago, were not a population previously served by computing education. With the shift into K-12 and with the introduction and proliferation of block-based with text environments and tools [45], we see a need developing for bridge (hybrid) languages to shift students from a blocks-based environments and to traditional text-based coding environments that still dominate professional software development. These areas seem to be poised to grow in the near future to fill these growing needs.

In terms of subcategories, Device is novel. The intent of this category is to capture environments for which the target of the computation may not be a traditional computer with monitor and keyboard. Examples of this include environments used to create programs to run on robots of various types, mobile phones, and circuit and LED boards. This is another area where there seems to be growth potential for new environments that create code to be run on external devices or embedded systems.

While we have currently separated the Simulation and Digital Media subcategories, this distinction may be artificial and not needed. There is evidence from the SLR that a category of environments that deal with simulations and microworlds should exist [28, 35, 56]. However, many of the taxonomies found in our SLR were created before digital media manipulation was possible to the degree it is today. While games could arguably be considered simulations, environments specifically designed to manipulate media as computation are not necessarily simulations, but all of these environments may be able to be combined into a single category with a comprehensive label. In our future work to vet the taxonomy, we will need to explore if this distinction is still necessary.

Data Manipulation programming environments are those that target the manipulation of data. As the field of data science continues to grow, and pushes are being made to include data science within K-12 education [1] as well as post-secondary education [17],

Table 4: Engagement & Empowerment Medium -> Non-programming environments used to teach computing. All examples come from the https://csedresearch.org dataset, 2012-2019, unless denoted by *. CT refers to Computational Thinking.

Category	Description	Digital	Non-Digital
CS/CT-Specific	TLEs specifically created to support learning of computing and computing concepts	Robot NAO, E-textiles, Raspberry Pi	Potato Pirates*, Robot Turtles*
Non-CS/CT-Specific	TLEs created to support general learning or for general usage	New York Times Mapping America, Google Drive, Youtube	Barrel of Monkeys*, Beads*, Buttons*

Table 5: Engagement & Empowerment Medium -> Programming Environments. All examples are from the https://csedresearch.org dataset, 2012-2019, unless denoted by *. See Discussion (Section 8) for more information on categories denoted by **.

Category	Provides capability for user to:	Block-based Symbolic	Block-based Text	Hybrid	Text-based
General-Purpose	Create any type of program.	Scratch Jr	Scratch	Pencil Code, Stride*	BlueJ*, Eclipse*
Device	Create programs that target a specific device or system.	Lego Mindstorms	AppInventor	-	Arduino
Simulation	Program a simulation or within a microworld environment.	Lightbot*	TurtleArt	Alice, Stride*	Greenfoot, Karel*
Digital Media	Produce media including graphics, sound, or games.	Kodu	AgentSheets, GameMaker	-	GameMaker, Earsketch, Processing
Data Manipulation	Manipulate data computationally (data science).	-	VisComposer*, Tableau*	-	R*
Physical Smart Blocks	Create a program using a programming interface by assembling physical elements.	Electronic Blocks, roBlocks, Bee-Bot*	KIBO*	N/A	N/A
Visualization of Computing Processes**	Create visualizations of computing processes or algorithms.	-	-	-	Jeliot*, Alvis*
Finite State Machine**	Create finite state machines as computational engines.	-	jFast*	-	-

we anticipate that future environments may be created (including those that are symbolic block-based) to meet this need.

Physical Smart Blocks are a relatively new subcategory where physical objects, in most cases physical blocks, are the computation devices [32]. This is distinct from the idea of a robot or external device onto which a program is loaded. Instead, it is the assembling of these devices themselves that creates the computation. Because of the nature of these types of environments, they would seem unlikely to be categorized as hybrid or text-based and we have marked them as N/A in the table to indicate this.

The last two subcategories, Visualization of Computing Processes and Finite State Machines, represent categories where modern examples were scarce in the K-12 literature. This was not surprising given that these concepts are more advanced computing concepts that would often be covered in the post-secondary level. However, there is considerable historical context for their inclusion that is derived from the SLR [12, 30, 33, 47, 48, 50, 58, 59, 62, 65, 66, 75]. More research is needed to find the current state of these environments. We leave open the possibility that, while historically these environments were created and may have been important, their importance and significance may not be what it once was and these categories may need to be removed–particularly, if these environments were not currently active, used, or maintained.

Table 6: Digital Programming Environment Dimensions.
All examples come from the https://csedresearch.org dataset, 2012-2019, unless denoted by *.

Dimension	Description	Example
Automated Assessment and Submission Tools	Provide ability to submit and give feedback and scoring/grading for programs	Web-CAT*, Gradescope*
Gamified Systems	Provide gamified elements to the learning process (e.g., points, badges, leaderboards)	Codespells, Gidget
Integrated Development Environment	Provides programming language(s), code editing, compilation, debugging, runtime diagnostics	Eclipse*, Processing, Scratch
Teaching Systems	"Simple programming tools that provide novice programmers with exposure to some of the fundamental aspects of the programming process" [35, p. 84]	Lego Mindstorms, Kodu
Sandbox Systems	Provide open-ended spaces for exploration and learning	Minecraft, Alice

Turning our attention to the non-programming environments (Table 4), we acknowledge other environments that complement programming environments and can be used to teach computing concepts (e.g., computational thinking) and to engage students. For most, the environment itself would be ineffective at teaching computing concepts. It is the environment coupled with a structured curriculum that becomes a conduit for students to learn computing concepts. For example, it is not uncommon for teachers to use physical manipulatives (e.g., beads, buttons, toys, divided bins, egg cartons or games) in their classrooms to help students understand computing concepts or algorithms. The tool itself is not a programming environment, but rather the curriculum built around the tool empowers the learning of computing.

Dimensions (Table 6) should be viewed as orthogonal to the categorizations in Table 5. They represent broad features that may be available across many programming environments to varying degrees. We placed the notion of Teaching Systems as defined by Kelleher and Pausch in *Dimensions* [35]. Any of the programming environments that are categorized in Table 5 could be Teaching Systems if they were designed with instruction and learning at the forefront (e.g., reduced environment features [28]). However, programming environments may not have strong Teaching Systems features if they were originally designed for another purpose, though still used to teach computing (e.g., using Emacs with a C++ compiler running from a command line). There is certainly room for additional research into these *Dimensions* to describe the many features that each contain and to ensure that this is a complete set.

One key feature that is conspicuously missing from our tables and categories is the notion of programming languages. This was an interesting development for us during the taxonomy creation process. We decided that support for a language or paradigm is actually a feature of an environment, but one of only many dimensions that teachers may be looking for in an environment to support pedagogy. In our SLR, we identified 31 such features (Table 3) that could be important, including reduced language features [28], support for collaboration [54], version control [12], debugging [28, 62], and other user interface issues. While support for a particular language or paradigm can be an important feature, it is also the case that many modern environments support multiple languages. This trend is likely to continue to allow for wider adoption in more varied circumstances. For example, in our Device subcategory (Table 5), many of the robots can be programmed in multiple languages and the executable code loaded onto and executed on the robot.

9 CONCLUSION AND FUTURE WORK

With the completion of the first phase of our taxonomy, our next steps mirror steps 4 through 7 of Whittaker and Breininger's taxonomy building process [73] (Section 3) and are directed towards gathering evidence of validity and reliability. In the case of validating the Engineering Education Research taxonomy, Finelli et al. were able to verify that researchers could assign sets of data to the same terms with a high degree of agreement, that the taxonomy was complete and that there was an even distribution of the data across the proposed terms [22]. This is similar to a Closed Card Sorting exercise in UX design [53, 68], in which participants are given a set of items (typically on individual cards) and predefined categories (like our taxonomy categories), and are asked to place each item in a category. This provides information on the level of agreement in which participants place the cards in the same categories and whether all of the items can be placed.

Borrowing from these methods, our next stage of this process includes the following steps:

- Gather evidence for face validity by reviewing the taxonomy with users and subject matter experts,
- Refine the taxonomy based on feedback,
- Gather evidence of content validity by administering a Closed Card Sorting exercise,
- Revise groupings based on results of the Closed Card Sorting exercise, and if needed administer a second Closed Card Sorting exercise, and
- Manage and maintain the taxonomy, including integrating the taxonomy where possible.

Through these measures, we can ensure that the wider computer science education research community has input into the terms and definitions used in the taxonomy and that it further meets the needs of the community.

10 ACKNOWLEDGEMENTS

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. 1625005, 1625335, 1757402, 1745199, and 1933671.

REFERENCES

[1] ACM Computer Science Teachers Association (CSTA). 2017. Computer science standards. *Computer Science Teachers Association* (2017).

[2] Amal Al-Abri, Yassine Jamoussi, Naoufel Kraiem, and Zuhoor Al-Khanjari. 2017. Comprehensive classification of collaboration approaches in E-learning. *Telematics and Informatics* 34, 6 (2017), 878–893.

[3] Julian M Angel-Fernandez and Markus Vincze. 2018. Towards a definition of educational robotics. In *Austrian Robotics Workshop 2018*. 37.

[4] Tehreem Anwar, Arturo Jimenez, Arsalan Najeeb, Bishakha Upadhyaya, and Monica M. McGill. 2020. Exploring the Enacted Computing Curriculum inK-12 Schools in South Asia:Bangladesh, Nepal, Pakistan, and Sri Lanka. In *Proceedings of the International Computer Education Research conference*. Association for Computing Machinery, New York, NY, USA.

[5] Henri Barki, Suzanne Rivard, and Jean Talbot. 1988. An information systems keyword classification scheme. *MIS quarterly* (1988), 299–322.

[6] Laura Beckwith and Margaret Burnett. 2004. Gender: An important factor in end-user programming environments?. In *2004 IEEE symposium on visual languages-human centric computing*. IEEE, 107–114.

[7] Jyoti Belur, Lisa Tompson, Amy Thornton, and Miranda Simon. 2018. Inter-rater reliability in systematic review methodology: exploring variation in coder decision-making. *Sociological methods & research* (2018), 0049124118799372.

[8] John B Biggs and Kevin F Collis. 1982. *Evaluation the quality of learning: the SOLO taxonomy (structure of the observed learning outcome)*. Academic Press.

[9] S Bloom Benjamin and David R Krathwohl. 1956. Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive Domain. *New York: David McKay* (1956).

[10] Luca Botturi, Michael Derntl, Eddy Boot, and Kathrin Figl. 2006. A classification framework for educational modeling languages in instructional design. In *6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*.

[11] Matt Bower. 2008. A taxonomy of task types in computing. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*. 281–285.

[12] Marc H Brown. 1988. Perspectives on algorithm animation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 33–38.

[13] Peter Brusilovsky, Stephen Edwards, Amruth Kumar, Lauri Malmi, Luciana Benotti, Duane Buck, Petri Ihantola, Rikki Prince, Teemu Sirkiä, Sergey Sosnovsky, et al. 2014. Increasing adoption of smart learning content for computer science education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*. 31–57.

[14] Margaret M. Burnett and Marla J. Baker. 1994. A classification system for visual programming languages. *Journal of Visual Languages and Computing* 5, 3 (1994), 287–300.

[15] Charles H Calisher. 2007. Taxonomy: What's in a name? Doesn't a rose by any other name smell as sweet? *Croatian medical journal* 48, 2 (2007), 268. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2080517/

[16] David D Chen. 2003. A classification system for metaphors about teaching. *Journal of Physical Education, Recreation & Dance* 74, 2 (2003), 24–31.

[17] Andrea Danyluk, Paul Leidig, Lillian Cassel, and Christian Servin. 2019. ACM Task Force on Data Science Education: Draft Report and Opportunity for Feedback. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 496–497.

[18] S. Dart, R. Ellison, A. Habermann, and P. Feiler. 1987. Software Development Environments. *Computer* 20, 11 (nov 1987), 18–28. https://doi.org/10.1109/MC.1987.1663413

[19] Fadi P Deek and James A McHugh. 1998. A survey and critical analysis of tools for learning programming. *Computer Science Education* 8, 2 (1998), 130–178.

[20] Melvil Dewey. 1876. *Decimal classification and relative index...*

[21] Katrina Falkner, Sue Sentance, Rebecca Vivian, Sarah Barksdale, Leonard Busuttil, Elizabeth Cole, Christine Liebe, Francesco Maiorana, Monica M. McGill, and Keith Quille. 2019. An International Study Piloting the MEasuring TeacheR Enacted Computing Curriculum (METRECC) Instrument. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 111–142. https://doi.org/10.1145/3344429.3372505

[22] Cynthia J Finelli, Maura Borrego, and Golnoosh Rasoulifar. 2015. Development of a taxonomy of keywords for engineering education research. *IEEE Transactions on Education* 58, 4 (2015), 219–241.

[23] Ursula Fuller, Colin G Johnson, Tuukka Ahoniemi, Diana Cukierman, Isidoro Hernán-Losada, Jana Jackova, Essi Lahtinen, Tracy L Lewis, Donna McGee Thompson, Charles Riedesel, et al. 2007. Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin* 39, 4 (2007), 152–170.

[24] Tatiana Gavrilova, Rosta Farzan, and Peter Brusilovsky. 2005. One practical algorithm of creating teaching ontologies. In *12th International Network-Based Education Conference NBE*. Citeseer, 29–37.

[25] Ephraim P Glinert. 1990. *Visual programming environments: Applications and issues*. IEEE Computer Society Press.

[26] Ephraim P Glinert. 1990. *Visual programming environments: paradigms and systems*. IEEE Computer Society Press.

[27] Anabela Gomes and António Mendes. 2009. Bloom's taxonomy based approach to learn basic programming. In *EdMedia+ Innovate Learning*. Association for the Advancement of Computing in Education (AACE), 2547–2554.

[28] Mercedes Gómez-Albarrán. 2005. The teaching and learning of programming: a survey of supporting software tools. *Comput. J.* 48, 2 (2005), 130–144.

[29] TRG Green. 1991. NC Shu,£ 25.95, Visual Programming, Van Nostrand Reinhold, New York (1988), 315 pp, ISBN: 0-442-28014-9.

[30] Thomas R. G. Green and Marian Petre. 1996. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing* 7, 2 (1996), 131–174.

[31] Isidoro Hernán-Losada, JÁ Velázquez-Iturbide, and CA Lázaro-Carrascosa. 2006. Programming learning tools based on Bloom's taxonomy: proposal and accomplishments. In *Proceedings of the 8th International Symposium of Computers in Education (SIIE 2006).(Leon, Spain*. 325–334.

[32] M. Horn and M. Bers. 2019. Tangible Computing. In *The Cambridge Handbook of Computing Education Research*, S.A. Fincher and A.V. Robins (Eds.). Cambridge University Press, Cambridge, UK, Chapter 22, 663–678.

[33] Petri Ihantola, Ville Karavirta, Ari Korhonen, and Jussi Nikander. 2005. Taxonomy of effortless creation of algorithm visualizations. In *Proceedings of the first international workshop on Computing education research*. 123–133.

[34] Cruz Izu, Amali Weerasinghe, and Cheryl Pope. 2016. A study of code design skills in novice programmers using the SOLO taxonomy. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. 251–259.

[35] Caitlin Kelleher and Randy Pausch. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)* 37, 2 (2005), 83–137.

[36] Khalid S Khan, Regina Kunz, Jos Kleijnen, and Gerd Antes. 2003. Five steps to conducting a systematic review. *Journal of the royal society of medicine* 96, 3 (2003), 118–121.

[37] David R Krathwohl. 2002. A revision of Bloom's taxonomy: An overview. *Theory into practice* 41, 4 (2002), 212–218.

[38] Adidah Lajis, Haidawati Md Nasir, and Normaziah A Aziz. 2018. Proposed assessment framework based on bloom taxonomy cognitive competency: Introduction to programming. In *Proceedings of the 2018 7th International Conference on Software and Computer Applications*. 97–101.

[39] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L Whalley, and Christine Prasad. 2006. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin* 38, 3 (2006), 118–122.

[40] Loucas T Louca and Zacharia C Zacharia. 2008. The use of computer-based programming environments as computer modelling tools in early science education: The cases of textual and graphical program languages. *International Journal of Science Education* 30, 3 (2008), 287–323.

[41] Lauri Malmi, Ian Utting, and Amy Ko. 2019. Tools and environments. In *The Cambridge Handbook of Computing Education Research*, S.A. Fincher and A.V. Robins (Eds.). Cambridge University Press, Cambridge, UK, Chapter 21, 639–662.

[42] Spiros Mancoridis. 1993. A multi-dimensional taxonomy of software development environments. In *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering-Volume 1*. IBM Press, 581–594.

[43] Raina Mason, Graham Cooper, and Michael de Raadt. 2012. Trends in introductory programming courses in Australian universities: languages, environments and pedagogy. In *Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123*. 33–42.

[44] Matchcraft Sales & Training. 2015. What is Taxonomy (And Why is it So Important to Search Marketing)? https://www.matchcraft.com/what-is-taxonomy-and-why-is-it-so-important-to-search-marketing/

[45] Monica M McGill and Adrienne Decker. 2020. Tools, Languages, and Environments Used in Primary and Secondary Computing Education. In *Proceedings of the 25th annual conference on Innovation and technology in computer science education*. ACM.

[46] Greg Moody, Taylor Wells, and Paul Benjamin Lowry. 2007. The Interactive Digital Entertainment (IDE) unification framework: creating a taxonomy of IDE and lifestyle computing. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. IEEE, 160a–160a.

[47] Brad A Myers. 1986. Visual programming, programming by example, and program visualization: a taxonomy. *ACM SIGCHI Bulletin* 17, 4 (1986), 59–66.

[48] Brad A Myers. 1990. Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing* 1, 1 (1990), 97–123.

[49] Rafi Nachmias and Inbal Tuvi. 2001. Taxonomy of scientifically oriented educational websites. *Journal of Science Education and Technology* 10, 1 (2001), 93–104.

[50] Thomas L Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, et al. 2002. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education*. 131–152.

[51] National Center for Biotechnology Information, U.S. National Library of Medicine. 2020. The Taxonomy Database. https://www.ncbi.nlm.nih.gov/taxonomy

[52] Robert Neches, Richard E Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R Swartout. 1991. Enabling technology for knowledge sharing. *AI magazine* 12, 3 (1991), 36–36.

[53] Optimal Workshop. 2020. Card Sorting 101: Your guide to creating and running an effective card sort. https://www.optimalworkshop.com/learn/101s/card-sorting/

[54] Tihomir Orehovački, Goran Bubaš, and Andreja Kovačić. 2012. Taxonomy of Web 2.0 applications with educational potential. *Transformation in teaching: Social media strategies in higher education* (2012), 43–72.

[55] Kevin R Parker, Joseph T Chao, Thomas A Ottaway, and Jane Chang. 2006. A formal language selection process for introductory programming courses. *Journal of Information Technology Education: Research* 5, 1 (2006), 133–151.

[56] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. 2007. A survey of literature on the teaching of introductory programming. In *Working group reports on ITiCSE on Innovation and technology in computer science education*. 204–223.

[57] Dewayne E Perry and Gail E Kaiser. 1988. Models of software development environments. In *Proceedings of the 10th international conference on Software engineering*. IEEE Computer Society Press, 60–68.

[58] Blaine A Price, Ronald M Baecker, and Ian S Small. 1993. A principled taxonomy of software visualization. *Journal of Visual Languages & Computing* 4, 3 (1993), 211–266.

[59] G-C Roman and Kenneth C Cox. 1993. A taxonomy of program visualization systems. *Computer* 26, 12 (1993), 11–24.

[60] Timo Rongas, Arto Kaarna, and Heikki Kalviainen. 2004. Classification of computerized learning tools for introductory programming courses: learning approach. In *IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings.* IEEE, 678–680.

[61] Timo Rongas, Arto Kaarna, and Heikki Kälviäinen. 2004. *Classification of tools for use in introductory programming courses.* Lappeenranta University of Technology.

[62] Daisuke Saito, Ayana Sasaki, Hironori Washizaki, Yoshiaki Fukazawa, and Yusuke Muto. 2017. Program learning for beginners: survey and taxonomy of programming learning tools. In *2017 IEEE 9th International Conference on Engineering Education (ICEED).* IEEE, 137–142.

[63] David Scaradozzi, Laura Screpanti, and Lorenzo Cesaretti. 2019. Towards a definition of educational robotics: a classification of tools, experiences and assessments. In *Smart Learning with Educational Robotics*. Springer, 63–92.

[64] Gurminder Singh. 1990. Graphical support for programming: A survey and taxonomy. In *CG International'90*. Springer, 331–359.

[65] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)* 13, 4 (2013), 1–64.

[66] John T Stasko and Charles Patterson. 1992. Understanding and characterizing software visualization systems. In *Proceedings IEEE Workshop on Visual Languages*. IEEE, 3–10.

[67] B Swartout, R Patil, K Knight, and T Russ. 1997. Towards distributed use of large-scale ontologies. Spring Symposium Series on Ontological Engineering. (1997).

[68] U.S. Department of Health & Human Services. 2020. Card Sorting. https://www.usability.gov/how-to-and-tools/methods/card-sorting.html

[69] Michael S. Vasta. 2020. Product Taxonomy: Categorizing Your Website Hierarchy to Increase Sales. https://www.bigcommerce.com/blog/product-taxonomy/

[70] Euripides Vrachnos and Athanassios Jimoyiannis. 2017. Secondary education students' difficulties in algorithmic problems with arrays: An analysis using the SOLO taxonomy. *Themes in Science and Technology Education* 10, 1 (2017), 31–52.

[71] Zach Wahl. 2013. Taxonomy Consulting and the Importance of UX Design. https://enterprise-knowledge.com/taxonomy-consulting-and-the-importance-of-ux-design/

[72] Betsy Walli. 2014. Taxonomy 101: The Basics and Getting Started with Taxonomies. https://www.kmworld.com/Articles/ReadArticle.aspx?ArticleID=98787

[73] Mary Whittaker and Kathryn Breininger. 2008. Taxonomy development for knowledge management. In *74th general conference and council of the world library and information, Quebec, Canada.*

[74] Tim Wright and Andy Cockburn. 2003. A language and task-based taxonomy of programming environments. In *IEEE Symposium on Human Centric Computing Languages and Environments, 2003. Proceedings. 2003.* IEEE, 192–194.

[75] Cecile Yehezkel. 2002. A taxonomy of computer architecture visualizations. *ACM SIGCSE Bulletin* 34, 3 (2002), 101–105.