IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 22, 2020, accepted January 6, 2021, date of publication January 12, 2021, date of current version January 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051043

Characterizing Visual Programming Approaches for End-User Developers: A Systematic Review

MOHAMMAD AMIN KUHAIL[1], (Member, IEEE), SHAHBANO FAROOQ[2], RAWAD HAMMAD[3], AND MOHAMMED BAHJA[4]

[1]College of Technological Innovation, Zayed University, Abu Dhabi, United Arab Emirates
[2]University College, Zayed University, Abu Dhabi, United Arab Emirates
[3]Department of Computer Science and Digital Technologies, University of East London, London E16 2RD, U.K.
[4]School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K.

Corresponding author: Mohammad Amin Kuhail (mohammad.kuhail@zu.ac.ae)

This work was supported by the Research Incentive Fund under Grant R20051, and by the Zayed University, United Arab Emirates.

ABSTRACT Recently many researches have explored the potential of visual programming in robotics, the Internet of Things (IoT), and education. However, there is a lack of studies that analyze the recent evidence-based visual programming approaches that are applied in several domains. This study presents a systematic review to understand, compare, and reflect on recent visual programming approaches using twelve dimensions: visual programming classification, interaction style, target users, domain, platform, empirical evaluation type, test participants' type, number of test participants, test participants' programming skills, evaluation methods, evaluation measures, and accessibility of visual programming tools. The results show that most of the selected articles discussed tools that target IoT and education, while other fields such as data science, robotics are emerging. Further, most tools use abstractions to hide implementation details and use similar interaction styles. The predominant platforms for the tools are web and mobile, while desktop-based tools are on the decline. Only a few tools were evaluated with a formal experiment, whilst the remaining ones were evaluated with evaluation studies or informal feedback. Most tools were evaluated with students with little to no programming skills. There is a lack of emphasis on usability principles in the design stage of the tools. Additionally, only one of the tools was evaluated for expressiveness. Other areas for exploration include supporting end users throughout the life cycle of applications created with the tools, studying the impact of tutorials on improving learnability, and exploring the potential of machine learning to improve debugging solutions developed with visual programming.

INDEX TERMS Visual programming, end-user development, human-computer interaction, systematic literature review.

I. INTRODUCTION

An increasing number of software applications are being written by end users without formal software development training. This inspired large technology companies such as Microsoft [91] and Amazon [90] to invest in low-code development environments empowering end users to create web and mobile applications. According to the 2019 Q1 Forrester report, the low-code market will witness an annual growth rate of 40%, with spending forecast to reach $21.2 billion by 2022 [102].

End-User Development (EUD) has emerged as a field that is concerned with tools and activities allowing end users who are not professional software developers to write software applications [11]. This is promising as end users know their own domain and needs more than anyone else, and are often aware of specificities in their respective contexts. Further, as end users outnumber developers with professional software development training by a factor of 30-to-1, EUD enables a much larger pool of people to participate in software development [12].

A visual programming language (VPL), among other EUD techniques, allows end users to create a program by piecing together graphical elements rather than textually specifying them [9].

Traditionally, visual programming has been successfully used to help novices learn basics of programming by visualizing elements of a program. However, visual programming

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan Abid.

TABLE 1. An overview of the related review studies in terms of dimension coverage.

Article	Focus of the Survey	Survey Year	Survey Approach	No. of Articles	Dimensions Addressed											
					VPL Approach	Interaction Styles	Multiple Domains	Target Users	Platform	Empirical Evidence	Test Participant Types	Number of Test Participants	Skills of Test Participants	Evaluation Methods	Evaluation Measures	Tool Accessibility
[26]	Surveying End-user development approaches generally	2013	Informal	Not mentioned	×	✓	✓	✓	✓	×	×	×	×	×	×	×
[7]	Describing End-user development tools generally	2019	Systematic Review	21	×	✓	✓	×	✓	×	×	×	×	×	×	×
[29]	Surveying End-user development, end-user programming, and end-user software engineering approaches	2018	Systematic Mapping	165	✓	✓	✓	✓	✓	✓	×	✓	×	×	×	×
[8]	Review of End-user development tools aimed at social robots	2020	Systematic Review	16	✓	×	×	✓	✓	✓	✓	×	×	✓	×	✓
[10]	Surveying VPL tools aimed IoT applications	2017	Informal	13	×	×	×	×	✓	×	×	×	×	×	×	✓
This study	Characterizing VPL approaches aimed at end-user developers	2020	Systematic Review	30	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

✓ Fully addressed ✓ Partially Addressed. × Not Addressed

is increasingly being used by end users in various domains to create and tailor applications that are useful beyond the realm of education. For instance, VPLs are now being used in fields such as the Internet of Things (IoT) [3], [10], mobile application development [51], robotics [8], and Virtual/Augmented Reality [4].

A few review studies have been conducted recently aimed at analyzing and comparing different approaches to EUD including visual programming. Table 1 shows an overview of how these review studies compare with this study in several dimensions. The works found in [7], [26] and [29] analyzed a limited number of visual programming approaches and techniques since article retrieval mainly focused on EUD approaches in general as opposed to visual programming approaches. For example, the work found in [29] classified the articles based on broad EUD techniques that do not cover all VPL classifications, whilst the authors of [7] primarily investigated EUD with reference to EUD-related conferences and missed relevant visual programming journals. Other review studies such as [8] and [10] surveyed visual programming environments in specific domains such as robotics and IoT, thereby overlooking a broach view of VPL approaches applied in multiple domains beyond IoT and robotics.

Furthermore, as Table 1 shows, the existing review studies have barely touched on the empirical evidence, evaluation methods and measures that back up the validity of the visual programming tools.

Therefore, there is a need for a systematic identification of articles describing and analyzing approaches and visual programming techniques used in multiple domains and backed up by empirical evidence, in order to obtain an in-depth analysis and understanding of the visual programming research.

By systematically analyzing 30 articles presenting visual programming tools representing various approaches (block-based, diagram-based, form-based and icon-based), this study contributes: (1) an in-depth analysis of the visual programming approaches currently used to enable the creation of software applications used in several domains beyond the traditional ones (such as computer science education); (2) a characterization of the trends and technologies used for the development of visual programming tools; (3) an in-depth explanation of the empirical evidence used to back up the validity of the study, and (4) the discussion of open challenges and future research directions specific to visual programming tools.

This study will help the research community in the field of end-user development aiming at designing and evaluating visual programming tools. Such tools might adapt some ideas from the tools surveyed in this study, while addressing the discussed challenges and considering the suggested future research directions.

The rest of the article is organized as follows. It first gives a background of visual programming as a subset of EUD as well as its categories. Then, the article discusses the review studies in related areas. Thereafter, the study explains the methodology of this systematic literature review including the research questions. Subsequently, the study presents the results of answering the research questions by examining the

visual programming tools presented in the selected articles against 12 dimensions: classification of visual programming, interaction style, target users, domain, platform, empirical evidence, types of test participants, number of test participants, programming skills of test participants, evaluation methods, evaluation measures, and accessibility. Thereafter, challenges and future research directions are discussed. Finally, the key findings are presented in the conclusion.

II. BACKGROUND

Visual programming is a subset of the end-user development field (EUD). Other subsets of EUD include End-user programming (EUP), which mainly focuses on enabling end users to create their own programs [12], while EUD tackles the entire software life cycle including maintainability, and extensibility. End-user Software Engineering (EUSE) is another related research area which focuses on the quality attributes of the software developed by end users including reusability, security, and verifiability [77].

Visual programming refers to approaches and methods that use two-dimensional graphical elements to allow non-programmer end-users to create, extend, and customize software applications [9]. Visual programming languages (VPLs) are described by programming constructs and rules which are visually depicted [13].

There are two widely known taxonomies for visual programming languages: (1) Myers [1] classified visual programming languages by specification techniques. Certain categories in Myers' classification can be generalized into one category. As an example, data-flow graphs, directed graphs, and flowcharts can be considered diagrammatic VPLs. (2) Burnett and Baker [2], on the other hand, listed three broad subcategories under ''visual representations'' namely: diagrammatic languages, iconic languages, and languages based on static pictorial sequences. This classification, while highly useful, does not list the form-based VPLs mentioned in Myers' classification.

Combining the taxonomies developed by Myers [1] and Burnett and Baker [2], we divide VPLs into four categories: form-based languages, block-based languages, diagram-based languages, and icon-based languages. All categories (or subcategories), based on visual program representation, presented in both [1] and [2] are assigned to a category under our new categorization. For instance, block-based languages represent jigsaw puzzle pieces in [1], while diagram-based languages cover diagrammatic languages in [2], data flow graphs and directed graphs in [1]. Further, we filtered out the subcategories in [1] which apply to textual programming languages.

We define the VPL categories as follows:

Block-based languages allow users to drag and drop ''blocks'' (program elements) from a predefined list of commands into the development area. These blocks are pieced together to make a program. This paradigm prevents syntax errors, which reduces the mental load of end users allowing them to focus on concepts rather than implementation details. Many block-based languages such as Scratch [92] and App Inventor [93] have made application development accessible to numerous end users.

Icon-based or *Iconic languages* capitalize on the use of icons, graphical symbols representing objects or action [14]. Chang [15] explained that icons can be classified as complex and elementary icons. Elementary icons represent objects (e.g., file) or actions (e.g., delete, edit), whereas complex icons are composite object icons and visual sentences. Composite object icons are the outcome of assembling elementary object icons. Visual sentences are spatial arrangements of elementary icons. Recently iconic languages have been used to enable end users without programming experience to create applications based on triggers and actions. For instance, end users may specify an alert to be sent to them when the indoor temperature is less than 40 °F. More examples can be found in [16]–[18], and [94].

Form-based visual programming languages allow end-user developers to configure a form, in which triggers and actions are added by textual drop-down menus or visual drag-and-drop [19]. Some form-based approaches are mostly visual, whereas others use some textual specifications. The textual form-based paradigm utilizes a declarative approach to programming, that is based on a dependency-driven, direct-manipulation model [20]. Users of form-based languages create or configure cells, and define formulas for those cells [21]. These formulas reference values contained in other cells and use them in calculations. Whenever a cell's formula is defined, the underlying evaluation engine calculates the cell's value, recalculates the values of cells that reference recalculated cells, and displays new results on the screen.

Diagrams have been utilized as communication and thinking tools across many domains [22]–[24]. *Diagram-based visual programming languages*, also known as diagrammatic or data flow languages, are characterized by connecting graphical objects (e.g., boxes) by arrows, lines, or arcs that represent relations. To understand a diagram-based program, users traverse the diagram. Such a diagram uses different means of perceptual coding to represent the flow of the program. For example, flowcharts use connectedness and directionality to represent how a piece of information is related to one another, and how it flows from one to the other [25].

This study will classify several visual programming tools according to the aforementioned VPL categories.

III. RELATED WORK

We divide the related work into two categories: end-user development review studies, and visual programming review studies.

A. END-USER DEVELOPMENT REVIEW STUDIES

There have been a few EUD review studies in the last decade. Ko *et al.* [6] conducted a survey that focused on EUSE methods. The surveyed articles are relatively old (prior to 2010). The study emphasized the need for software engineering practices for programs created by end-user developers as they

often contain errors that they are not aware of or unable to track. The study highlighted several practices that help end users to detect errors with testing, check for consistency, and check the program against specifications. The study briefly mentioned some visual programming tools applications, but there was no systematic analysis of visual programming approaches in the study. Moreover, the review presented in the study has been performed and analyzed in the research domain of their authors [12].

Paternò [26] conducted a survey of EUD that introduced the motivations behind end-user development, discussed its roots, and presented the state of art as of 2013. The author discussed various approaches and classified them in terms of their main features and the technologies and platforms for which they have been developed. Further, the study provided an indication of direction for further research. For instance, a possible future research could be investigating how end users could customize software compositions according to the context of use. Despite mentioning visual programming as an EUD field, Paternò did not provide a systematic analysis of visual programming approaches. Further, the article cited less than ten articles discussing novel visual programming tools, eight of them were published prior to 2010.

Burnett and Scaffidi [12] discussed possible research areas for EUD such as the need to help end-user developers produce software with stronger guarantees of security and privacy, and without interfering with the lightweight, iterative nature of the EUD life cycle. Burnett and Scaffidi briefly discussed only three examples representing three approaches of visual programming languages (block-based, diagram-based, and form-based). However, the examples are from industrial tools such as LabView [100] and Microsoft Word [101] as opposed to research articles in the literature.

Another review study in [27] presented a literature review study of EUD, covering the years between 2004 and 2013. The study pointed to an increase interest in EUD research. Further, the field is strongly dominated by the engineering of systems and the evaluation of these systems in a lab setting as opposed to a natural setting. Like the previous studies, this study does not analyze visual programming research in detail.

Hang and Zhao conducted a systematic review study to examine activities and tools that support service composition by end-user developers [28]. The study identified service composition as a challenging task to end-users, requiring further investigation to identify the impediments encountered by end-users when composing services. Only a handful of articles on visual programming were reviewed in the study and they all were published prior to 2010.

A recent study [29] gave an overview of recent approaches and techniques that support end-users in the three related research areas discussed in the introduction, EUP, EUD, and EUSE. The study investigated 165 articles between 2000 and 2017 to cover recent trends in EUP, EUD, and EUSE. This research included proposed frameworks and approaches in various domains and recognized research focus towards supporting domain experts in business data management, web application, mashups, and smart technology environments. The study revealed that most EUD tools are general purpose. Further, the study identified component-based programming as the most used interaction style among 14 classifications proposed by the authors. Further, the authors suggested to conduct comparative studies of EUD approaches and emphasized the need for further investigation of EUD activities to guide the design of future tools. Despite the useful insights, the study classified the articles based on broad EUD, EUP, and EUSE techniques that do not cover all VPL classifications discussed earlier in the introduction. As such, the findings are not specific to visual programming approaches.

Another recent study [7] reviewed the most recent trends and classification of 21 end-user development tools developed between 2007 and 2017. According to the review, the main direction in EUD is towards generalized tools facilitated on the web to support multi-platform accessibility. The study outlined major limitations in the tools, for example end-users are limited to customizing existing applications as opposed to creating them from the ground up. Further, the authors uncovered insufficient use of usability principles in developing EUD tools. Based on the four interaction styles attributed by Burnett and Scaffidi [12], visual programming was identified as the most used interaction style to support EUD. Nevertheless, the study did not cover all VPL classifications discussed in the introduction. Further, the majority of the articles surveyed in the study were conference articles, and only a few journal articles were from journals relevant to visual programming such as the Journal of Visual Languages and Computing.

To sum up, despite the variety of review studies that targeted EUD in general, none of these studies focused specifically on visual programming with all its classifications (e.g. diagrammatic, iconic, block-based). Instead, it was partially covered in some of the studies as a subset of EUD.

B. VISUAL PROGRAMMING REVIEW STUDIES

Only a handful of studies reviewed visual programming related research. The studies reviewed how visual programming tools were used in specific domains such as IoT, robotics, and education.

As an example, the authors in [8] surveyed 16 visual programming environments allowing end users to create applications involving robots with social capabilities. Only one article out of the 16 articles was published in a journal. The study shows that recent tools are adopting component-based software engineering approaches, but the tools need to be evaluated with real end users as opposed to university students, and should be validated in a real setting as opposed to a laboratory. The study mainly focused on the analysis and challenges specific to robotics-related approaches such as scripting-based, rule-based systems, state-based and behavior-based systems.

Another example of a domain-specific visual programming review study is the study in [10]. The author surveyed

13 articles that use visual programming to support the creation of IoT-based applications. The vast majority of the articles were conference articles. The study reported that VPLs make it easy for end users to visualize the programming logic and eliminate the burden of handling syntactical errors. However, significant time is spent on creating small-scale IoT applications with visual programming. The study focused on IoT-specific attributes such as the hardware platform (e.g. Raspberry Pi, Arduino) and overlooked the broad VPL classifications (e.g. diagrammatic, iconic, block-based).

A recent study conducted a systematic literature review to examine the role of visual and textual programming languages in helping students learn how to program [30]. The study reported that the choice of textual programming language is not a crucial one as languages are alike. However, the use of visual programming to introduce students to programming concepts is of utility as long as it is within a short time frame.

To sum up, all the aforementioned review studies are domain specific, and as such do not provide a generic view of VPL approaches applied in multiple domains beyond the domains of education, IoT and robotics.

Our study differs from the aforementioned reviews by focusing on the articles related to visual programming tools that represent several domains, closely examining the visual programming approaches with all its classifications, and identifying how such approaches are used to solve particular problems in several domains. Moreover, the study sheds light on commonalities and differences between tools of the same or different approaches or domains. Finally, the study discusses only the approaches that are backed up by empirical evidence. The details of the empirical evidence are thoroughly discussed in the study.

IV. METHODOLOGY

This systematic literature review analyzes the literature related to visual programming approaches, providing a background for new tools and methods, and identifying direction for further investigation. This review follows the guidelines described by Kitchenham and Charter [5]. The process consists of these main phases: (1) defining the review protocol including defining the research questions, methods to answer them, search strategy, and inclusion and exclusion criteria. (2) conducting the study by selecting the articles, evaluating their quality, and analyzing the results. (3) reporting the results.

A. RESEARCH QUESTIONS

We formulated two main research questions:

- **RQ1:** *What visual programming tools have been proposed in the literature to support end-user developers?*
- **RQ2:** *What evaluation methods have been used by the authors of these tools?*

We used several dimensions to answer the two research questions. Tables 2 and 3 show the list of dimensions used to answer the questions. The formulation of dimensions is inspired by those proposed in [29] to obtain general information of EUD, EUP, and EUSE tools.

TABLE 2. Research question 1 (RQ1) dimensions.

Label	Dimension	Description
RQ1-D1	VPL Classification	What VPL classification do the proposed tools fall under? Examples: Block-based, form-based, diagram-based, icon-based
RQ1-D2	Interaction Style	What interaction techniques and styles do the proposed tools utilize? Examples: Direct manipulation, menu selection, form filling.
RQ1-D3	Target Users	What types of users do the proposed tools target? Examples: generic, students, domain experts.
RQ1-D4	Domain	In what field or realm will the proposed tools be used? Examples: education, IoT, robotics.
RQ1-D5	Platform	In which hardware environment are the proposed tools executed? Examples: web, mobile, desktop.

TABLE 3. Research question 2 (RQ2) dimensions.

Label	Dimension	Description
RQ2-D1	Type of Empirical Evaluation	What type of empirical evaluation was used to to back the validity of the proposed tools? The evidence can be of varying strength. Examples: Formal experiment, evaluation study, informal evaluation.
RQ2-D2	Type of Test Participants	What type of users participated in the evaluation of the proposed tools? Examples: Students, domain experts.
RQ2-D3	Number of Test Participants	What is the number of users who participated in the evaluation of the proposed tools?
RQ2-D4	Programming Skills of Test Participants	What are the coding skills of the users who participated in the evaluation of the proposed tools? Examples: Novice, beginner, intermediate, advanced
RQ2-D5	Evaluation Methods	What evaluation methods were used to evaluate the proposed tools? Examples: Survey-based, task-based
RQ2-D6	Evaluation Measures	What evaluation measures were used to evaluate the proposed tools? Examples: Number of errors, completion time
RQ2-D7	Accessibility	Is the tool accessible online?

We used five dimensions to answer the first research question (RQ1). RQ1-D1 documents the VPL classification of the tools. RQ1-D2 describes the specific interaction styles the tools use. We used the interaction styles described in [81]. RQ1-D3 documents the type of users whom the tools are targeting. RQ1-D4 describes the fields where the tools are used. RQ1-D5 identifies the hardware environment in which the tools live.

We used seven dimensions to answer the second research question (RQ2). RQ2-D1 reports the existing empirical evidence for the efficacy of the proposed tools. RQ2-D2 identifies the types of users who participated in the evaluation.

RQ2-D3 reports the number of test participants, while RQ2-D4 describes the level of the programming skills that test participants have. RQ2-D5 states the evaluation methods used to evaluate the proposed tools, while RQ2-D6 reports the evaluation measures used to evaluate the tools. Finally, RQ2-D7 reports whether the tools are accessible online. Tool accessibility allows authors to receive feedback from a larger community.

B. SEARCH PROCESS

We conducted our search in popular databases in the field of end-user development, namely Scopus, IEEE Xplore, ACM Digital Library, Springer Link, and Eric. The publications covered are published between 2010 and 2020.

To identify keywords for the search string, we analyzed our research questions, goals, and previous related literature review studies. Thereafter, we conducted a pilot study in which we executed and refined the keywords and the search string iteratively. The process was inspired by [31]. The keywords we used for the search were "End-User Development" and "Visual Programming". A correlated keyword for "End-User Development" is "End-User Programming", and one for "Visual Programming" is "Visual Language". Further, we included keywords associated with the VPL classification such as "block", "diagram", "dataflow", etc. The search string was defined using the Boolean operators as follows:

('End-User Development' OR 'End-User Programming') AND ('Visual Programming' OR 'Visual Language') AND ('block' OR 'diagram' OR 'diagrammatic' OR 'dataflow' OR 'data-flow' OR 'icon' OR 'iconic')

In order to evaluate the articles according to their relevance to our research questions, we defined the inclusion and exclusion criteria shown in Tables 4 and 5. The inclusion and exclusion criteria helped us to reduce the number of studies that are not relevant to our research question. We excluded articles such as tutorials, posters, technical reports, and PhD thesis reports as they are not peer reviewed. Therefore, they may be less rigorous and mature than those that are peer reviewed.

TABLE 4. Inclusion criteria.

Criterion	Description
IC-1	The article describes a tool that uses a visual programming language.
IC-2	The article primarily targets novice programmers or end-user developers who do not have professional IT training.

TABLE 5. Exclusion criteria.

Criterion	Description
EC-1	The article is published in a language other than English.
EC-2	The article is a short paper, book, tutorial, editorial, abstract, poster, panel, lecture, round table, workshop, demonstration, workshop, PhD thesis, or a review paper.
EC-3	Duplicate article that discusses a tool or approach reported in another article (Only the most recent article is included in this case)
EC-4	The article does not meet the quality criteria defined in Table 6

TABLE 6. Quality criteria.

Criterion	Description
QC-1	The article presents the problem statement unambiguously.
QC-2	The article discusses the technique, platform, and target users of the tool.
QC-3	The article presents empirical evidence that supports the validity of the study. The evidence is in the form of formal experiment, evaluation study, or informal feedback.

We executed our search after defining the inclusion and exclusion criteria. We conducted our search in the selected search databases (Scopus, IEEE Xplore, ACM Digital Library, Springer Link, and Eric) using our search query. We restricted our search to article titles, abstract, and keywords. We executed the same search string in the search databases. However, some customization was needed. For instance, in IEEE Xplore and ACM Digital Library, rather than using the full textual search string, we broke the string into three parts: ('End-User Development' OR 'End-User Programming'), ('Visual Programming' OR 'Visual Language'), and ('block' OR 'diagram' OR 'diagrammatic' OR 'dataflow' OR 'data-flow' OR 'icon' OR 'iconic'). Thereafter, we combined them visually using an "AND" operator.

Our search process consists of the following steps:

1) Reading the article title, abstract and keywords, and applying inclusion criteria IC-1 and IC-2.
2) Reading the publication information e.g. language, publication type (e.g. journal, conference), publisher and length (to determine if it's full or short), titles, and abstracts, and applying exclusion criteria EC-1, EC-2 and EC-3.
3) Reading the introduction, evaluation and conclusion, and skimming through the other content of articles included in step 2 to eliminate irrelevant articles which do not meet the quality criteria QC-1, QC-2 and QC-3.
4) Reading all the content of the selected articles in step 3 and collecting data.

Figure 1 shows the flowchart of the article selection process. 550 articles were returned by an automatic search in the selected databases. After applying the inclusion criteria, only 223 articles were included out of 550. After applying the exclusion criteria, 91 articles were removed. Examining the articles against the quality criteria listed in Table 6 caused the removal of 102 articles. Finally, a total of 30 article were selected for this article.

V. RESULTS

Table 7 shows an overview of the 30 articles selected in this study. The articles were classified according to the five dimensions we used to answer the first research question (RQ1): VPL classification (RQ1-D1), interaction style (RQ1-D2), target users (RQ1-D3), domain (RQ1-D4), and

TABLE 7. The selected articles evaluated against first research question (RQ1) dimensions.

No.	Article	RQ1-D1	RQ1-D2	RQ1-D3	RQ1-D4	RQ1-D5
A-1	Johnsson & Magnusson (2020) [32]	Form-based	Direct Manipulation	Generic	IoT	Mobile
A-2	Bak et al. (2020) [46]	Block-based	Direct Manipulation, Form Filling, Menu Selection	Generic	IoT	Web
A-3	Cannavò et al. (2020) [42]	Diagram-Based	Direct Manipulation, Form Filling, Menu Selection	Generic	Interactive Displays	Desktop
A-4	Huang et al. (2020) [33]	Diagram-based	Direct Manipulation	Generic	IoT, Robotics	Web
A-5	Sorce et al. (2019) [34]	Form-based	Direct Manipulation, Form Filling	Engineers	Engineering	Web
A-6	Kunimune et al. (2019) [38]	Block-based	Direct Manipulation, Form Filling	Students	Education	Web
A-7	Abe et al. (2019) [39]	Block-based	Direct Manipulation, Form Filling	Students	Education	Web
A-8	Tamilselvam at al. (2019) [41]	Diagram-based	Direct Manipulation, Menu Selection	Novice Engineers	AI/Software	Web
A-9	Valtolina et al. (2019) [43]	Diagram-based	Direct Manipulation, Form Filling, Menu Selection	City operators	IoT (Smart City)	Web
A-10	Rao et al. (2018) [44]	Block-based	Direct Manipulation, Form Filling, Menu Selection	Students	Education (Data Science)	Web
A-11	De Luca et al. (2018) [45]	Diagram-based	Direct Manipulation, Form Filling	Students	Education (Robotics)	Web
A-12	Broll et al. (2018) [49]	Block-based	Direct Manipulation, Form Filling, Menu Selection	Students	Education (Distributed Computing)	Web
A-13	Mei et al. (2018) [47]	Digram-based	Direct Manipulation, Form Filling, Menu Selection	Generic	Information Visualization	Web
A-14	Stratton et al. (2017) [50]	Block-based	Direct Manipulation, Form Filling, Menu Selection	Museum Employees	Interactive Displays	Web
A-15	Francese et al. (2017) [51]	Icon-based	Direct Manipulation, Menu Selection	Generic	IoT	Mobile
A-16	Merkouris et al. (2017) [36]	Block-based	Direct Manipulation, Form Filling, Menu Selection	Generic	Robotics	Desktop
A-17	Feng et al. (2017) [37]	Block-based	Direct Manipulation, Form Filling, Menu Selection	Students	Education (Distributed Computing)	Web
A-18	Turchi et al. (2017) [59]	Block-based	Direct Manipulation, Menu Selection	Generic	Interactive Displays	Mobile
A-19	Valderas et al. (2016) [48]	Form-based	Direct Manipulation, Menu Selection, Form Filling	Generic	Service composition	Mobile
A-20	Thamsen et al. (2016) [52]	Diagram-based	Direct Manipulation, Menu Selection	Programmers	Data Science	Web
A-21	Alexandrova et al. (2015) [53]	Diagram-based	Direct Manipulation, Menu Selection	Generic	Robotics	Mobile
A-22	Rough & Quigley (2015) [54]	Block-based	Direct Manipulation, Form Filling, Menu Selection	Researchers	Research, Psychiatry	Mobile
A-23	D. Lizcanoa et al., (2014) [35]	Diagram-based	Direct Manipulation, Form Filling, Menu Selection	Generic	Service Composition	Web
A-24	Danado & Paternò (2014) [60]	Block-based, Icon-based	Direct Manipulation	Generic	IoT	Mobile
A-25	Booth & Stumpf (2013) [56]	Block-based	Direct Manipulation, Menu Selection	Generic	IoT	Desktop
A-26	Weber et al. (2013) [58]	Form-based	Form Filling	Generic	Business	Web
A-27	Chen & Tu (2013) [61]	Diagram-based	Direct Manipulation, Menu Selection	Generic	Reporting	Desktop
A-28	Luong et al. (2012) [57]	Diagram-based	Direct Manipulation, Menu Selection	Generic	GIS	Web
A-29	Cabitza & Gesso (2012) [55]	Block-based	Direct Manipulation, Menu Selection	Generic	Document Management	Desktop
A-30	Drey & Consel (2012) [62]	Diagram-based	Direct Manipulation, Menu Selection	Generic	IoT	Desktop

platform (RQ1-D5). Table 8 shows the same articles examined against the seven dimensions we used to answer the second research question (RQ2): Empirical evidence (RQ2-D1), test participants (RQ2-D2), number of test participants (RQ2-D3), programming skills of test participants (RQ2-D4), evaluation methods (RQ2-D5), evaluation measures (RQ2-D6), and accessibility (RQ2-D7).

Figure 2 shows a timeline of the selected articles. 50% (15) of the selected articles were published in journals and the other 50% (15) of the articles are conference papers. 66.67% (20) of the articles were published after 2015. Noticeably five journal articles were published in the Journal of Visual Languages and Computing, a highly relevant journal to visual programming articles. This journal has recently

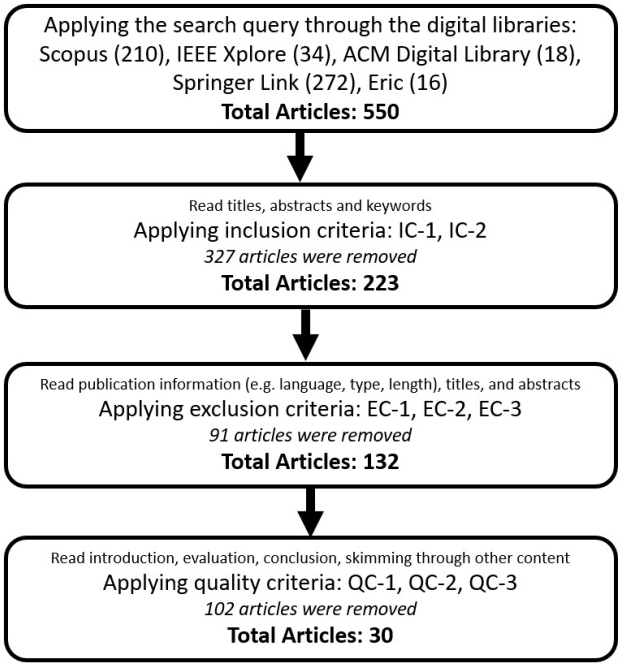

FIGURE 1. Flowchart of article selection process.

merged with the Journal of Computing Languages (COLA). Interestingly, one of the selected article was published in COLA. Two journal articles were published in the Journal of Parallel and Distributed Computing. The remaining journal articles were published in several venues such as ACM Transactions on Computing Education, ACM Transactions on the Web, International Journal of Human-Computer Studies, The Journal of Systems and Software, CAAI Transactions on Intelligence Technology, Future Generation Computer Systems and Multimedia Tools and Applications. The majority of these journals are ranked Q1 or Q2 according to Scimago Journal and Country Rank [119].

Four conference articles were published in the International Symposium on End-User Development (IS-EUD). Two conference articles were published in the IEEE Symposium on Visual Languages and Human-Centric (VL/HCC). The remaining conference articles were published in mostly ACM or IEEE conferences.

Figure 3 shows a geographical mapping of the selected articles. By far the majority of the selected articles were written or co-written by researchers from European universities predominantly British and Italian universities. The remaining articles were written by researches from Asian and North American universities.

VI. FIRST RESEARCH QUESTION DIMENSIONS

To answer the first research question, we analyzed the selected articles against the five research question dimensions discussed in the subsequent subsections. Furthermore, we used examples from some of the selected articles to shed light on details of interest.

A. RQ1-D1: VPL CLASSIFICATION

43.3% (13) of the articles used a block-based approach, while 40% (12) of the selected articles used a diagram-based approach. 13.3% (4) of the selected articles used a form-based approach, whereas only 6.6% (2) articles used an icon-based approach (Figure 4). Despite the differences of the approaches, all the tools use high-level abstractions to simplify the process of creating a program and hide implementation details. The abstractions are often represented as visual components that end users can drag and drop. The abstractions allow end users to incorporate into their applications: functions [32], [51], web services [43] or software modules [41]. Abstractions are used to increase tool accessibility though this may be at the cost of expressiveness as the abstractions are often designed to operate in a restricted way. Another common theme is the use of color to represent data flowing between components [51], [60].

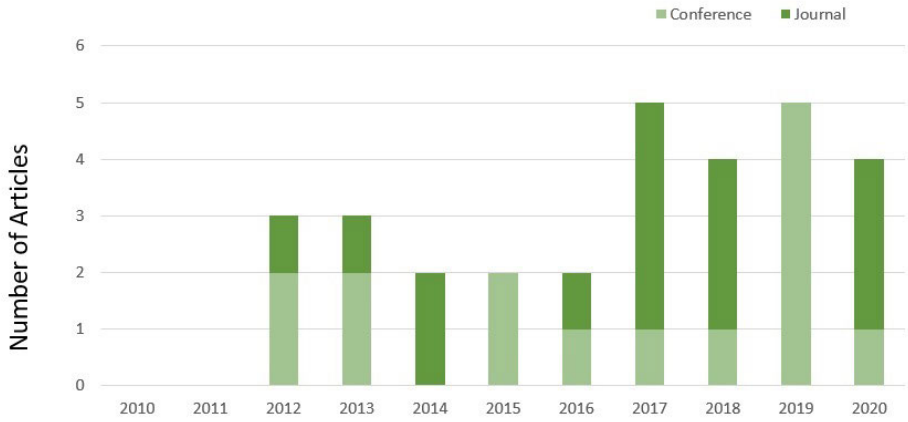

FIGURE 2. A timeline of the selected articles.

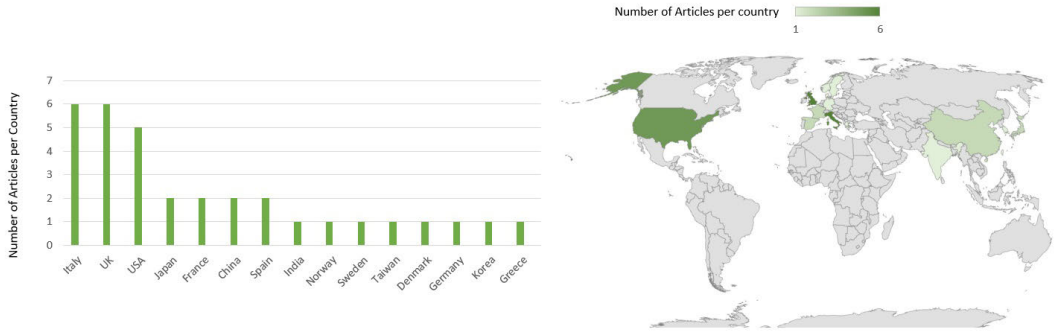

FIGURE 3. Geographical mapping of the selected articles.

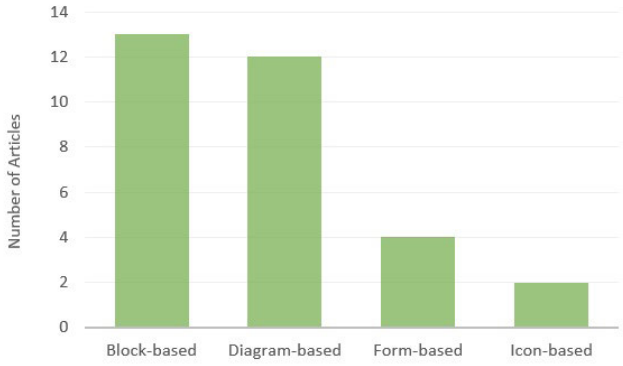

FIGURE 4. VPL Classification of the selected articles.

1) FORM-BASED TOOLS

Tools using the form-based approach empower users to construct a functional user interface by dragging and dropping visual components to a form. Often these visual components represent services such as ''weather condition'' [48] and ''phone camera'' [32]. Thereafter, the user can configure these services visually to construct a program.

The tool, named as ''GPE'' [32], is intended for IoT applications. End users use a visual builder to build a user interface. However, instead of dragging and dropping graphical components as in conventional visual builders, users drag and drop functionalities such as ''Take Photo'', ''Starting a motor'', etc. Thereafter, the system suggests graphical components (e.g. button, panel) to match the functionality.

Once an end user has selected the visual representation, he/she may customize the link that is created between the functionality component and the graphical component, thus specifying the action that will be performed for the selected graphical components. The authors call this approach the ''inverted approach'', which focuses on visually presenting functionality. This approach allows for a strict separation of concerns between GUI and business logic.

Another example of the form-based approach is explained in [48]. End users compose services by simply graphically adding and configuring elements to a form in a step-by-step approach. For instance, end users may add a service that allows users to book a seat in a library. Thereafter, end users may add some logic for custom behavior. As an example, end users may add a conditional that provides bike parking if the weather condition is sunny.

Another tool that capitalizes on the form-based approach is described in [34]. The tool helps end users to detect possible patent infringement in the field of mechanical engineering. The tool presents end users with visual blocks which allow them to find various types of products. Once end users have configured their search criteria, search results show up, and a red color code is used in search results to highlight possible infringement.

2) DIAGRAM-BASED TOOLS

Tools using the diagram-based approach empower users to construct a program by connecting together visual components where the output of a component serves as a data input to another component. In some tools such as [41], the visual components may represent data sets and algorithms, whereas in other tools such as [42], the visual components simply represent graphical components. Further, the user can configure these services by setting its properties.

As an example, the tool in [41] uses a diagram-based approach to building deep learning models. End users drag and drop layers (available on the left menu), configure their properties, and connect them to build a deep neural network without writing code. The layers include data sets and activation functions. The tool provides initial default configuration of the parameters of each layer which the end user can change. Further, the tool displays error messages and suggestions on the right side.

Another example of a diagram-based approach is described in [42]. End users design 3D interactive graphics by dragging and dropping blocks to the tool environment. Each block represents a 3D object. The end-user can connect the objects together using lines. Further, the end user can specify interactivity by graphically configuring an object to do some logic (e.g. when the user taps an object, another object can show up or a sound segment is played).

The article in [45] uses a diagram-based approach to allow the development of IoT and robotics-based applications. The

tool uses Microsoft Visual Programming Language (MVPL) which provides a graphical dataflow-based programming model [95]. The program allows concurrent development as it uses the metaphor of multiple workers on an assembly line, who perform tasks as jobs arrive. The article presents an example of building an application of a robot navigating through a maze autonomously. The application is entirely built with blocks such as variables, data items, and conditionals that are connected visually to show the dataflow.

3) BLOCK-BASED TOOLS

Tools using the block-based approach allow users to construct a program by combining together visual blocks that fit together like a jigsaw puzzle. In some tools such as [49] and [54], the blocks represent programming constructs such as variable setting or loops, whereas in other tools such as [60], the visual blocks represent interactive components such as a map.

For example, the authors of [60] designed a tool, ''Puzzle'', that allows end users to develop mobile IoT applications. Puzzle environment has three components: start, development, and execution of an application. The start component allows users to create or access a previously created application. The development component allows end users to drag and drop jigsaw pieces to the canvas and combine them together to obtain the desired functionality. Depending on the data, each jigsaw piece has input and output that are color coded so that end users combine with the pieces with matching color codes. The execution component allows end users to run the application.

As another example, the authors of [49] used a block-based approach to teaching distributed computing topics such as peer-to-peer communication. For instance, the tool has a mechanism for communication with messages. End users can specify one or more variables on the ''send msg'' block. On the receiver side, these data items will appear in the ''when I receive'' block header as variables with the appropriate names.

The authors of [54] designed a block-based tool that allows researchers to build Experience Sampling Method (ESM) applications which record thoughts and feelings of participants. Researchers drag and drop program components (blocks). Blocks allow researchers to define variables, expressions, actions, conditions, and triggers.

Another example of a block-based approach to application is the tool explained in [50]. The tool allows museum employees to build applications to engage visitors with museum exhibits. The tool builds on the block-based diagram of Scratch [92], but is event driven. For instance, end users can decide images to display for certain exhibits, as well as play an audio, and configure the styling of the exhibit.

4) ICON-BASED TOOLS

Tools using the icon-based approach allow users to construct a program by connecting icons together to represent data flow. Icons represent services such as determining user location or saving a file.

For example, MicroApp is an icon-based tool, which allows end users to create mobile applications [51]. The applications use services that are represented by icons that are connected to show data flow. The connectors use different color codes. Colors represent the type of data that is transmitted between services. For instance, pink corresponds to images whereas yellow represents email objects. For instance, the end user would like to send pictures to their contacts in conjunction with the address of the picture locations. To that effect, the end user places the icon ''Take'', connects it with ''Save'' so that the picture is saved on her device. Thereafter, the end user chooses the contact (Static icon). The Location service gets the geographical coordinates and outputs them to the Address GeoLocation. Finally, the end user selects the StaticMail and joins it to Save, Static, and Address GeoLocation. This results in an email being sent to the selected contact containing the saved picture and location.

B. RQ1-D2: INTERACTION STYLE

Figure 5 shows the interaction styles of the selected articles. Since the articles present VPL-based tools, they all use direct manipulation. For instance, all the presented tools use a form of drag and drop of visual objects such as blocks and GUI components. 73.3% (22) of the articles use menu selection. Some of the menus are textual whereas others are visual. Often the menus help end users find the correct visual objects needed to build an application. For instance, the tool

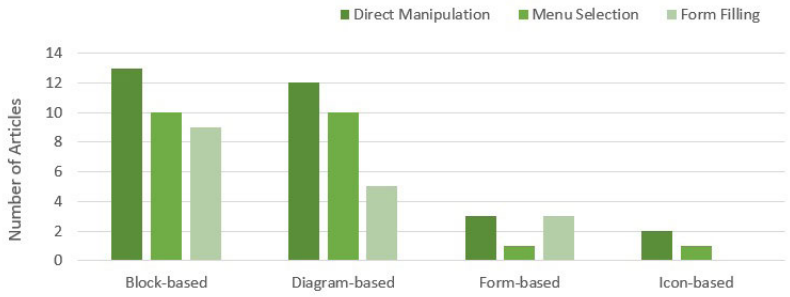

FIGURE 5. Interaction styles of the selected articles.

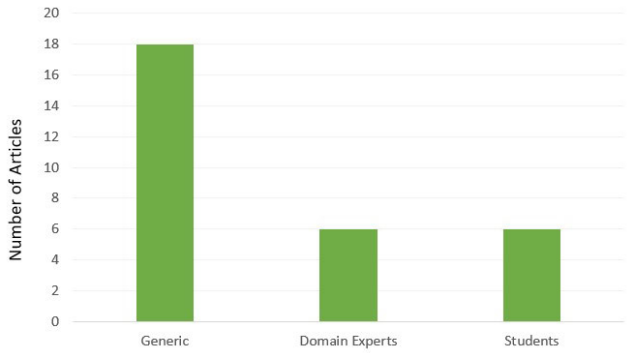

FIGURE 6. Target users of the selected articles.

discussed in [43] uses a menu of components organized in two categories: input and output [43]. End users drag and drop the components to build smart city applications, for instance, monitoring city temperature as well as water consumption. As an another example, the tool explained in [62] uses a menu to help end users find visual objects representing components in a smart home such as sensors and actuators.

56.6% (17) of the selected articles used form filling as an interaction technique. Form filling can be used in a variety of ways. It can be used to allow users to change application settings, appearance, or behavior. For instance, the tool discussed in [58] allows non-programmer domain experts to model and deploy business processes by editing form-based web services. Specifically, the tool allows the user to import news data via a news web service. The form fields are highlighted with different color codes. For instance, the blue color refers to unmapped fields whereas other colors represent data mappings and static assignments. As another example, the tool developed by Lizcano *et al.* [35] uses a form filling to allow users to edit the properties of visual components representing services.

C. RQ1-D3: TARGET USERS

Figure 6 shows the target users of the selected articles. 60% (18) of the tools presented in the selected articles do not target a specific class of end users. These tools are intended to be used by average end users who have basic IT skills, but no formal training in any specific domain.

20% (6) of the tools are intended to be used in class for students to teach them basic programming concepts such as variables, loops, etc. Only one of these educational tools uses a diagrammatic approach [45]. The remaining tools are block based.

20% (6) of the tools are intended to be used by domain experts. Two of the tools discussed in the articles are designed to be used by engineers. One of the tools, discussed earlier, helps engineers to detect possible patent infringement in the field of mechanical engineering [34]. The other tool [41] is designed for novice software engineers to help them design deep learning models with a diagram-based approach. The remaining tools target different classes of users. For instance, the tool in [54] is intended to empower researchers to build Experience Sampling Method (ESM) applications. The tool discussed in [52] allows programmers familiar with data science skills to develop distributed data analytical systems using a diagram-based approach. This approach facilitates the accessibility and efficiency of developing such applications. Aimed at museum employees, the tool presented in [50] uses a block-based tool to allow the development of interactive exhibits to engaged museum visitors. The tool discussed in [43] allows city operators to build smart-city applications use a diagram-based approach.

D. RQ1-D4: DOMAIN

Figure 7 shows the target domain of the selected articles. All the surveyed tools are domain specific, and not meant to be used for multiple purposes. 23.3% (7) of the tools presented in the selected articles fit in the category of IoT. Some of these tools allow end users to utilize the resources of their personal phones such as [32] and [51]. Others enable the usage of sensors deployed in cities such as [43] or sensors used in variety of contexts such as smart homes or available for medical staff to monitor patients' health [60].

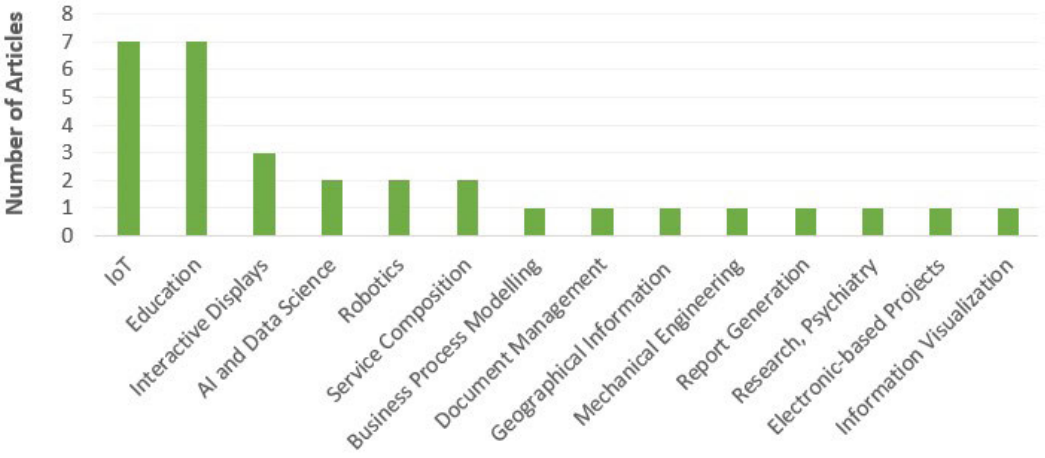

FIGURE 7. Domains of the selected articles.

While tools built for IoT applications use different VPL approaches (form-based, diagram-based, and block-based), a common theme among the tools is allowing end users to access services such as sensors and phone resources by means of visual abstractions. For instance, the tool presented in [46] shows various types of sensors (such as motion and presence sensors) as visual blocks. End users can define logic that is related to the the states of such sensors. As a simple example, end users may specify a welcoming message upon switching on a certain light.

23.3% (7) of the tools are intended to be used in an educational setting. Some of these tools such as [38] and [39] are utilized for teaching basic programming principles such as loops and conditionals, whereas the tool in [44] aims at teaching data science concepts to non-programmers, and the tool in [45] is intended for teaching computing and engineering concepts and for programming robots.

Apart from one tool [45], all the tools built for educational applications used a block-based approach. The block-based educational tools use similar visual blocks to represent programming constructs such as variable setting, control flow, and conditionals. The tools aimed at teaching a specific topic in computer science such as parallel programming represent related concepts (e.g. parallel blocks, message passing, events) visually. For instance, the concept of events is represented by a ''When'' visual block in [37] and [49].

The tool explained in [45] uses a diagram-based approach to programming education. For instance, a while loop is represented by a sequence of connected boxes. Each box represents a command, e.g. setting a variable. The first box starts the loop, and the last one ends it.

Three of the surveyed articles present tools that specialized in interactive displays. As discussed earlier, the tool in [50] uses a block-based approach to help end users build interactive museum exhibitions, while the tool presented in [42] uses a diagram-based approach where end users connect objects representing 3D models, configure the objects, and add some logic to build 3D interactive exhibitions.

Two of the surveyed articles present tools that specialized in artificial intelligence and data science. Both of the tools use a diagram-based approach. For instance, the tool in [41] allows end users to build a deep neural network. As discussed earlier, the tool represents layers with visual components that can be configured and connected to build a deep neural network. The layers include data sets and activation functions. The tool in [52] enables the construction of distributed data analytical systems. Like [41], end users connect components to build big data applications. The components can be data sources and data sinks as well as Map, Filter, Reduce, Join, and GroupBy operators.

The remaining articles target a variety of domains such as robotics [53], research and psychiatry [54], museum exhibits [50], report generation [61], mechanical engineering [34], geographical information [57], document management [55], information visualization [47], and business process modelling [58].

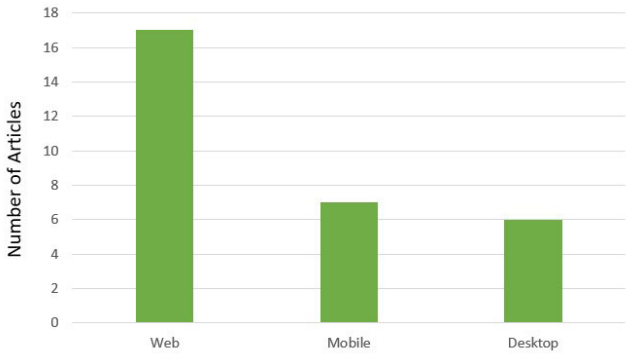

FIGURE 8. Platforms of the selected articles.

E. RQ1-D5: PLATFORM

Figure 8 shows the platforms for which the tools presented in the articles were developed. A high number (17) of the tools were developed for web platforms (56.6%), followed by 7 (23.3%) tools that were developed for mobile platforms, and 6 (20%) desktop-based tools.

Most of the tools that were developed for desktop platforms were built in or before 2013. This can be explained by the fact that desktop-based tools are cumbersome to contemporary users. They must be downloaded and installed, are operating-system dependent, and need frequent updates. Web platforms, on the other hand, offer various advantages as they are operating system independent, do not need to be updated, downloaded or installed. Tools developed for mobile platforms are on the rise. This can be explained by users increasingly preferring to use mobile applications. According to an App Annie report, users spent 120 billion dollars on application stores [78].

VII. SECOND RESEARCH QUESTION DIMENSIONS

To answer the second research question, we analyzed the selected articles against the seven research question dimensions discussed in the subsequent subsections. Furthermore, we used examples from some of the selected articles to shed light on details of interest.

A. RQ2-D1: TYPE OF EMPIRICAL EVALUATION

Figure 9 shows the different types of empirical evaluation performed to assess the learnability or efficiency of the tools discussed in the surveyed articles. We classified the types of empirical evaluation as follows: formal experiment, evaluation study, and informal evaluation. *A formal experiment* is a scientific test performed under controlled conditions, that is one factor is changed at a time, while all others are kept constant. The purpose of the test is to support, refute, or validate a hypothesis. Statistical analyses are carried out to accept or reject the hypothesis. *An evaluation study* is a test carried out to provide insights about certain parameters. There is typically no hypothesis to prove, and the results are often not statistically significant. *Informal evaluation* is a form of collecting data regarding certain aspects of a product. No test or hypotheses are established.

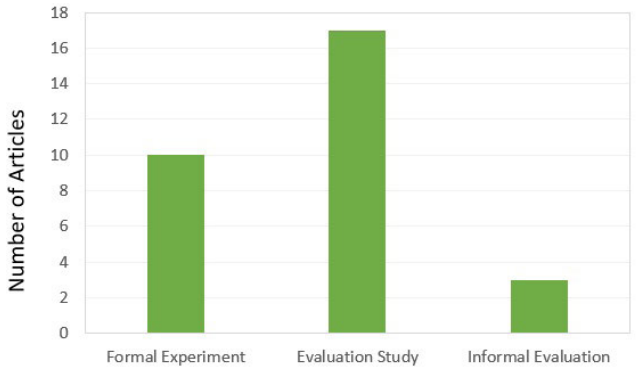

FIGURE 9. Types of empirical evaluation mentioned in selected articles.

33.3% (10) of the tools presented in the selected articles were evaluated by a formal experiment, while 56.6% (17) of the articles used an evaluation study to assess the presented tools. Additionally, only 10% (3) of the tools presented in the selected articles were evaluated by informal evaluation.

1) STATISTICAL SIGNIFICANCE

Only the tools evaluated with a formal experiment had statistically significant results. For instance, GPE [32], a form-based tool discussed earlier, was evaluated for task efficiency in the context of GUI development. The authors conducted the experiment with two groups. Participants used GPE in the treatment group, and Android Studio in the control group. The authors asked to participants to carry out three tasks. The tasks were designed to cover a wide range of GUI development ideas and components. The authors measured the quality of the solution as well as the completion time to obtain quantifiable data. The experiment concluded that GPE is more accessible to new users and more efficient to use than Android Studio. The authors used a t-test to determine statistical significance of the results. The results from the first two tasks were statistically significant (with p-values were 0.010, 0.009).

The tool discussed in [55] is a custom version of MIT Open Blocks [96]. The authors set a controlled experiment to test this hypothesis: the participants that received training and those who did not would take an equal time, on average, to perform each task. The participants were split into two groups: those who received a five-minute training session, and those who did not. The results did not point to a significant difference in performance, and therefore, the hypothesis could not be accepted.

The authors of the tool discussed in [33] conducted an experiment with 12 participants to evaluate a diagram-based tool that allows robotics applications with spatial visual programming (Vipo). To compare Vipo with a popular VPL tool (Blockly) [97], six participants used Vipo first and then Blockly, whereas the other six participants used Blockly first and then Vipo. One of the tasks given to the participants was a programmed workflow, and asked to answer questions such as identifying the optimal steps, conditionals, and number of machines. A paired t-test was performed, and the results suggest that participants spent less time on tasks with Vipo compared to Blockly (p-value was 0.009).

The tool, discussed in [61], VisualTP, is a tool that allows end users to generate reports using visual programming. To evaluate the tool, the authors conducted an experiment that compared VisualTPL with Microsoft Reporting Service. The results showed that end users have comparable performance with both tools when creating report layouts, but prefer VisualTPL for custom layouts. Their perference of VisualTPL had statistical significance (p-value < 0.05).

The authors of the tool discussed in [36] set up an experiment with 36 students to explore the benefits of learning to code for tangible computers, such as robots, in comparison to programming for a desktop computer. For this purpose, the authors used block-based visual programming environments, and measured engagement, attitudes, and learning performance. An ANOVA test was performed, and the study found out with statistical significance ($p = .001$) that students had higher engagement and learning performance with the visual programming tool geared for robotics as opposed to the desktop computer.

2) EVALUATION STUDIES

56.6% (17) of the articles used an evaluation study to assess the presented tools. For instance, the authors in [51] conducted a study to evaluate the perceived enjoyment and usefulness, behavioral control, intention to use, simplicity, and subjective satisfaction of the tool, MicroApp, the tool discussed earlier that allows end users to create mobile applications. 24 participants were recruited. The participants had varying programming skills (end users and programmers). The study involved a 20-minute training session, participants working on tasks related to the tool, followed by a questionnaire. The participants successfully completed the tasks at different times. Further, the participants appeared to have enjoyed the tool. Four end-user participants expressed the need of a technical person to use the system. The perceived usefulness was higher for end-user participants. Further, most participants found the system simple to use. They also expressed their intent to use the tool, and were satisfied with the tool.

The authors in [56] discussed the need to make programming Arduino-based single-board microcontrollers [98] accessible to end users via VPL. The authors presented a tool that takes a step in this direction (Modkit) using a block-based approach. To evaluate the tool, the authors conducted a think-aloud study with 11 participants using a textual programming editor as well as Modkit. None of the participants had prior experience with Arduino or Modkit, and they were not professional programmers. The authors first gave the participants a short tutorial on Arduino. Further, participants worked on two tasks, completed an interview, and filled out a background questionnaire. Only two participants out of eleven completed both tasks, and seven couldn't complete any task. Four participants who completed tasks were more successful using Modkit. The authors noticed that

TABLE 8. The selected articles evaluated against the second research question (RQ2) dimensions.

No.	Article	RQ2-D1	RQ2-D2	RQ2-D3	RQ2-D4	RQ2-D5	RQ2-D6	RQ2-D7
A-1	Johnsson & Magnusson (2020) [32]	Formal Experiment	Students	24	Beginner	Tasks	Time, Errors	Unavailable
A-2	Bak et al. (2020) [46]	Evaluation Study	Mostly students	33	Beginner, Novice	Tasks	Time, Errors	Online [103]
A-3	Cannavò et al. (2020) [42]	Formal Experiment	Students	14	Beginner	Tasks	Time, Errors	Unavailable
A-4	Huang et al. (2020) [33]	Formal Experiment	Students	12, 10	Mostly Novice	Tasks	Time, Errors	Unavailable
A-5	Sorce et al. (2019) [34]	Formal Experiment	Domain Experts	21	Unspecified	Tasks, Survey	Perc. usability, usefulness, workload	Unavailable
A-6	Kunimune et al. (2019) [38]	Evaluation Study	Students	37	Beginner	Survey	Perc. usefulness	Unavailable
A-7	Abe et al. (2019) [39]	Evaluation Study	Students	7	Novice	Tasks, Survey	Time, Perc. usability	Unavailable
A-8	Tamilselvam at al. (2019) [41]	Evaluation Study	Students	18	Intermediate	Tasks, Survey	Errors, Perc. usability	Online [104]
A-9	Valtolina et al. (2019) [43]	Evaluation Study	Students	18	Intermediate	Tasks, Survey	Time, Perc. usefulness & usability	Online [105]
A-10	Rao et al. (2018) [44]	Informal Evaluation	Students	20	Mostly Novice or Beginner	Survey	Perc. usefulness & usability	Online [106]
A-11	De Luca et al. (2018) [45]	Informal Evaluation	Students	Unspecified	Novice, Beginner	Survey	Perc. usefulness and usability	Online [118]
A-12	Broll et al. (2018) [49]	Evaluation Study	Students	24, 16	Beginner	Tasks, Survey	Perc. usefulness & usability	Online [111]
A-13	Mei et al. (2018) [47]	Evaluation Study	Students	15	Beginner, Intermediate	Tasks, Survey	Time, Perc. usability	Online [112]
A-14	Stratton et al. (2017) [50]	Informal Evaluation	Domain Experts	Unspecified	Novice	Tasks	Errors	Online [107]
A-15	Francese et al. (2017) [51]	Evaluation Study	Students, University Staff	24	Novice, Beginner, Intermediate	Tasks, Survey	Perc. usability & usefulness	Unavailable
A-16	Merkouris et al. (2017) [36]	Formal Experiment	Students	36	Novice	Survey	Perc. usefulness & Emotions	Online [113]
A-17	Feng et al. (2017) [37]	Evaluation Study	Students	96-100	Novice	Survey	Perc. usefulness	Online [114]
A-18	Turchi et al. (2017) [59]	Evaluation Study	Students, Domain Experts (Designers)	15, 3	Beginner, Novice	Tasks, Interviews	Perc. usability & usefulness	Online [115]
A-19	Valderas et al. (2016) [48]	Evaluation Study	Students	17	Novice, Advanced	Tasks, Survey	Perc. usability	Unavailable
A-20	Thamsen et al. (2016) [52]	Evaluation Study	Students	10	Intermediate, Advanced	Tasks, Survey	Perc. usability	Online [108]
A-21	Alexandrova et al. (2015) [53]	Evaluation Study	Unspecified	9	Expert, Advanced, Novice	Tasks	Time, Errors	Online [109]
A-22	Rough & Quigley (2015) [54]	Evaluation Study	Students	20	Novice, Beginner, Advanced	Tasks	Time, Errors	Online [116]
A-23	D. Lizcanoa et al., (2014) [35]	Formal Experiment	Students, Researchers, Employees	180	Mostly Novice	Tasks, Survey	Time, Perc. usability	Unavailable
A-24	Danado & Paternò (2014) [60]	Evaluation Study	Mostly University Staff	11	Novice	Tasks, Survey	Perc. usability	Unavailable
A-25	Booth & Stumpf (2013) [56]	Evaluation Study	Students	11	Beginner	Tasks, Survey, Interviews	Errors, Perc. workload	Online [110]
A-26	Weber et al. (2013) [58]	Evaluation Study	Domain Experts	18	Beginner, Good	Tasks, Survey	Time, Perc. usability	Online [117]
A-27	Chen & Tu (2013) [61]	Formal Experiment	Students	66	Beginner	Tasks, Survey	Time, Perc. usability	Unavailable
A-28	Luong et al. (2012) [57]	Evaluation Study	Students	32	Novice	Survey	Perc. usability	Unavailable
A-29	Cabitza & Gesso (2012) [55]	Formal Experiment	Students	32	Novice	Tasks	Time	Unavailable
A-30	Drey & Consel (2012) [62]	Evaluation Study	Students	12, 6	Novice	Survey	Perc. usability	Unavailable

participants were more successful in modification tasks as opposed to creation tasks. Further, participants rated their workload as higher in the textual environment, but they rated Modkit as more physically demanding. The authors noticed that participants carried out more clicks and mouse movement in Modkit. On the qualitative side, some participants found Modkit ''fun'', and ''easy to learn''. Other participants found Modkit confusing as it is like ''a puzzle on top of a puzzle''. In the post-session interviews, most participants mentioned that they were in favor of Modkit.

The authors of the tool described in [37] assessed their tool (Snap!) which uses block-based visual programming at the 18th Annual Women in Computing Day (WCD) in 2016. The authors taught basic parallel programming topics using the tool to middle school girls. They were divided into four groups of 24-25 students. Thereafter, the students filled out a survey as a form of assessment to the tool. As an example of an interesting insight, 86% of the students indicated that Snap! made their impression of computer science more favorable.

3) INFORMAL EVALUATION

10% (3) of the tools presented in the selected articles were evaluated by informal evaluation. For instance, the authors of the tool presented in [45] collected informal feedback by using the tool (VIPLE) in educational settings. For instance, using the tool in a robotics summer camp allowed the authors to focus on teaching computational thinking as opposed to spending considerable time on teaching the syntax of a textual language. Further, VIPLE was used in a course that introduces engineering to students. Feedback from students allowed the authors to improve the design of the tool. VIPLE was also used to teach parallel and distributed computing (PDC) topics to students in a course on advanced software. The authors of Quando, the tool presented in [50] presented a case study where they observed how end users use Quando to create museum exhibits with the tool. The authors observed that end users were able to add complexity to the application by copying existing logic and modifying it. The authors believe that copying may have happened because the tool had a limited set of blocks, and end users needed to find ways to overcome the limitations.

B. RQ2-D2: TYPE OF TEST PARTICIPANTS

Figure 10 shows an overview of the types of test subjects who participated in the evaluation of the tools. Overwhelmingly the authors evaluated their tools with students (25 articles out of 30), while only 4 articles evaluated the tools with domain experts. Some articles such as [46], [51], [59] and [35] evaluated their tools with different types of users.

It is natural for tools that are aimed at computer science education such as [38], [39], [44], [45] to be evaluated with students. However, for tools aimed at other purposes such as [32], [33], [41], [43], [51], [52], [57], [61], it would have be ideal to evaluate the usability and learnability of the tools with typical users or domain experts as this would generate more realistic feedback for the tool designers.

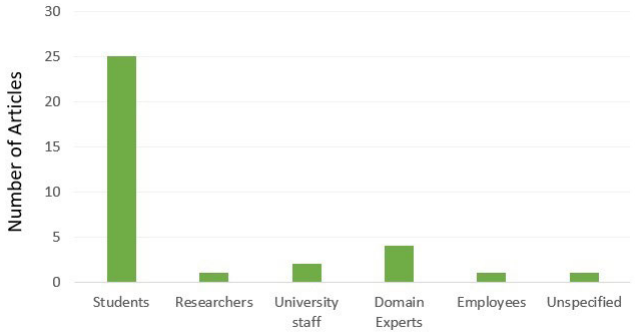

FIGURE 10. Types of users who participated in the evaluation of the tools discussed in the articles.

C. RQ2-D3: NUMBER OF TEST PARTICIPANTS

Figure 11 shows an overview of the number of test subjects who participated in the evaluation of the tools. The vast majority of the articles (27 out of 30) used 40 or less test subjects for evaluating the tools.

The article that used the highest number of test participants (180) for evaluation can be found in [35]. The authors set an experiment to evaluate a tool named ''FAST'' that helps end users compose services. The participant sample was diverse in terms of age, gender, educational attainment, and employment. In conducting the experiment, the researchers wanted to know if FAST indeed helps end users solve real-world problems, and whether it outperforms similar industrial tools. The results point to a positive outcome for FAST as significantly more end users managed to complete test tasks with FAST compared with the other tools. Further, participants rated FAST higher than the other tools in terms of usability and functionality.

Some authors such as [42] conducted formal experiments with a relatively low number of participants (14). Nevertheless, the results were statistically significant. Using several tasks, the authors compared the completion time of the visual tool they designed (Visual Scene Editor) against a similar tool (Leap Embedder) [40]. The results showed that participants were 51.58% faster on average with Visual Scene Editor, and participants encounter 11% fewer errors.

On the lower end of the spectrum, the tool described in [39] was evaluated with only 7 test participants. They were asked to carry out tasks with pre-made programming tasks using both a visual programming tool and MS Visual Studio. While not statistically significant, the results show that participants completed the tasks at a shorter time with the visual programming tool and rated it more positively with respect to ease of learning.

To sum up, determining the number of test participants is not a straightforward task. An important consideration is the purpose of the evaluation. For discovering the number of usability problems, Macefield recommends between 3 and 20 participants, with 5 to 10 being a good baseline [82].

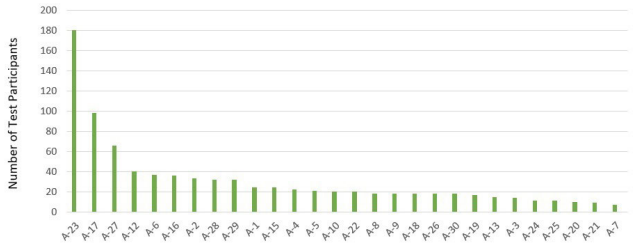

FIGURE 11. Number of users who participated in the evaluation of the tools discussed in the articles.

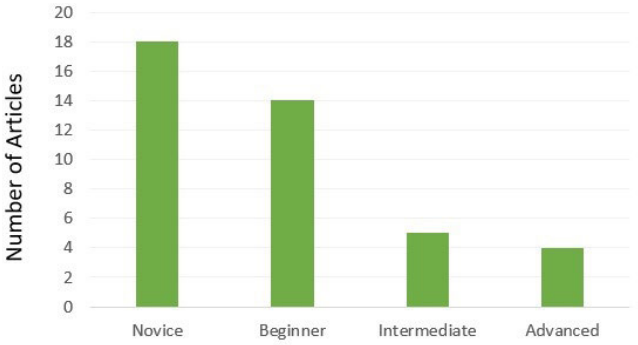

FIGURE 12. Programming skills of users who participated in evaluation of the tools discussed in the articles.

Nielsen argues that generally 5 test participants can be sufficient to discover most of the usability problems [85]. The more users are added, the less can be learned.

According to Macefield, for comparative studies, group sizes of between 8 and 25 participants typically provide valid results, with 10 to 12 being a good baseline [82]. For statistically significant results, group sizes should be increased.

D. RQ2-D4: PROGRAMMING SKILLS OF TEST PARTICIPANTS

We classify the programming skills into four categories: Novice, beginner, intermediate, and advanced. *Novices* have never studied programming. They never wrote any code and they are not aware of basic programming concepts such as variables, functions, etc. *Beginners* have basic programming knowledge such as variables and functions. Their knowledge corresponds to the programming concepts discussed in the first programming course in an undergraduate program. End users with *intermediate* programming skills have taken multiple programming courses, and they may have knowledge of data structures such as queues, stacks, as well as algorithms. Their knowledge corresponds to the programming skills of a third or fourth-year undergraduate computer science student. End users with *advanced* programming skills have professional programming experience. For instance, they may have worked as developers, or they write code as part of their profession.

Naturally, most of the articles (18 out of 30) presented evaluations with novice end users, whereas authors of 14 articles evaluated their tools with beginners. A few articles included intermediate and advanced users in their evaluation. Some

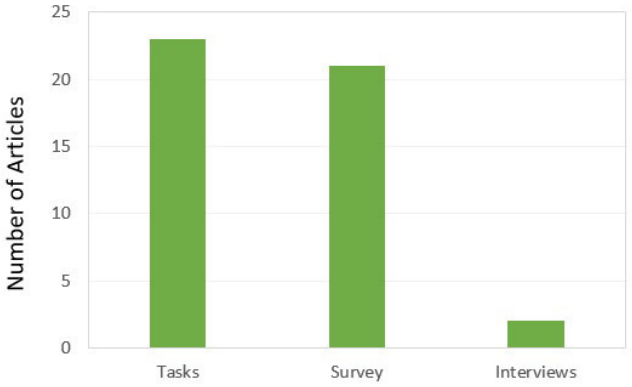

FIGURE 13. Evaluation methods used to evaluate the tools discussed in the articles.

articles such as [47], [59] and [48] evaluated their tools with users of various degrees of programming skills.

We believe it is ideal to evaluate visual programming tools with novices or beginners as they are expected to encounter more conceptual problems compared to intermediate and advanced end users, thereby giving tool authors more valuable feedback.

E. RQ2-D5: EVALUATION METHODS

We classify the evaluation methods into three categories: Task-based, survey-based, and interview-based. *Tasks* are used as hands-on activities where participants are expected to perform a certain functionality. *Surveys* are given to participants to ask their opinions on how they perceive certain qualities (such as usability, usefulness). *Interviews* are generally carried out after test participants perform tasks in order to obtain qualitative data such as end users' understanding of certain concepts. With respect to visual programming tools, the two most interesting metrics to measure are usability and usefulness of the tool.

Figure 13 shows the evaluation methods used to evaluate the tools discussed in the articles. Naturally, most of the articles (23 out of 30) used tasks to assess the usability of the tools, while the authors of 21 articles used surveys to evaluate the perceived usability, usefulness, and workload. Only two articles used interviews to obtain in-depth information about the understandability of the tool.

Lauesen recommends using hands-on tasks to measure certain factors of usability such as ease of learning, task efficiency, surveys to measure the subjective qualities such as usefulness and user experience, and interviews to evaluate the understandability of the system [67]. As such, in the context of visual programming tools, we believe it's ideal to test the usability of the tools with hands-on tasks, usefulness and user experience with a survey, and understandbility of the tools with an interview.

F. RQ2-D6: EVALUATION MEASURES

We classify the evaluation measures into five categories: Completion Time, Number of errors, Perceived usability, Perceived usefulness, and Perceived workload. *Completion*

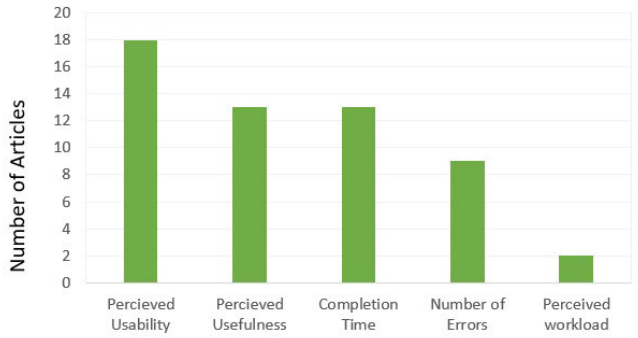

FIGURE 14. Evaluation measures used to evaluate the tools discussed in the articles.

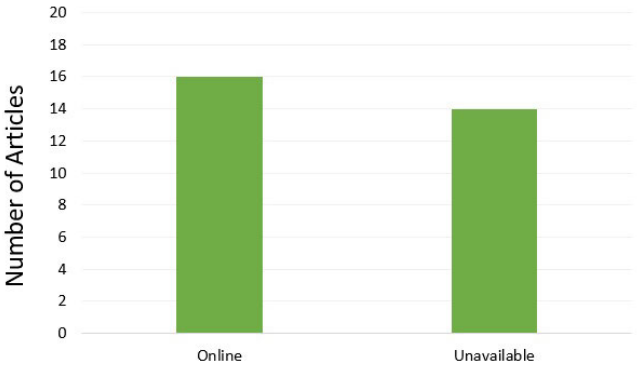

FIGURE 15. Accessibility of the tools discussed in the articles.

time is the time it takes participants to complete a task. This metric measures ease of learning and task efficiency [67], [86]. *Number of errors* affects ease of learning. The more errors users or problems encounter, the harder the tool is to learn [67], [86]. *Perceived usability and usefulness* are what participants feel about the usability and usefulness of the system. Naturally, measuring perceived usability and usefulness is not as accurate as measuring completion time and number of errors, as it is subjective, but it gives some indications. *Perceived workload* is the perception of participants of the amount of work it requires to perform certain tasks with the tool. This metric correlates with task efficiency.

Figure 14 shows the evaluation measures used to evaluate the tools discussed in the articles. Most of the articles (18 out of 30) measured perceived usability, while the authors of 13 articles measured perceived usefulness and completion time. Only 9 articles measured the number of errors, while 2 articles measured the perceived workload.

We strongly recommend measuring the usability factors of ease of learning and task efficiency using completion time and number of errors as they are objective and quantitative. Measuring the perceived usability, usefulness, and workload using surveys provide some indication, but they remain subjective [67].

G. RQ2-D7: ACCESSIBILITY

To better analyze the tools presented in the selected articles, we attempted to find the tools to experiment with them and get a feel for how they work in practice. 53.3% (16) of the tools such as [41], [43] and [44] are available online. Most of these tools are available on GitHub [99] with decent documentation on how to run the tools. Others are available on the authors' personal websites as binary files that can be installed. 14 tools are unfortunately unavailable. It is imperative for authors to make tools easily accessible coupled with documentation beyond installation instructions. This would be beneficial for authors in terms of receiving valuable feedback, researchers to obtain an in-depth understanding of the tool approach, and for users who could potentially use it in their domain. We believe tool availability and documentation ensure its sustainability.

VIII. CHALLENGES AND FUTURE RESEARCH DIRECTIONS

Despite the considerable interest in novel approaches to visual programming, progress still needs to be achieved to realize the potential of visual programming. In this section, we provide a discussion of challenges in relation to the various visual programming approaches presented in the articles. We also provide an insight into future research directions.

A. USABILITY PRINCIPLES

Usability is a quality feature that assesses how easy a user interface is to use. Usability can be broken into five elements: learnability, efficiency, memorability, subjective satisfaction, and error recoverability [64]. There is a plethora of usability principles that can serve as guidance for designing user interfaces. For instance, Nielson identified ten general heuristics that are broad rules of thumb [63]. Further, Shneiderman listed eight golden rules of user interface design [65]. Another example of general heuristics used to guide the design of user interfaces is the framework of cognitive dimensions of notations [66]. In terms of the design phase, it is recommended to design user interfaces iteratively by involving users during the design phase [67], [68].

The goal of the tools discussed in the selected articles is to make application development accessible to end users. Nevertheless, none of the tools explicitly articulated the reliance on usability principles in the design phase. However, it could be argued that some of the authors designed the tools with usability in mind based on some design choices. For instance, the tool presented in [32] uses a user interface that is rather consistent with users' expectations as it is based on a typical visual builder [76] where users drag and drop objects. The authors of [51] divided the mobile screen into columns to allow the user sequential and parallel composition of actions. This conforms with the Juxtaposability dimension described in [66]. As another example, the authors of [33] and [50] involved the users in the design phase, collected feedback, and improved the design.

Despite not explicitly following usability heuristics during the design phase, many of the authors of the selected articles conducted usability studies after building the tools. For instance, a usability study is documented in [43]. The study was conducted in the lab with potential users and real-world use cases. Efficiency, usefulness, quality and satisfaction

were measured, and the results were encouraging. Another example can be found in [44]. The authors conducted a focus group of 20 participants. Usefulness, ease of use and perceived level of understanding were evaluated, and feedback was collected from participants.

B. APPLICATION LIFE CYCLE

The tools surveyed in this study present solutions that could meet computational needs of end-user developers. However, none of the selected articles discussed the life cycle of applications made with the tools. How are the applications maintained, debugged, and extended? This area falls under the umbrella of end-user development. Despite the significant research in end-user development [69]–[71], more work needs to be done with respect to applications developed with visual programming languages. Further, future studies need to investigate how such tools can be adopted in the work environments of end-user developers. For instance, is it easy to scale the solutions developed by the tools? Indeed, some researchers such as [72] indicated that scaling applications created with VPLs remains challenging.

One promising attempt towards providing support for the application life cycle is allowing end users to debug applications built with MIT App Inventor [93]. End users can add watches, comments, run blocks in isolation, collapse a few blocks to keep the screen real-estate small.

C. EXPRESSIVENESS

Expressiveness can be defined as the breadth of ideas that can be communicated with a language. Expressiveness is crucial so that end users develop custom applications beyond the textbook basic examples. Unfortunately, only one of the the surveyed tools was evaluated for expressiveness [46]. Understandably highly expressive languages tend to be less accessible to end users. As such, authors need to strike the balance between usability and expressiveness by providing a usable tool that is expressive enough that can solve the problems of end-user developers in their work environments. There are some attempts in that direction in the field of end-user development. For instance, Aghaee and Pautasso proposed a hybrid approach that balances expressiveness and usability by combining several techniques such as natural language programming, what-you-see-is-what-you-get, and live recommendations [18]. Further, Kuhail and Lauesen proposed a tool that utilizes the expressiveness of spreadsheet-like formulas and the accessibility of a visual builder to help end-user developers build custom visualizations [73]. Future tools using visual programming need to tackle the trade-off between expressiveness and usability in its design and evaluation. Further, it would be helpful for end users to access documentation that showcases the expressiveness of the tools by providing several examples of the tool capabilities.

D. EVALUATION FRAMEWORK

The objective of this systematic literature review was to characterize and analyze the current evidence-based visual programming approaches, trends, interaction styles, and techniques. To achieve this objective, we examined 30 articles against 12 relevant dimensions. However, future researchers may develop an evaluation framework that can be used to compare visual programming tools. The research community has contributed such frameworks to be used to compare regular programming languages [84]. While the existing framework of cognitive dimensions of notations [66] can be used to compare tools in terms of their usability, a potentially new evaluation framework is needed to evaluate other quality attributes such as extensibility, security, testability, and portability.

E. TUTORIALS FOR LEARNABILITY IMPROVEMENT

Some articles such as [56] reported the use of a tutorial to train end users prior to participation in an evaluation study. However, none of the articles explored the impact of tutorials on improving tool learnability. Most of the popular end-user development tools such as Microsoft PowerApps [91] and MIT App Inventor [93] have a large community of contributors who develop tutorials. In fact, typing ''Microsoft PowerApp tutorial'' in Google search engine yields 272,000 results. Such tutorials contribute to the popularity and success of such tools. With respect to visual programming, investigating the impact of tutorials on learnability remains a topic for exploration. Moreover, as video-based tutorials are on the rise, it is imperative to investigate whether video-based tutorials are preferred to text-based tutorials by end users to effectively learn visual programming tools. Recent research suggests the effectiveness of multi-media-based tutorials. For instance, Dalal has shown that multi-media tutorials are effective in teaching junior students operating system concepts [80]. Likewise, Some researchers such as [79] found out that video-based tutorials were more effective than paper-based ones in training users to use software.

Another related direction for future research could be using gamification in conjunction with effective tutorials. *Gamification* is the use of game elements such as leaderboards and badges in non-game contexts such as education and marketing. There is plenty of evidence that suggests the effectiveness of gamification in education, particularly for teaching STEM-related topics [83]. Consequently, investigating the combination of gamification and effective tutorials on tool learnability could be a topic to explore by future researchers.

F. UTILIZATION OF MACHINE LEARNING

Recent advances in machine learning have paved the way for new possibilities such as voice recognition and conversational interaction with computers. To explore the potential of natural language for programming, Van Brummelen et al developed a tool that uses a conversational agent to enable end users to develop applications [74]. The tool receives and transcribes users' spoken input and responds using a dialog manager. As another example, Paschoal et al proposed using conversational agents to support software testing education [75].

Machine learning algorithms are increasingly being used in several aspects of software development. As an example, Ubisoft [87], a video gaming industry utilizes machine learning for code reviewing. It is estimated that 70% of the bugs are caught by machine learning prior to testing [89]. As another example, Functionize, a recently founded company, uses machine learning to generate automatic test cases that adapt to new scenarios [88]. However, in the context of visual programming, more investigation is needed to identify how machine learning can understand the intent of the code and look for logical errors, thereby suggesting code fixes.

IX. LIMITATIONS OF THE STUDY

We identified a few limitations that affect our study. We limited our research to the period January 2010 to November 2020. We needed to limit our paper search to November 2020, to be able to realistically proceed with the analysis of articles, which required several months. There are probably other published articles that could be of interest for this study at the date of submission.

Our initial search resulted in a total of 550 articles. Exclusion criteria were applied to identify relevant articles that were practical to assess. As such, this decision might have resulted in a bias: for instance, we could have excluded original ideas in short papers not presenting evidence.

Article classification may have been inaccurate, as analysis has been conducted by different researchers, who have different experience in the research area discussed in this study. To mitigate this risk, during the application of exclusion criteria, the work done by each reviewer was crosschecked in order to avoid that some relevant article was incorrectly excluded. Further, after the analysis of each selected paper, all gray areas and doubts were discussed at least by two researchers.

We only selected articles that presented empirical evidence; causing a bias towards selecting articles that were empirically evaluated. Additionally, we decided to only analyze the most recent articles when multiple articles reported on the same problem by the same authors.

Finally, it is possible that there are articles that report a visual programming tool that could not be found in the search databases we selected. In order to reduce this risk, we conducted a manual search to find significant work beyond what we found in the search databases. However, the articles found by manual search were already included in our set of articles.

X. CONCLUSION

This study characterized how various visual programming approaches empower end users to develop applications. The study analyzed 30 VPL tools proposed in the literature. To analyze the tools, the study examined how each tool fares across 12 dimensions: VPL classification, interaction style, target users, domain, platform, type of empirical evaluation, types of test participants, number of test participants, programming skills of test participants, evaluation methods, evaluation measures, and accessibility of visual programming tools.

All the tools the study surveyed were domain specific. The results show that the tools were proposed in various areas including IoT, education, robotics, and more, and targeted mostly general users as well as students and domain experts.

Despite the difference of VPL approaches, most tools utilize high-level abstractions to hide implementation details, and use similar interaction styles such as direct manipulation (in the form of drag-drop) and menu selection. Some commonalities have been observed among tools of the same domain. For instance, tools that target IoT allow end users to access services using visual abstractions, while tools built for educational purposes mostly use a block-based approach, and use similar visual abstractions representing programming constructs.

Most of the tools produced some empirical evidence that points to the success of their approaches with end users, though only a few of the tools conducted a formal experiment. Moreover, the test subjects were mostly students with little to no programming skills in a university setting as opposed to typical users in a real setting. To evaluate the tools, most authors used tasks or surveys. Further, most of the tools measured perceived usability using surveys, while fewer tools measures usability factors such as ease of learning and task efficiency using time completion and number of errors.

There are several challenges to be addressed by future tools. None of the tool authors explicitly mentioned the reliance of usability principles in designing the tools, though a few authors performed usability tests to evaluate the tools. Further, only one of the tools was evaluated the expressiveness of their tools. It is imperative for end users to understand the limits of the tools. Finally, 53.3% (16) of the tools were available publicly with some documentation. We strongly recommend that future tools are made available for end users as well as comprehensive documentation to ensure tool adoption and sustainability.

This study identified several areas for further investigation. Future tools should examine how end user developers extend, debug, and deploy applications created with visual programming tools throughout the application life cycle. Further, future researchers may need to develop an evaluation framework to conduct a comparative analysis of VPL tools, particularly for quality attributes such as security and extensibility. Moreover, we recommend the study of the impact of tutorials on improving visual programming tool learnability. Finally, it is imperative to explore conversational agents and machine learning algorithms to assist end-user developers to develop and debug programs created with visual programming tools.

REFERENCES

[1] B. A. Myers, "Taxonomies of visual programming and program visualization," *J. Vis. Lang. Comput.*, vol. 1, no. 1, pp. 97–123, Mar. 1990.
[2] M. M. Burnett and M. J. Baker, "A classification system for visual programming languages," *J. Vis. Lang. Comput.*, vol. 5, no. 3, pp. 287–300, Sep. 1994.

[3] D.-Q. Zhang and K. Zhang, "On the design of a generic visual programming environment," in *Proc. IEEE Symp. Vis. Lang.*, Sep. 1998, p. 88–89, doi: 10.1109/VL.1998.706147.
[4] J. M. Mota, I. Ruiz-Rube, J. M. Dodero, and I. Arnedillo-Sánchez, "Augmented reality mobile app development for all," *Comput. Electr. Eng.*, vol. 65, pp. 250–260, Jan. 2018, doi: 10.1016/j.compeleceng.2017.08.025.
[5] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Keele, U.K., Joint Rep. EBSE-2007-01, 2007.
[6] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, and S. Wiedenbeck, "The state of the art in end-user software engineering," *ACM Comput. Surveys*, vol. 43, no. 3, pp. 1–44, Apr. 2011, doi: 10.1145/1922649.1922658.
[7] M. Santos, and M. Villela, "Characterizing end-user development solutions: A systematic literature review," in *Human-Computer Interaction. Perspectives on Design* (Lecture Notes in Computer Science), vol. 11566, M. Kurosu, Ed. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-22646-6_14.
[8] E. Coronado, F. Mastrogiovanni, B. Indurkhya, and G. Venture, "Visual programming environments for end-user development of intelligent and social robots, a systematic review," *J. Comput. Lang.*, vol. 58, Jun. 2020, Art. no. 100970, doi: 10.1016/j.cola.2020.100970.
[9] B. Jost, M. Ketterl, R. Budde, and T. Leimbach, "Graphical programming environments for educational robots: Open roberta-yet another one?" in *Proc. IEEE Int. Symp. Multimedia*, Dec. 2014, pp. 381–386, doi: 10.1109/ISM.2014.24.
[10] P. P. Ray, "A survey on visual programming languages in Internet of Things," *Sci. Program.*, vol. 2017, May 2017, Art. no. 1231430, doi: 10.1155/2017/1231430.
[11] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, "End-user development: An emerging paradigm," in *End User Development* (Human-Computer Interaction Series), vol. 9. Dordrecht, The Netherlands: Springer, 2006, doi: 10.1007/1-4020-5386-X_1.
[12] M. M. Burnett and C. Scaffidi, "End-user development," in *The Encyclopedia Human-Computer Interaction*, 2nd ed. Aarhus, Denmark: Interaction Design Foundation, 2013. [Online]. Available: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed
[13] A. Repenning, "Moving beyond syntax: Lessons from 20 years of blocks programing in AgentSheets," *J. Vis. Lang. Sentient Syst.*, vol. 3, no. 1, pp. 68–91, Jul. 2017, doi: 10.18293/vlss2017-010.
[14] S. K. Chang, "Visual languages: A tutorial and survey," in *Visualization in Programming, Interdisciplinary Workshop on Informatics and Psychology*. Berlin, Germany: Springer-Verlag, 1987, pp. 1–23.
[15] S. K. Chang, "Icon semantics—A formal approach to icon system design," *Int. J. Pattern Recogn. Artif. Intell.*, vol. 1, no. 1, Apr. 1987, pp. 103–120, doi: 10.1142/S0218001487000084.
[16] U. Wajid, A. Namoun, and N. Mehandjiev, "Alternative representations for end user composition of service-based systems," in *Proc. Int. Symp. End User Develop.* (Lecture Notes in Computer Science), vol. 6654. Berlin, Germany: Springer, 2011, doi: 10.1007/978-3-642-21530-8_6.
[17] C. Ardito, M. Francesca Costabile, G. Desolda, R. Lanzilotti, M. Matera, A. Piccinno, and M. Picozzi, "User-driven visual composition of service-based interactive spaces," *J. Vis. Lang. Comput.*, vol. 25, no. 4, pp. 278–296, Aug. 2014, doi: 10.1016/j.jvlc.2014.01.003.
[18] S. Aghaee and C. Pautasso, "End-user development of mashups with *naturalmash*," *J. Vis. Lang. Comput.*, vol. 25, no. 4, pp. 414–432, Aug. 2014, doi: 10.1016/j.jvlc.2013.12.004.
[19] M. R. Reisinger, J. Schrammel, and P. Frohlich, "Visual languages for smart spaces: End-user programming between data-flow and form-filling," in *Proc. IEEE Symp. Vis. Lang. Hum.-Centric Comput. (VL/HCC)*, Raleigh, NC, USA, Oct. 2017, pp. 165–169, doi: 10.1109/VLHCC.2017.8103464.
[20] A. L. Ambler, M. M. Burnett, and B. A. Zimmerman, "Operational versus definitional: A perspective on programming paradigms," *Computer*, vol. 25, no. 9, pp. 28–43, Sep. 1992, doi: 10.1109/2.156380.
[21] G. Rothermel, L. Li, C. DuPuis, and M. Burnett, "What you see is what you test: A methodology for testing form-based visual programs," in *Proc. 20th Int. Conf. Softw. Eng.*, Kyoto, Japan, 1998, pp. 198–207, doi: 10.1109/ICSE.1998.671118.
[22] K. Stenning and J. Oberlander, "A cognitive theory of graphical and linguistic reasoning: Logic and implementation," *Cognit. Sci.*, vol. 19, no. 1, pp. 97–140, Jan. 1995, doi: 10.1207/s15516709cog1901_3.
[23] J. H. Larkin and H. A. Simon, "Why a diagram is (Sometimes) worth ten thousand words," *Cognit. Sci.*, vol. 11, no. 1, pp. 65–100, Jan. 1987, doi: 10.1111/j.1551-6708.1987.tb00863.x.
[24] J. M. Carroll, J. C. Thomas, and A. Malhotra, "Presentation and representation in design problem-solving," *Brit. J. Psychol.*, vol. 71, no. 1, pp. 143–153, Feb. 1980, doi: 10.1111/j.2044-8295.1980.tb02740.x.
[25] J. Chattratichart, "Exploring the effect of control-flow and traversal direction on VPL usability for novices," *J. Vis. Lang. Comput.*, vol. 13, no. 5, pp. 471–500, 2002, doi: 10.1006/jvlc.2002.0240.
[26] F. Paternò, "End user development: Survey of an emerging field for empowering people," *ISRN Softw. Eng.*, vol. 2013, Jun. 2013, Art. no. 532659, doi: 10.1155/2013/532659.
[27] D. Tetteroo and P. Markopoulos, "A review of research methods in end user development," in *End-User Development* (Lecture Notes in Computer Science), vol. 9083. Cham, Switzerland: Springer, 2015, doi: 10.1007/978-3-319-18425-8_5.
[28] F. Hang and L. Zhao, "Supporting end-user service composition: A systematic review of current activities and tools," in *Proc. IEEE Int. Conf. Web Services*, New York, NY, USA, Jun. 2015, pp. 479–486, doi: 10.1109/ICWS.2015.70.
[29] B. R. Barricelli, F. Cassano, D. Fogli, and A. Piccinno, "End-user development, end-user programming and end-user software engineering: A systematic mapping study," *J. Syst. Softw.*, vol. 149, pp. 101–137, Mar. 2019, doi: 10.1016/j.jss.2018.11.041.
[30] M. Noone and A. Mooney, "Visual and textual programming languages: A systematic review of the literature," *J. Comput. Educ.*, vol. 5, pp. 149–174, 2018, doi: 10.1007/s40692-018-0101-5.
[31] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," *Inf. Softw. Technol.*, vol. 53, no. 6, pp. 625–637, Jun. 2011, doi: 10.1016/j.infsof.2010.12.010.
[32] B. A. Johnsson and B. Magnusson, "Towards end-user development of graphical user interfaces for Internet of Things," *Future Gener. Comput. Syst.*, vol. 107, pp. 670–680, Jun. 2020, doi: 10.1016/j.future.2017.09.068.
[33] G. Huang, P. S. Rao, M.-H. Wu, X. Qian, S. Y. Nof, K. Ramani, and A. J. Quinn, "Vipo: Spatial-visual programming with functions for robot-IoT workflows," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2020, pp. 1–13, doi: 10.1145/3313831.3376670.
[34] S. Sorce, A. Malizia, V. Gentile, P. Jiang, M. Atherton, and D. Harrison, "Evaluation of a visual tool for early patent infringement detection during design," in *Proc. Int. Symp. End User Develop.*, in Lecture Notes in Computer Science, vol. 11553. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-24781-2_12.
[35] D. Lizcano, F. Alonso, J. Soriano, and G. López, "A component- and connector-based approach for end-user composite Web applications development," *J. Syst. Softw.*, vol. 94, pp. 108–128, Aug. 2014, doi: 10.1016/j.jss.2014.03.039.
[36] A. Merkouris, K. Chorianopoulos, and A. Kameas, "Teaching programming in secondary education through embodied computing platforms: Robotics and wearables," *ACM Trans. Comput. Edu.*, vol. 17, no. 2, pp. 1–22, Jun. 2017.
[37] A. Feng, M. Gardner, and W.-C. Feng, "Parallel programming with pictures is a snap!" *J. Parallel Distrib. Comput.*, vol. 105, pp. 150–162, Jul. 2017, doi: 10.1016/j.jpdc.2017.01.018.
[38] H. Kunimune, S. Kamijima, T. Yamamoto, and M. Niimura, "Trial to increase motivation on programming by using hardware control functions in the at visual programming environment," in *Proc. 2nd Int. Conf. Edu. Technol. Manage. (ICETM)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 50–53, doi: 10.1145/3375900.3375917.
[39] K. Abe, Y. Fukawa, T. Tanaka, "Prototype of visual programming environment for C language novice programmer," in *Proc. 8th Int. Congr. Adv. Appl. Informat. (IIAI-AAI)*, Toyama, Japan, 2019, pp. 140–145, doi: 10.1109/IIAI-AAI.2019.00037.
[40] A. Sanna, F. Lamberti, F. Bazzano, L. Maggio, "Developing touch-less interfaces to interact with 3D contents in public exhibitions," in *Proc. Int. Conf. Augmented Reality, Virtual Reality Comput. Graph.*, 2016, pp. 293–303.
[41] S. G. Tamilselvam, N. Panwar, S. Khare, R. Aralikatte, A. Sankaran, and S. Mani, "A visual programming paradigm for abstract deep learning model development," in *Proc. 10th Indian Conf. Hum.-Comput. Interact. (IndiaHCI)*. New York, NY, USA: Association for Computing Machinery, 2019, Art. no. 16, doi: 10.1145/3364183.3364202.
[42] A. Cannavò, F. D. Pace, F. Salaroglio, and F. Lamberti, "A visual editing tool supporting the production of 3D interactive graphics assets for public exhibitions," *Int. J. Hum.-Comput. Stud.*, vol. 141, Apr. 2020, Art. no. 102450, doi: 10.1016/j.ijhcs.2020.102450.

[43] S. Valtolina, F. Hachem, B. R. Barricelli, E. G. Belay, S. Bonfitto, and M. Mesiti, "Facilitating the development of iot applications in smart city platforms," in *End-User Development* (Lecture Notes in Computer Science), vol. 11553. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-24781-2_6.

[44] A. Rao, A. Bihani, and M. Nair, "Milo: A visual programming environment for data science education," in *Proc. IEEE Symp. Vis. Lang. Hum.-Centric Comput. (VL/HCC)*, Lisbon, Portugal, Oct. 2018, pp. 211–215, doi: 10.1109/VLHCC.2018.8506504.

[45] G. De Luca, Z. Li, S. Mian, and Y. Chen, "Visual programming language environment for different IoT and robotics platforms in computer science education," *CAAI Trans. Intell. Technol.*, vol. 3, no. 2, pp. 119–130, Jun. 2018, doi: 10.1049/trit.2018.0016.

[46] N. Bak, B.-M. Chang, and K. Choi, "Smart Block: A visual block language and its programming environment for IoT," *J. Comput. Lang.*, vol. 60, Oct. 2020, Art. no. 100999, doi: 10.1016/j.cola.2020.100999.

[47] H. Mei, W. Chen, Y. Ma, H. Guan, W. Hu, "VisComposer: A visual programmable composition environment for information visualization," *Vis. Inform.*, vol. 2, no. 1, pp. 71–81, 2018, doi: 10.1016/j.visinf.2018.04.008.

[48] P. Valderas, V. Torres, I. Mansanet, and V. Pelechano, "A mobile-based solution for supporting end-users in the composition of services," *Multimedia Tools Appl.*, vol. 76, no. 15, pp. 16315–16345, Aug. 2017, doi: 10.1007/s11042-016-3910-4.

[49] B. Broll, Á. Lédeczi, H. Zare, D. N. Do, J. Sallai, P. Völgyesi, M. Maróti, L. Brown, and C. Vanags, "A visual programming environment for introducing distributed computing to secondary education," *J. Parallel Distrib. Comput.*, vol. 118, 2018, pp. 189–200, doi: 10.1016/j.jpdc.2018.02.021.

[50] A. Stratton, C. Bates, and A. Dearden, "Quando: Enabling museum and art gallery practitioners to develop interactive digital exhibits," in *End-User Development* (Lecture Notes in Computer Science), vol. 10303. Cham, Switzerland: Springer, 2047, doi: 10.1007/978-3-319-58735-6_7.

[51] R. Francese, M. Risi, and G. Tortora, "Iconic languages: Towards end-user programming of mobile applications," *J. Vis. Lang. Comput.*, vol. 38, pp. 1–8, Feb. 2017, doi: 10.1016/j.jvlc.2016.10.009.

[52] L. Thamsen, T. Renner, M. Byfeld, M. Paeschke, D. Schroder, and F. Bohm, "Visually programming dataflows for distributed data analytics," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Washington, DC, USA, Dec. 2016, pp. 2276–2285, doi: 10.1109/BigData.2016.7840860.

[53] S. Alexandrova, Z. Tatlock, and M. Cakmak, "RoboFlow: A flow-based visual programming language for mobile manipulation tasks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 5537–5544, doi: 10.1109/ICRA.2015.7139973.

[54] D. Rough and A. Quigley, "Jeeves—A visual programming environment for mobile experience sampling," in *Proc. IEEE Symp. Vis. Lang. Hum.-Centric Comput. (VL/HCC)*, Atlanta, GA, USA, Oct. 2015, pp. 121–129, doi: 10.1109/VLHCC.2015.7357206.

[55] F. CabitzaIade and G. Gesso, "Rule-based programming as easy as a child's play. A user study on active documents," in *Proc. Int. Conf. Interface Hum. Comput. Interact. (IHCI IADIS)*, Lisbon, Portugal, Jul. 2012, pp. 73–80.

[56] T. Booth and S. Stumpf, "End-user experiences of visual and textual programming environments for Arduino," in *End-User Development* (Lecture Notes in Computer Science), vol. 7897. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-642-38706-7_4.

[57] T. N. Luong, P. Etcheverry, C. Marquesuzaà, and T. Nodenot, "A visual programming language for designing interactions embedded in Web-based geographic applications," in *Proc. ACM Int. Conf. Intell. User Interfaces (IUI)*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 207–216, doi: 10.1145/2166966.2167003.

[58] I. Weber, H.-Y. Paik, and B. Benatallah, "Form-based Web service composition for domain experts," *ACM Trans. Web*, vol. 8, no. 1, Dec. 2013, Art. no. 2, doi: 10.1145/2542168.

[59] T. Turchi, A. Malizia, A. Dix, "TAPAS: A tangible end-user development tool supporting the repurposing of pervasive displays," *J. Vis. Lang. Comput.*, vol. 39, pp. 66–77, Apr. 2017, doi: 10.1016/j.jvlc.2016.11.002.

[60] J. Danado and F. Paternò, "Puzzle: A mobile application development environment using a jigsaw metaphor," *J. Vis. Lang. Comput.*, vol. 25, no. 4, pp. 297–315, Aug. 2014, doi: 10.1016/j.jvlc.2014.03.005.

[61] W.-K. Chen and P.-Y. Tu, "VisualTPL: A visual dataflow language for report data transformation," *J. Vis. Lang. Comput.*, vol. 25, no. 3, pp. 210–226, Jun. 2014, doi: 10.1016/j.jvlc.2013.11.003.

[62] Z. Drey and C. Consel, "Taxonomy-driven prototyping of home automation applications: A novice-programmer visual language and its evaluation," *J. Vis. Lang. Comput.*, vol. 23, no. 6, pp. 311–326, Dec. 2012, doi: 10.1016/j.jvlc.2012.07.002.

[63] J. Nielsen, "Heuristic evaluation," in *Usability Inspection Methods*, J. Nielsen and R. L. Mack, Eds. New York, NY, USA: Wiley, 1994.

[64] J. Nielsen, *Usability Engineering*, 1st ed. San Mateo, CA, USA: Morgan Kaufmann, Sep. 1993.

[65] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 6th ed. London, U.K.: Pearson, May 2016. [Online]. Available: http://www.cs.umd.edu/hcil/DTUI6

[66] T. R. G. Green, "Cognitive dimensions of notations," in *People and Computers V*, A. Sutcliffe and L. Macaulay, Eds. Cambridge, U.K.: Cambridge Univ. Press, 1989, pp. 443–460.

[67] S. Lauesen, *User Interface Design: A Software Engineering Perspective*. Reading, MA, USA: Addison-Wesley, 2004.

[68] J. Nielsen, "Iterative user-interface design," *Computer*, vol. 26, no. 11, pp. 32–41, Nov. 1993, doi: 10.1109/2.241424.

[69] M. Burnett, C. Cook, O. Pendse, G. Rothermel, J. Summet, and C. Wallace, "End-user software engineering with assertions in the spreadsheet paradigm," in *Proc. 25th Int. Conf. Softw. Eng.*, Portland, OR, USA, 2003, pp. 93–105.

[70] A. Koesnandar, S. Elbaum, G. Rothermel, L. Hochstein, C. Scaffidi, and K. T. Stolee, "Using assertions to help end-user programmers create dependable Web macros," in *Proc. 16th ACM SIGSOFT Int. Symp. Found. Softw. Eng. (SIGSOFT/FSE)*, Atlanta, GA, USA, 2008, pp. 124–134.

[71] C. Scaffidi, "Topes: Enabling end-user programmers to validate and reformat data," Inst. Softw. Res. (ISR), Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-ISR-09-105, 2009.

[72] R. Schaefer, "On the limits of visual programming languages," *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 2, pp. 7–8, Mar. 2011, doi: 10.1145/1943371.1943373.

[73] M. A. Kuhail and S. Lauesen, "Uvis: A formula-based end-user tool for data visualization," *IEEE Access*, vol. 8, pp. 110264–110278, 2020, doi: 10.1109/ACCESS.2020.3002591.

[74] J. Van Brummelen, C. Yeo, and K. Weng, "Learning to program conversationally: A conversational agent to further democratize programming," in *Proc. 14th Annu. Int. Technol., Educ. Develop. Conf. (INTED)*, 2020, p. 5.

[75] L. N. Paschoal, L. F. Turci, T. U. Conte, and S. R. S. Souza, "Towards a conversational agent to support the software testing education," in *Proc. 33rd Brazilian Symp. Softw. Eng.*, New York, NY, USA, Sep. 2019, doi: 10.1145/3350768.3352456.

[76] B. Myers, S. E. Hudson, and R. Pausch, "Past, present, and future of user interface software tools," *ACM Trans. Comput.-Hum. Interact.*, vol. 7, no. 1, pp. 3–28, Mar. 2000.

[77] M. Burnett, "What is end-user software engineering and why does it matter?" in *End-User Development* (Lecture Notes in Computer Science), vol. 5435, V. Pipek, M. B. Rosson, B. de Ruyter, V. Wulf, Eds. Berlin, Germany: Springer, 2009, doi: 10.1007/978-3-642-00427-8_2.

[78] A. Annie. *The State of Mobile in 2020: The Key Stats You Need to Know*. Accessed: Oct. 4, 2020. [Online]. Available: https://www.appannie.com/en/insights/market-data/state-of-mobile-2020-infographic/

[79] H. van der Meij and J. van der Meij, "A comparison of paper-based and video tutorials for software learning," *Comput. Educ.*, vol. 78, pp. 150–159, Sep. 2014, doi: 10.1016/j.compedu.2014.06.003.

[80] M. Dalal, "Impact of multi-media tutorials in a computer science laboratory course—An empirical study," *Electron. J. e-Learn.*, vol. 12, no. 4, pp. 366–374, 2014.

[81] M. Soegaard, "Interaction styles," in *The Glossary of Human Computer Interaction*. Aarhus, Denmark: Interaction Design Foundation, 2020. Accessed: Oct. 7, 2020. [Online]. Available: https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/interaction-styles.

[82] R. Macefield, "How to specify the participant group size for usability studies: A practitioner's guide," *J. Usability Stud.*, vol. 5, no. 1, pp. 34–45, 2009.

[83] M. Ortiz, K. Chiluiza, and M. Valcke, "Gamification in higher education and stem: A systematic review of literature," in *Proc. Conf., 8th Annu. Int. Conf. Educ. New Learn. Technol. (Edulearn)*, Barcelona, Spain, Jul. 2016, pp. 6548–6558, doi: 10.21125/edulearn.2016.0422.

[84] M. S. Farooq, S. A. Khan, F. Ahmad, S. Islam, and A. Abid, "An evaluation framework and comparative analysis of the widely used first programming languages," *PLoS ONE*, vol. 9, no. 2, Feb. 2014, Art. no. e88941, doi: 10.1371/journal.pone.0088941.

[85] J. Nielsen. (2000). *Why You Only Need to Test with 5 Users*. Nielsen Norman Group. Accessed: Nov. 25, 2020. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[86] J. Nielsen. (2001). *Usability Metrics*. Nielsen Norman Group. [Online]. Available: https://www.nngroup.com/articles/usability-metrics/
[87] *Ubisoft*. Accessed: Sep. 29, 2020. [Online]. Available: https://www.ubisoft.com/
[88] *Functionize: An Intelligent-Based Testing Company*. Accessed: Sep. 29, 2020. [Online]. Available: https://www.functionize.com/
[89] *PMI's Report on Artificial Intelligence*. Accessed: Sep. 29, 2020. [Online]. Available: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/ai-innovators-cracking-the-code-project-performance.pdf?v=acf03326-778f-4e64-925e-70c1149f37ea&sc_lang_temp=en
[90] *Amazon Honey Code*. Accessed: Sep. 21, 2020. [Online]. Available: https://www.honeycode.aws/
[91] *Microsoft PowerApps*. Accessed: Sep. 21, 2020. [Online]. Available: https://powerapps.microsoft.com/en-us/
[92] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.
[93] *App Inventor*. Accessed: Sep. 21, 2020. [Online]. Available: https://appinventor.mit.edu/
[94] *Atooma*. Accessed: Sep. 22, 2020. [Online]. Available: https://atooma.com/
[95] *Microsoft Visual Programming Language*. Accessed: Sep. 24, 2020. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/microsoft-robotics/bb483088(v=msdn.10)
[96] *MIT Open Blocks*. Accessed: Sep. 24, 2020. [Online]. Available: http://web.mit.edu/mitstep/openblocks.html
[97] *Blockly*. Accessed: Sep. 24, 2020. [Online]. Available: https://developers.google.com/blockly
[98] *Arduino*. Accessed: Sep. 24, 2020. [Online]. Available: https://www.arduino.cc/
[99] *GitHub*. Accessed: Sep. 26, 2020. [Online]. Available: https://github.com/
[100] *LabView*. Accessed: Nov. 16, 2020. [Online]. Available: https://www.ni.com/en-lb/shop/labview.html
[101] *Microsoft Word*. Accessed: Nov. 16, 2020. [Online]. Available: https://www.microsoft.com/en/microsoft-365/word
[102] The Forrester *Wave: Low-Code Development Platforms For AD&D Pros, Q1 2019*. Accessed: Nov. 13, 2020. [Online]. Available: https://www.outsystems.com/1/low-code-development-platforms-wave/
[103] *Smart Block*. Accessed: Nov. 20, 2020. [Online]. Available: https://github.com/baknayeon/smartblock
[104] *ThinkML*. Accessed: Nov. 21, 2020. [Online]. Available: https://goodboyanush.github.io/blogs/dlide.html
[105] *Snap4City Platform*. Accessed: Nov. 21, 2020. [Online]. Available: https://www.snap4city.org/
[106] *The Milo Project*. Accessed: Nov. 21, 2020. [Online]. Available: https://github.com/miloide
[107] *Quando—Visual Programming for Digital Interactive Exhibits*. Accessed: Nov. 21, 2020. [Online]. Available: https://github.com/andrewfstratton/quando.
[108] *Flink Visual Programming*. Accessed: Nov. 21, 2020. [Online]. Available: https://github.com/citmp2015/flink-visual-programming
[109] *RoboFlow*. Accessed: Nov. 21, 2020. [Online]. Available: https://github.com/sonyaa/roboflow
[110] *Modkit*. Accessed: Nov. 21, 2020. [Online]. Available: https://modkit.com/
[111] *NetsBlox*. Accessed: Nov. 21, 2020. [Online]. Available: https://netsblox.org/
[112] *VisComposer*. Accessed: Nov. 21, 2020. [Online]. Available: https://github.com/lyt9304/viscomposer
[113] *Enchanting*. Accessed: Nov. 21, 2020. [Online]. Available: http://enchanting.robotclub.ab.ca/tiki-index.php
[114] *Snap!* Accessed: Nov. 21, 2020. [Online]. Available: https://snap.berkeley.edu/snap/snap.html
[115] *TAPAS*. Accessed: Nov. 21, 2020. [Online]. Available: https://github.com/tommasoturchi/TAPAS
[116] *Jeeves ESM Tool*. Accessed: Nov. 21, 2020. [Online]. Available: https://danielrough.net/jeeves-esm-tool/
[117] *WS-BPEL Tool*. Accessed: Dec. 20, 2020. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=bpel4people
[118] *VIPLE Tool*. Accessed: Dec. 22, 2020. [Online]. Available: http://neptune.fulton.ad.asu.edu/VIPLE/
[119] *Scimago Journal and Country Rank*. Accessed: Nov. 21, 2020. [Online]. Available: https://www.scimagojr.com/

MOHAMMAD AMIN KUHAIL (Member, IEEE) received the M.Sc. degree in software engineering from the University of York, in 2006, and the Ph.D. degree in computer science from the IT University of Copenhagen, Denmark, in 2013. He has served as an Assistant Teaching Professor with the University of Missouri–Kansas City, USA, for six years. He joined Zayed University, United Arab Emirates, in 2019. He is currently a Computer Scientist and a Software Engineer with a diverse skill set that spans web development, object-oriented programming, algorithms, usability, and data science. He also serves as an Assistant Professor with Zayed University. His research interests include end-user development, usability analysis, and computer science education.

SHAHBANO FAROOQ received the M.Sc. degree in computer science from the University of Calgary, Canada. She is currently an Instructor with Zayed University, United Arab Emirates. Her research interests include HCI and pedagogical innovation. She also has professional experience in full stack application development.

RAWAD HAMMAD received the M.Sc. degree in cognitive computing from the Goldsmiths University of London, in 2010, and the Ph.D. degree in software engineering from the University of the West of England (UWE), in 2018. He is currently the Programme Leader for M.Sc. degree in computing and information communication technology and a Senior Lecturer in computer science and digital technologies with the University of East London. Prior to coming to the University of East London, he was a Senior Education Solutions Researcher/Analyst with the King's College London; a Researcher with the Centre for Complex Cooperative Systems, UWE, the Manager of the E-Learning Centre, IUG Gaza University, and a Developer and the Team Lead of Logicteca Company, Canada. He has extensive experience on software engineering, technology enhanced learning (TEL), artificial intelligence, and smart health research and practice. He has contributed and led various international projects, published research articles, and has been involved in conference programme committees. His research interests include artificial intelligence, software/requirement engineering, TEL, analytics, the smart cities/IoT, semantics, and enterprise architecture. He led numerous international initiatives, e.g., TRANSFER and SmartTech (including Japan, Germany, Finland, and Middle East).

MOHAMMED BAHJA is currently a Lecturer in computer science with the University of Birmingham. Before joining the University of Birmingham, he has served in the education sector at several universities in U.K., including the University College London (UCL). His research interest includes data science for e-learning applications, including natural language processing for capturing user experience and engagement behavior. He has participated in a variety of multidisciplinary projects, including the EU funded projects of Policy Compass, MINICHIP Decision Support System, and Green Datacentre. Apart from academia, he has strong connections within the ICT industry for both the public and private sphere. He provides robust data science solutions and software consulting services for both the public (including NHS) and the private sphere.