IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 22, 2020, accepted July 3, 2020, date of publication July 9, 2020, date of current version July 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3008321

Curriculum, Teaching and Learning, and Assessments for Introductory Programming Course

ERUM MEHMOOD[1], ADNAN ABID[1], (Member, IEEE), MUHAMMAD SHOAIB FAROOQ[1], (Member, IEEE), AND NAEEM A. NAWAZ[2]

[1]Department of Computer Science, School of Systems and Technology, University of Management and Technology, Lahore 54770, Pakistan
[2]Department of Computer Science (CFY), Umm Al-Qura University, Mecca 24231, Saudi Arabia

Corresponding author: Adnan Abid (adnan.abid@umt.edu.pk)

ABSTRACT Learning to program involves acquisition of various skills including problem solving, fundamental design techniques as well as critical thinking. Generally, most of the novice programmers struggle to develop all these important skill. The research community has addressed the problem in many different ways while involving improvisations in curriculum, pedagogical methods, cognitive aspects, supporting tools, and in designing assessments. This research aims to analyze and synthesize the existing literature in the aforementioned areas. Research articles pertaining to the area of Introductory Programming Courses (IPC) have been found using appropriate search queries, while nearly 60 research articles, published in last ten years, have been carefully selected by employing a systematic filtering process. The scope of this work only covers the research conducted for IPC in higher education. Main findings of this study show that ''solution proposal'' and ''evaluation research'' have been reported as two main research types adopted by these studies. Moreover, pedagogy, language choice and students' performance analysis are the most frequently addressed aspects of IPC; whereas, curriculum contents, assessment design, and teaching/learning through tools have appeared as less addressed aspects of IPC. Furthermore, a taxonomy of IPC has been presented based on the studied literature. Lastly, general considerations and future research directions have been presented for the practitioners and researchers in this area.

INDEX TERMS Introductory programming, higher education, programming education curriculum, language choice, systematic review.

I. INTRODUCTION

Introductory programming generally refers to course introduced to undergraduate students with little or no programming experience. Introductory programming courses (IPCs) are included in various undergraduate degree programs. Program design, data structures, syntax, problem solving, programming logic and design techniques [1] are the basic concepts IPC aims to cover. Final report of Curriculum Guidelines for Undergraduate Degree Programs in Computer Science [2] says that introductory courses differ across institutions to help students with significant variance in pre-requisite. Some introductory courses are developed to deliver a broader introduction to computing concepts without the limitations of learning the syntax of a programming language, according to this report. Dropout rates are still high despite progressing in methods/tools for teaching and learning IPC [3]. Moreover, suitability of a language, as IPC, needs to be evaluated in order to analyse challenges faced by novice students. Impact of language choice in learning IPC is evaluated by developing a framework in a study by [4]. Despite having several published reviews in IPC domain, studies either cover papers up to 2017 or focus more on learning methods and less on curricula and assessment design for IPC as mentioned in Table 1. This table provides comparison among existing reviews based on five major perspectives: quality assessment scoring, curriculum, teaching and learning, assessment and targeted digital repositories. We have included comparison for only reviews published in quality journals (excluding conferences

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

TABLE 1. Comparison with related works.

					IPC perspectives addressed							
					Curriculum		Teaching & Learning		Assessment			
Paper	Focus of Survey	Newest Ref.	Survey Approach	Quality Assessment Scored	Language Choice	Content	Pedagogy	Tool	Design	Tool	Student Performance Analysis	Targeted Digital Repositories
[5]	Teaching programming using robots to novices	2012	Systematic search and snowballing	✓	✓	x	✓	✓	✓	x	x	9
[8]	Generic program visualization systems	2013	Informal	x	x	x	✓	✓	x	x	x	Not mentioned
[7]	Students Misconceptions and Other Difficulties in Introductory Programming	2017	Informal	x	x	x	✓	✓	x	x	x	Not mentioned
[3]	Challenges faced by students and teachers during IPC	2018	Systematic search and snowballing	x	✓	x	✓	x	x	x	✓	5
[6]	Blended learning models applied in IPC	2019	Systematic search	✓	x	x	✓	x	x	x	x	9
This survey	Research on challenges related to curricula and language choice for learning IPC	2020	Systematic search, snowballing and assessment	✓	✓	✓	✓	✓	✓	✓	✓	11

and workshops) in our study as more matured research gets published in journals. This comparison help us build the need of this survey.

This systematic literature review (SLR) provides a detailed examination of problems faced by novice programmers in learning introductory programming education and research trends in IPC while covering all five major perspectives shown in Table 1. Based on the systematic review criteria, 60 research papers have been finalized for further review and analysis. The selected papers are empirically and qualitatively evaluated through multiple aspects. The novelty of our systematic literature review is that it provides a new classification criteria, IPC research targeting channels, IPC curricula [2], [49], programming language choice [4], [24], teaching and learning approaches [29], [45], assessment design [19], [21] and tools [23], [57], and approaches to address challenges faced by novice students after validating programming studies empirically. This SLR will help educators in development of standardized IPC learning environment together with curricula, teaching and learning tools, and effective assessment tools and design.

This article is arranged as follows: Section II discusses about existing relevant surveys, and provides a motivation for this SLR. Section III presents adopted methodology to conduct this survey together with objectives and research questions. Answers to these research questions have been described and analyzed in Section IV. Section V presents synthesis of reviewed literature by describing a taxonomy in this domain. Lastly, the article has been concluded in Section VI.

II. LITERATURE REVIEW

Most of the surveys and systematic reviews on introductory programming do not cover publication channels, curricula standards and language choice as IPC, and focus more on teaching methods and tools than on student problems. A more recent systematic review on challenges faced by novice programmers in learning introductory programming course evaluated studies till year 2016 conducted by [3]. Authors focus on analyzing three stages of computational thinking (problem formulation, solution expression and solution execution and evaluation). This is the only other survey to our knowledge that thoroughly presents key issues for the research road map on introductory programming learning and teaching in higher education. However, this study does not focus on other factors playing important role in learning IPC such as: curricula standards and language choice as IPC.

Another SLR presented by [5] investigates the effectiveness of using robots as teaching tools for IPC. Authors discuss different technologies appearing helpful for learners to overcome existing barriers in this context. This SLR discovers that most frequently adopted programming by educators is JAVA for those who use robots as a teaching tool. According to this study, high quality and large scale research is still required in order to discover true efficacy of robots as tool for teaching programming. However, other tools and methodologies of learning IPC are not discussed in this survey.

Models of learning styles for IPC are recently reviewed by [6]. Total of five learning approaches appeared in this review: online self-paced, face-to-face instructor, online collaboration, face-to-face collaboration and online instructor-led. Effectiveness of various learning approaches have been evaluated on basis of students' performance when blended together. Impact of five blended learning models on the learning experience of novice programmers have been reviewed in this survey which are: flipped, mixed, flex,

supplemental and online practicing. Mixed model appeared to be the most appropriate learning model according to this review for enhancing students' performance. This review, however, focused only of learning models not on curricula, and assessment for IPC.

There have been surveys on other aspects of programming languages. One of these surveys is review of literature by [7] which investigated the factors that contribute to the difficulties faced by students in learning introductory programming. Another survey on program visualization tools for learning introductory programming education is done by [8]. However, none of them focus on channels publishing literature on IPC and problems faced by novice students required to achieve desired solutions.

Our review distinguishes itself from the above reviews by focusing only on the publication channels related to IPC, closely examining the IPC curricula, and identifying the approaches addressing challenges faced by novice students. Furthermore, we conduct a more balanced and comprehensive approach than all of the above reviews: we select tools in a systematic way following strict criteria, and we code them using a predetermined labelling.

III. RESEARCH METHODOLOGY

Guidelines for systematic reviews presented in software engineering research by [9] and [10] are followed by our survey. Based on these guidelines, a search protocol was listed after finalizing research questions to reduce possibility of any research bias. According to this protocol, we have included three main stages in our research methodology: plan, conduct and report of review, presented in next sections.

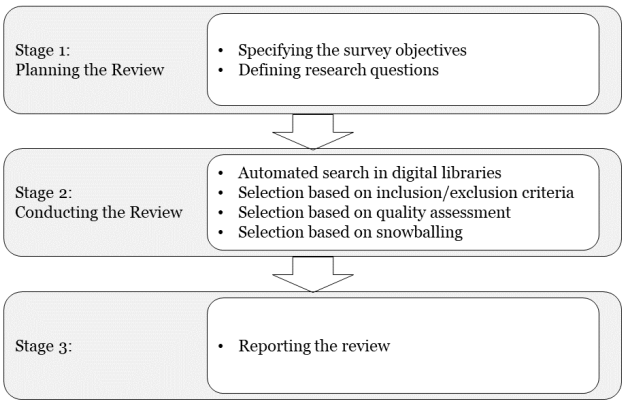

FIGURE 1. Research strategy.

A. REVIEW PLAN

Adequate search strategy have been developed to find all relevant studies. Figures1 and 2 show the research methodology, which demonstrate search process for relevant publications, definition of a classification scheme, and mapping of publications. A highly structured process has been followed in this review that involved:

- Research objectives
- Specifying research questions(RQs)
- Organizing searches of databases
- Studies selection
- Screening relevant studies
- Data extraction
- Results synthesising
- Finalizing the review report

It is important to formulate primary RQs (see Table 2) in order to achieve following core objectives:

a) RQ1 attempts to address our objective to develop a library of articles related to the challenges faced by novice while learning IPC focusing on curricula and language choice, and make this dataset available to other researchers. Moreover, to identify more significant work that provides direction to investigate students' problems in programming education. Answer of RQ1 will discover gaps in terms of research approaches and challenges in the state-of-the-art.

b) Another objective to identify complexities faced by novices based on the selection of any particular language and curriculum while learning IPC will be achieved during assessment of RQ2.

c) The objective to classify existing solutions addressing problems in teaching/learning an IPC for novice programmers, and identify their solutions will be addressed by answer of RQ3.

d) RQ4 attempts to achieve another objective: to identify effectiveness of existing tools developed for teaching and learning IPC.

e) RQ5 tends to determine the recently developed methodologies to analyze and validate students' performance in learning IPC in terms of assessment tools, design and students' performance analysis.

B. REVIEW CONDUCT

The process of conducting this review has been articulated in four steps presented below. In first step, relevant primary studies have been searched from most commonly used digital libraries. Selection of studies based on pre-defined inclusion/exclusion criteria has been performed during second step. We have designed quality assessment criteria to further enhance quality of our review described in third step. Backward snowballing is then performed to extract important candidate papers during final fourth step.

1) AUTOMATED SEARCH IN DIGITAL LIBRARIES

A systematic research has been carried out to filter irrelevant studies and extract appropriate information. Therefore, automatic and manual search techniques have been followed while exploring the search terms. Multiple digital libraries were explored during this process, and only those repositories have been selected for our search process that are commonly accessed by other systematic literature surveys globally. These widely used public venues happen to be relevant for

FIGURE 2. Search strategy.

TABLE 2. Research Questions (RQs).

(RQ)	Research Question Statement	Objectives and Motivations
RQ1:	Which are relevant publication channels for IPC research? Which channel types and geographical areas target IPC research?	To identify 1) high quality publication venues for IPC research 2) IPC research published during Jan-2011 till Jun-2020 3) scientometric analysis based on meta information of the selected articles including research type, approaches, and validation methods
RQ2:	What are the world wide accepted standards for IPC curriculum and what factors affect language choice for IPC?	To understand the requirements and concerns that need to be considered for programming education curricula; and frameworks and methods for the selection of first programming language
RQ3:	Which teaching and learning approaches have been reported to address the problems of novice students?	To identify various IPC approaches reported in the existing IPC literature covering following aspects: 1) pedagogical 2) cognitive 3) supporting tools
RQ4:	How effective are the tools devised for teaching and learning IPC?	To identify effectiveness of different IPC teaching/learning tools developed in literature in terms of: 1) evaluation method (statistical, questionnaire, interview etc) 2) Research type, quality assessment score 3) No. of participants in survey
RQ5:	Which methodologies have been adopted to analyze and validate performance of novice programmers? How assessment methods affect the learning of an IPC?	To discover different assessment tools and methods reported so far to evaluate students' performance during IPC in terms of: 1) programming errors investigation 2) programming and learning behaviour 3) course arrangement and programming aptitude 4) correlation among various assessment methods

our SLR too, therefore considered. In addition google scholar, that covers other such venues which are not directly explored, also included in our survey. Therefore, following eleven digital venues possibly covering almost all relevant research have been selected as primary search sources for automatic search:

- ACM Digital Library (http://dl.acm.org)
- IEEE eXplore (http://ieeexplore.ieee.org)
- PLOS ONE (https://journals.plos.org/plosone/)
- ScienceDirect (https://www.sciencedirect.com)
- SpringerLink (https://link.springer.com/)
- WileyOnlineLibrary (https://onlinelibrary.wiley.com/)
- arXiv (https://arxiv.org/search/cs)
- AIS eLibrary (https://aisel.aisnet.org/)
- IGI Global (https://www.igi-global.com/search/)
- Central and Eastern European Online Library (https://www.ceeol.com/)
- Google Scholar(https://scholar.google.com/)

The objective of manual search is to collect more literature relevant to introductory programming education curriculum and approaches domain. Extracted information can be more relevant for limited search terms therefore following conditions were applied to limit our search terms:

- Based on formulated RQs, determine primary keywords.
- Identification of secondary keywords and synonyms for additional keywords.
- 'AND' and 'OR' Boolean operators have been incorporated with keywords to develop a search string.

Possible arrangements of search string used can be noted from Figure 3, while a sample query has been presented

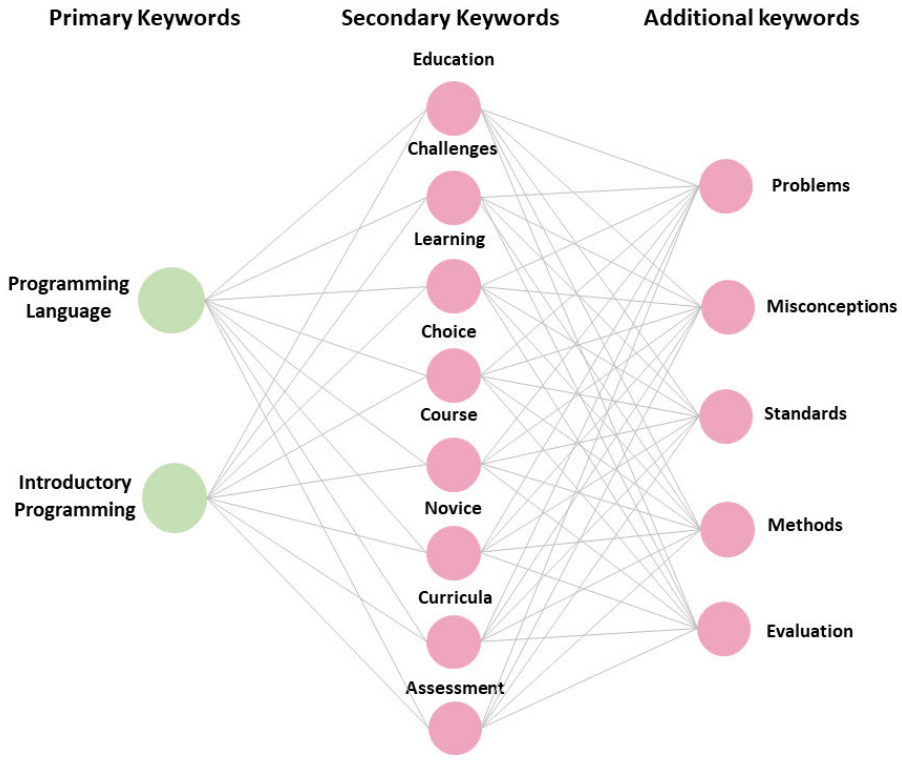

FIGURE 3. Search strings used to describe work included in our knowledge base.

```
(Program* Language OR Introductory Program*) AND
(Education OR Challenges OR Learning OR Choice OR
Course OR Novice OR Curricul* OR Assessment) AND
(Problems OR Misconceptions OR Standards OR Methods
OR Evaluation)
```

Listing 1. Search query.

in Listing 1. Primary keywords were selected as key identifiers for research of programming education. Primary keywords were chosen along with any of secondary or additional keywords. Combination of keywords, Boolean operators and wildcard have developed a final search query mentioned as:

Above search query appeared to be restrictive during initial search process. It was observed that above search string could not help inclusion of articles for only IPC curriculum.

Table 3 enlists final search strings used to explore eleven digital libraries with specific applied filters by applying limited keywords. Only titles were searched for ACM journals, ScienceDirect and PLOS ONE during automatic search. Other digital libraries were explored with ''all fields'' setting, as these do not allow a more specific search configuration. Search string being too restrictive failed to find relevant articles for IGI Global and CEEOL journals, therefore final search string designed for this library contains less number of keywords shown in Table 3.

2) SELECTION BASED ON INCLUSION/EXCLUSION CRITERIA

1) Inclusion criteria:
   a) Papers included in review must be in the domain of introductory Programming Education.
   b) Papers must target the research questions.
   c) Papers published in journals or conferences are included in review.
   d) Papers discussing IPC focusing on teaching or learning tools. Example: Intelligent tutoring system, board games etc.
2) Exclusion criteria:
   a) Remove papers written in non-english.
   b) Remove papers that do not discuss IPC in higher education.
   c) Remove the papers published before 2011.
   d) Remove papers discussing the IPC in school education.
   e) Papers written by same research group in this subject matter were removed (most recent was kept in this case).

3) SELECTION BASED ON QUALITY ASSESSMENT

Selection of relevant studies on the basis of quality assessment (QA) is considered as most important step for conducting any review. As the primary studies vary in design therefore quantitative, qualitative, and mixed-method critical appraisal tool used by [11] and [12] are followed to perform

TABLE 3. Search strategy for digital libraries.

Digital library	Search Query	Applied Filters
ACM digital library	[[All: program* language] OR [All: introductory program*]] AND [[All: education] OR [All: challenges] OR [All: learning] OR [All: choice] OR [All: course] OR [All: novice] OR [All: curricul*] OR [All: assessment]] AND [[All: problems] OR [All: misconceptions] OR [All: standards] OR [All: methods] OR [All: evaluation]]	[(01/01/2011 TO 06/30/2020)]
IEEEXplore	(Program* Language OR Introductory Program*) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curricul* OR Assessment) OR (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)	2011-2020
PLOS ONE	"(Program* Language OR Introductory Program*) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curricul* OR Assessment) OR (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)"	Jan 1, 2011 TO Jun 30, 2020
Google Scholar	(Program* Language OR Introductory Program*) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curricul* OR Assessment) OR (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)	2011-2020
ScienceDirect	(Program* Language OR Introductory Program*) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curricul* OR Assessment) OR (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)	2011-2020
SpringerLink	'(Program* Language OR Introductory Program*) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curricul* OR Assessment) OR (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)'	Computer Science, 2011-2020
Wiley Online Library	"(Program* Language OR Introductory Program*) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curricul* OR Assessment) OR (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)"	2011-2020
arXiv	(Programming Language OR Introductory Programming) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curriculum OR Assessment) AND (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)	Computer Science, 2011-2020
AISeL	(Program* Language OR Introductory Program*) AND (Education OR Challenges OR Learning OR Choice OR Course OR Novice OR Curricul* OR Assessment) OR (Problems OR Misconceptions OR Standards OR Methods OR Evaluation)	2011-2020
IGI Global	(introductory programming course)AND(learning OR curriculum OR language choice)	Individual Journal Articles (2011-2020)
CEEOL	(Introductory Programming OR Computer Programming OR Programming Education OR Factors)	Journal Articles, (2011-2020, English

TABLE 4. Possible ratings for recognized and stable publication source.

Publication Source	+4	+3	+2	+1	+0
Journals	Q1	Q2	Q3	Q4	No JCR Ranking
Conferences, Workshops, Symposia	CORE A*	CORE A	CORE B	CORE C	Not in CORE Ranking

QA in our review. In order to enhance our study, we have carried out QA by designing a questionnaire to assess the quality of selected papers. The QA of our study was carried out by three authors and each study is scored based on the following criteria:

a) The study has awarded score (1) if it contributes towards three identified aspects in IPC, otherwise scored (0).
b) If clear solutions for identified challenges in IPC have been provided by the study: "Yes (2)", "Limited (1)", and "No (0)" were the possible scores.
c) Score (1) is awarded to studies which presents empirical results otherwise scored (0).
d) The studies were rated by taking computer science conference rankings [13], and the journal and country ranking lists [14] into account. Possible scores for publications from recognized and stable sources are shown in Table 4.

A final score has been calculated for each study after adding scores of above questions: (a number between 0 to 8). Articles achieving scores 4 or more have been included in finalized results.

4) SELECTION BASED ON SNOWBALLING

After performing quality assessment, we conducted backward snowballing [15] through reference list of each finalized study to extract papers. Only those important candidate papers are selected which passed through inclusion/exclusion criteria. Once the paper is found, inclusion/exclusion of that

TABLE 5. Selection phases and results.

Phase	Selection	Selection criteria	ACM digital library	IEEExplore	PLOS ONE	ScienceDirect	SpringerLink	Wiley Online Library	arXiv	AISeL	IGI Global	CEEOL	Google Scholar	Total Papers
1	Search	Keywords (Figure 3)	180546	1017793	5836	178501	385085	2195499	48	10602	107	120	1350000	5328684
2	Filtering	Title	284	147	987	1532	654	549	10	199	19	12	9215	13735
3	Filtering	Abstract	71	57	98	541	63	82	3	40	8	6	322	1294
4	Filtering	Introduction and conclusion	48	34	14	21	12	17	3	1	4	5	56	189
5	Inspection	Full article	21	20	2	1	5	3	1	1	2	4	3	63

paper has been decided after reading its abstract and then other parts of paper. After having examined selected papers thoroughly we identified five more studies [16]–[19], [20] and totally added up to 60 primary studies.

C. REVIEW REPORT

Overview of selected studies is provided in this section.

1) OVERVIEW OF INTERMEDIATE SELECTION PROCESS OUTCOME

IPC is an extremely active field, therefore our review methodology had to empirically and systematically draw relevant studies from all related digital libraries. The next stage of our systematic review is to select the papers that will form the knowledge base for this review. Around 500k papers are left after removing papers older than year 2011. However, only one report published in year 2010 related to curriculum has also been included in this review to build strong basis for curricula comparison.

After building a knowledge base from eleven (11) digital libraries, authors examined title, abstract, and corresponding full paper if required of each search result. Papers less than four pages long and irrelevant papers were eliminated in this process.

To ascertain the relevance and contribution, in the field of IPC, accepted publications have been read thoroughly during inspection phase. To achieve the core goal of this study, we build a systematic knowledge base of articles based on their contributions.

2) OVERVIEW OF SELECTED STUDIES

Significant results of primary search, filtering and inspection phases, covering eleven digital libraries, are presented in Table 5. The automated search resulted in a very big number of papers while filtering/inspection phases helped reduce this number to 60 articles.

IV. ASSESSMENT AND DISCUSSION OF RESEARCH QUESTIONS

In this section, we analyzed the finalized 60 primary studies based on our research questions.

A. ASSESSMENT OF RQ1: WHICH ARE RELEVANT PUBLICATION CHANNELS FOR IPC RESEARCH? WHICH CHANNEL TYPES AND GEOGRAPHICAL AREAS TARGETING IPC RESEARCH?

The analysis of IPC learning tools, methods, contents, assessments, and language choice is a key challenge for researchers for the development of international standards for novice programmers. For this purpose, identification of high quality publication venues and scientometric analysis based on meta information in the domain of IPC is required. In this section, an insightful knowledge of publication sources, types, year and geographical distribution, publication channel wise distribution of selected studies for the analysis of IPC research is presented.

After inspection phase, maximum of 20 studies have been finalized from ACM digital library and nearly equal number is selected from IEEExplore as shown in Table 5, proving these publication sources as world's largest professional societies publishing more than 50 scholarly peer-reviewed journals in dozens of computing and information technology disciplines.

Figure 4 represents the number of publications w.r.t. year of publication. Maximum of 10 publications have been selected from year 2018 out of 60 showing more interest of researchers towards development in teaching and learning IPC in that year. However, less interest towards IPC research has been observed in most recent year resulting less improvement in

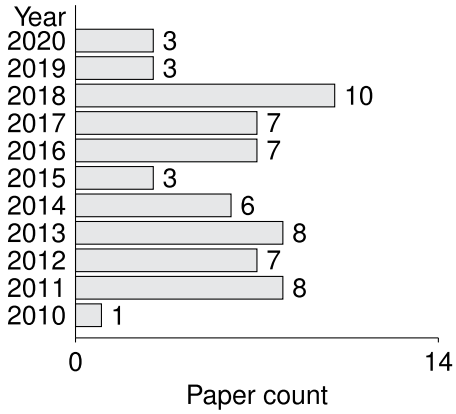

FIGURE 4. Introductory programming publications identified by our search.

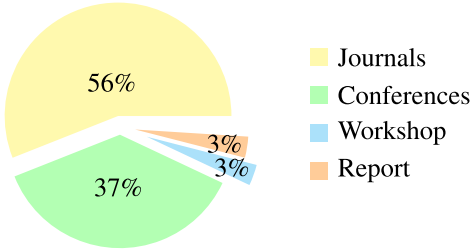

FIGURE 5. Percentage of publication type.

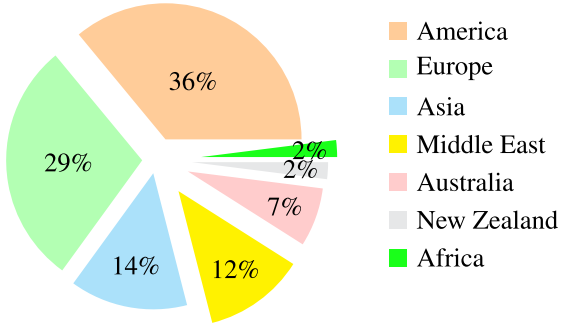

FIGURE 6. Percentage of distribution of publications.

IPC curriculum/assessment analysis in relevance to students and market needs.

It is observed from Figure 5 that highest number of studies have been selected from recognized journals and second highest number of studies have been picked from good rank conferences, whereas few relevant studies have been selected from workshops. Publication type for curriculum contents included in our SLR has been categorized as report, which are 3% of the total number.

Figure 6 shows geographical distribution of selected studies. It is observed that majority of publications i.e, 21 out of 60, have been published from different countries of America.

QA score for each finalized study awarded according to criteria defined in section III.B.3, shown in Table 7, it is clearly noted that QA scores are in the range of 4-8 discarding studies scored less than 4. IPC developers may find this QA helpful to choose related studies while addressing IPC challenges. Studies published in Q1 journals mostly scored highest while studies scoring total of 4 are from less recognized journal but highly relevant to the subject matter. Exclusion of studies from less stable publication channels might miss any relevant studies, which in turn can compromise the quality standards of this SLR, therefore have been included. Most of such studies scored four (4) making a total of 16 studies out of 60. Whereas, 15 out 60 studies scored highest (i.e, an eight 8) showing all QA criteria met by these studies to their maximum.

Overall classification results and QA of finalized studies have been presented in Table 6. Finalized studies have been classified based on five factors: research type, empirical type and methodology. We have categorized types for research as: SLR, solution proposal, evaluation research, experience paper, framework, review or curicula. Selection based on these defined research types helped us to build taxonomy presented in section V. All studies other then review papers have been empirically validated their results by performing surveys, statistical analysis, experiments or case studies increasing their quality standards and awarded one score each. Only 12 papers out of 60 have not presented empirical results indicated by zero score for category (c) of quality assessment criteria, including eight those studies which are SLRs (no proposed methodology hence no need to be empirically validated), other review papers. Only 17 papers score zero for category (d) of quality assessment criteria, rest of them score higher indicating competent sources. Four (4) was the lowest score any study has been awarded. In addition, only formally executed SLRs have been included in our selection, methodologies adopted by selected studies other than SLRs are also listed in Table 6.

Furthermore, we have identified sources of finalized studies, their publication channels and total number/percentage of studies per publication source mentioned in Table 8. About 14% of finalized studies have been published in (Q1) journal named: ''ACM Transaction on Computing Education'', whereas proceedings of ''IEEE Frontier in Education Conference'' (Rank A) appeared to be second highest publication source contributing 8% into our selected studies. In addition to that, Table 9 presents contribution and approaches developed by such studies whose research types are either 'solution proposal' or 'framework'.

B. ASSESSMENT OF RQ2: WHAT ARE WORLD WIDE ACCEPTED STANDARDS FOR IPC CURRICULUM AND WHAT FACTORS AFFECT LANGUAGE CHOICE FOR IPC?

Appropriate content and language choice are basic parameters while designing curriculum for IPC. These parameters play an important role in teaching and learning IPC. Following sub-sections describe existing studies in the domain of language choice and content selection for teaching and learning IPC.

TABLE 6. Classification.

References	Classification					Quality Assessment				
	P.Channel	P.Year	Research Type	Empirical Type	Methodology	(a)	(b)	(c)	(d)	Score
[1]	Journal	2010	Curricula	No	Content	1	0	0	3	4
[2]	Report	2013	Curricula	No	Content	1	0	0	4	5
[3]	Journal	2018	SLR	No	Formal	1	2	0	4	7
[4]	Journal	2014	Evaluation framework	Survey	Feature analysis	1	2	1	4	8
[5]	Journal	2012	SLR	No	Formal	1	2	0	2	5
[6]	Journal	2019	SLR	No	Review	1	2	1	4	8
[7]	Journal	2017	SLR	No	Formal	1	2	0	4	7
[8]	Journal	2013	SLR	No	Formal	1	2	0	4	7
[16]	Journal	2016	Evaluation research	Survey	Paradigm taught	1	2	1	4	8
[17]	Journal	2016	Solution proposal	Survey	Assessment design	1	2	1	4	8
[18]	Symposium	2018	Solution proposal	Experiment	Curriculum design	1	2	1	0	4
[19]	Journal	2018	Evaluation research	Survey	Formative assessment	1	1	1	4	7
[20]	Journal	2019	Evaluation research	Statistical analysis	Linear regression	1	2	1	4	8
[21]	Journal	2011	Solution proposal	Survey	Statistical analysis	1	2	1	4	8
[22]	Journal	2015	Solution proposal	Experiment	GUI based tool	1	2	1	4	8
[23]	Journal	2012	Experience paper	Experiment	Model	1	2	1	4	8
[24]	Journal	2013	Evaluation research	Survey	Statistical analysis	1	2	1	4	8
[25]	Journal	2014	Evaluation research	Framework	Formative assessment	1	2	1	4	8
[26]	Journal	2016	Solution proposal	Experiment	Assessment design	1	2	1	4	8
[27]	Journal	2016	Solution proposal	Experiment	Pedagogical comparison	1	2	1	4	8
[28]	Journal	2013	Evaluation research	Statistical analysis	Questionnaire	1	2	1	4	8
[29]	Journal	2017	Evaluation research	Statistical analysis	Questionnaire	1	2	1	4	8
[30]	Conference	2015	Evaluation research	Statistical analysis	Paradigm taught	1	2	1	4	8
[31]	Conference	2014	Evaluation research	Survey	Interviews	1	2	1	3	7
[32]	Conference	2014	Evaluation research	Survey	Observing code errors	1	2	1	3	7
[33]	Journal	2018	Evaluation research	Statistical analysis	Linear regression	1	1	1	4	7
[34]	Journal	2016	Evaluation research	Survey	Paradigm taught	1	2	1	3	7
[35]	Conference	2018	Solution proposal	Survey	Feature analysis	1	2	1	3	7
[36]	Conference	2011	Solution proposal	Experiment	Statistical analysis	1	1	1	4	7
[37]	Conference	2017	SLR	No	Formal	1	2	0	3	6
[38]	Conference	2018	SLR	No	Formal	1	2	0	3	6
[39]	Journal	2013	Review	No	Informal	1	1	0	4	6
[40]	Journal	2012	Evaluation research	Survey	Phone interviews	1	0	1	4	6
[41]	Journal	2015	Solution proposal	Model	Game	1	2	0	3	6
[42]	Journal	2018	Evaluation research	Survey	Statistical analysis	1	1	1	3	6
[43]	Conference	2014	Evaluation research	Survey	Observing code errors	1	2	1	2	6
[44]	Conference	2013	Solution proposal	Survey	Paradigm taught	1	2	1	2	6
[45]	Journal	2016	Evaluation research	Survey	Paradigm taught	1	2	1	2	6
[46]	Conference	2018	SLR	No	Formal	1	2	0	1	5
[47]	Journal	2017	Evaluation research	No	Review	1	1	0	3	5
[48]	Journal	2019	Evaluation research	Statistical analysis	Pedagogical comparison	1	2	1	1	5
[49]	Report	2017	Curricula	No	Content	1	0	0	4	5

TABLE 6. *(Continued.)* **Classification.**

[50]	Conference	2012	Evaluation research	Survey	Statistical analysis	1	0	1	3	5
[51]	Conference	2013	Evaluation research	Survey	Statistical analysis	1	2	1	1	5
[52]	Conference	2012	Evaluation research	Survey	Paradigms taught	1	2	1	1	5
[53]	Conference	2017	Evaluation research	Statistical analysis	Content	1	2	1	0	4
[54]	Conference	2018	Solution proposal	Experiment	Tool	1	2	1	0	4
[55]	Journal	2018	Solution proposal	Experiment	Assessment design	1	1	1	4	7
[56]	Journal	2011	Evaluation research	Survey	Paradigms taught	1	2	1	0	4
[57]	Journal	2013	Evaluation research	Case study	Tool	1	2	1	0	4
[58]	Journal	2012	Solution proposal	Experiment	Tool	1	2	1	0	4
[59]	Conference	2014	Evaluation research	Survey	Feature analysis	1	2	1	0	4
[60]	Conference	2011	Evaluation research	Survey	Observation	1	2	1	0	4
[61]	Conference	2011	Evaluation research	Survey	Code quiz	1	1	1	1	4
[62]	Conference	2012	Evaluation research	Survey	Assessing cognitive load	1	1	1	1	4
[63]	Workshop	2011	Solution proposal	Experiment	Web based application	1	2	1	0	4
[64]	Conference	2011	Evaluation research	Survey	Design classes	1	2	1	0	4
[65]	Conference	2016	Evaluation research	Survey	Questionnaire	1	2	1	0	4
[66]	Conference	2011	Evaluation research	Survey	Observation	1	2	1	0	4
[67]	Journal	2017	Evaluation research	Survey	Statistical analysis	1	2	1	0	4
[68]	Journal	2020	Solution proposal	Experiment	Tool	1	2	1	4	8

TABLE 7. **Quality assessment score.**

References	Score	Total
[4] [21] [6] [16] [17] [20] [22] [23] [24] [25] [26] [27] [28] [29] [30] [68]	8	16
[3] [7] [8] [19] [31] [32] [33] [34] [35] [36]	7	10
[37] [38] [39] [40] [41] [42] [43] [44] [45]	6	9
[2] [5] [46] [47] [48] [49] [50] [51] [52]	5	9
[1] [53] [18] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67]	4	17

1) LANGUAGE CHOICE

Many programming languages had been taught as IPC reported by one SLR [46] and several studies [24], [30], [35], [51], [52], [56], [59], [60], [67], [69] to assess best choice for an introductory programming language. Various languages used as IPC have been evaluated with the help of a comprehensive framework proposed by [4] and categorized using technical and environmental feature sets. Based on this framework researchers can develop new IPCs as it provides quantitative score computation for programming languages. Studies suggest selection of a language for IPC is based on the following factors:

- presence of important features (repetition structures and functional decomposition features)
- industry relevance and/or marketability to students
- online help and accessibility of a community
- availability of libraries and capability of being extended
- platform independent
- ease of installation
- compiled strongly typed languages (More errors have been captured at compile time in them therefore less vulnerable to runtime failures)
- pedagogical benefits

Another study by [36] has also assessed impact of different programming languages on software standards and developers productivity by performing statistical analysis on development of projects in C and C++ languages. They have concluded that use of C++ considered to produce better software quality and minimize maintenance efforts as compared to using C language. Furthermore, [50] reported that, C# is rarely adopted as IPC in CS1 & CS2 courses whereas often demanded in industry job ads. According to same study, Java and C++ are also demanded by industry frequently and often taught in CS1 & CS2. Findings by another study [25] revealed that teaching of a syntactically simple language (Python) as IPC accelerate students' understanding of programming concepts, in comparison to teaching a syntactically complex one (Java). In addition, significant improvement has been found in students' performance when problem solving has been taught before programming concepts. It is further noticed that choice of programming paradigm also helps in selecting programming language for IPC. According to our assessment of selected studies while investigating language choice and curriculum contents, that is elaborated well in Table 10, it is found that different programming languages appeared as choice for an IPC

TABLE 8. Publication source.

Publication source	Channel	References	No.	%age
PLOS ONE	Journal	[4], [6]	2	3
Electronic Journal of Information Systems in Developing Countries	Journal	[45]	1	2
Informatics in Education	Journal	[16], [17], [19], [20]	4	6
Journal of Computer Assisted Learning	Journal	[22], [55]	2	3
Journal of Computing Sciences in Colleges	Journal	[56]–[58]	3	5
Computers & Education, Elsevier	Journal	[23]	1	2
ACM Transaction on Computing Education	Journal	[7], [8], [21], [24]–[26], [39], [40]	8	14
IEEE Transaction on Education	Journal	[3], [27], [28], [33]	4	7
IET Software	Journal	[5]	1	2
IEEE Revista Iberoamericana de Tecnologias del Aprendizaje	Journal	[47]	1	2
Communications of the Association for Information Systems	Journal	[1]	1	2
International Journal of Game-Based Learning	Journal	[41]	1	2
Cluster Computing	Journal	[34]	1	2
Education and Information Technologies	Journal	[29]	1	2
Systemic Practice and Action Research	Journal	[42]	1	2
International Journal of Information and Communication Technology Education	Journal	[48]	1	2
ACM	Book	[2]	1	2
ACM/IEEE-CS	Book	[49]	1	2
IEEE Frontiers in Education Conference Proceedings	Conference	[31], [32], [35], [37], [50]	5	8
Conference on International computing education research	Conference	[43]	1	2
IEEE International Conference on Software Engineering	Conference	[30], [36]	2	3
ACM SIGITE conference on Information technology education	Conference	[51]	1	2
International Conference on Teaching and Learning in Computing and Engineering	Conference	[59]	1	2
International Convention MIPRO	Conference	[60]	1	2
IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)	Conference	[54]	1	2
IEEE International Conference on Advanced Learning Technologies	Conference	[44]	1	2
International Conference on Informatics in Schools: Situation, Evolution, and Perspectives	Conference	[64]	1	2
International Conference on Learning and Teaching in Computing and Engineering (LaTICE)	Conference	[65]	1	2
International Conference on Computer Systems and Technologies	Conference	[66]	1	2
Australasian Computing Education Conference	Conference	[46], [52], [61], [62]	4	7
ACM Conference on Innovation and Technology in Computer Science Education	Conference	[38]	1	2
Annual Conference of the Southern African Computer Lecturer's Association	Conference	[53]	1	2
Workshop on Open Source and Design of Communication	Workshop	[63]	1	2
ACM Technical Symposium on Computer Science Education	Symposium	[18]	1	2

based on multiple criterion. Both Java and Python have been well regarded first programming languages. Whereby, dominance of Java language can be clearly noticed where institutions select programming language based on industry relevance.

Whereas Python initially appeared as the first choice based on pedagogical benefits, but now it is a popular language in data science with strong supportive libraries. However, it is clear from the Table 10 that it is hard to clearly rate any programming language as the unanimous choice for first programming language. On the other hand, the most comprehensive and reasonably generic criteria for the selection of appropriate first programming language has been presented by [4].

TABLE 9. Solutions proposed by selected studies.

Study	Contribution	Approach
[4]	Evaluation framework to evaluate existing imperative or object oriented programming languages for their suitability as an appropriate IPC	Qualitative scores are assigned to each language after analyzing its features and parameters by following four qualitative values (fully supported, mostly supported, partially supported, not supported)
[17]	Identification of factors that effect student performance in an IPC by applying various assessment methods	Correlation among various assessment methods, students' participation and their final performance is measured after teaching course contents
[18]	To meet the needs of substantial fraction of community of non-majors, a course is developed focusing on use of programming in their own educational discipline	Course C0.5 was designed, launched and validated during this survey. Course contents delivered to students, which were taught in Python as blended learning approach. Survey is conducted in terms of students' feedback and pass rates.
[21]	Set out to rehabilitate a troubled first programming course by not changing inherent characteristics but introduced holistic changes in the course arrangements over a five year period	The course arrangements were observed to include both essential and accidental complexities in the goals, substance, implementation, and technical solutions.
[22]	A novel architecture for designing an intelligent tutoring system for teaching C++ programming language to novice programmers	A multi-agent system using an automatic text-to-flowchart conversion approach along with Bayesian technology. Additionally, making learning possible with minimum pre-requisites.
[23]	An online assessment system is developed based on peer-code review model called: EduPCR system	Students complete certain assignments created by teachers with medium difficulty after training students on peer code review. Scores of each assignment are then classified into process and quality scores. This approach was applied in a pedagogical study of two undergraduate courses in a university.
[26]	An instrument has been developed to evaluate students' knowledge of programming concepts. Using this assessment tool, impact of various languages and teaching approaches on students' knowledge has been evaluated.	Development of assessment tool based on: taught topics in IPC, errors made by novices and language-independent pseudo-code. 61 participants from two universities were recruited for the evaluation.
[28]	A literature to investigate students' mindset by conducting self-belief survey hat needs to be consider during design and evaluation of IPC	The questionnaire measured following aspects: mindset for programming aptitude and intelligence, programming practice behavior and performance. The sampling frame consisted of 296 undergraduate students of first and second year on programming modules.
[41]	This paper proposed a game-based learning (GBL) model as well as developed a game-play that solved puzzles. This game-play used concepts of computer science as elements of this tool.	GBL model and puzzle-solving game-play have been developed on the top of the work of an existing research. As players progress through this game, their problem visualisation and solution development skills continuously need to be increased.
[54]	Development of a framework for motivating and engaging students during an IPC	SplashKit API consists of educational resources, associated tools and dependant libraries that work together to support different teaching approaches and methods. This API is written on dependent libraries like: libCURL, SQLite, SDL with multiple language adapters including: python, C/C++, C# and Pascal.
[57]	A web-based platform for instructors to ask free-form short response questions to students named: StudentResponse System.	The system is implemented with a typical web stack of HTML, CSS, JavaScript, MySQL, PHP, and AJAX. The software runs on a portable XAMPP lite server on a portable jump drive or hard drive.
[58]	Visual Logic is an interactive, graphical software tool that enables a novice programmer to interactively develop executable flowcharts to help them visualizing abstract programming concepts.	Visual Logic includes a graphics package containing LOGO style commands which can be used to create colorful graphics programs.
[63]	A web-based application was proposed that allows the students to get the flowchart based on any code.	HTML5 and JavaScript are considered for the development of this application.
[68]	LearnBlock: First robot-agnostic academic tool has been developed that is loosely coupled software architecture. It allows the possibility of updating the operational blocks from code without extending the core code of proposed tool.	Model of weakly-coupled software modules has been followed for the design of LearnBlock, that allows visual programming independent from code generation and implementation of program. This tool is cross-platform and developed using Python programming language.

2) CONTENT

Surveys/reports on content design while developing curriculum of IPC have been reported by [1], [2], [18], [53] [49]. New IPC curriculum CS0.5 has been developed and taught in study [18] resulting improved pass rates of students in comparison to CS1. Curriculum developed in [53] and [49] clearly presented topic wise content comparison of all volumes.

According to (P27) of curriculum [2], students need to gain problem solving skills by applying knowledge they have learned more than just write code. They should to be able to develop and enhance a system based on its functionality, performance and usability while performing its quantitative and qualitative assessment. Much focus of introductory programming courses on ''Software Development Fundamentals'' topics along with a subset of ''Programming

TABLE 10. Curriculum: Findings of the reviewed studies.

Aspect	Criterion	Ref.	Evaluation Method	Findings
Language selection for an IPC	based on presence of important features (technical/ environmental)	[4]	Scoring function to compute quantified suitability score of each language	Python appeared on top based on technical features scoring, on the other hand Java scored highest based on environmental features. Whereas C language scored lowest based on technical features.
		[67]	Questionnaire	Survey indicates a dominance of Python, 800 members over 100 institutions of UK Universities, 80 IPC had been taught.
		[30]	This study is based on 7087 solution programs corresponding to 745 tasks in 8 widely used languages representing the major programming paradigms (procedural: C and Go; object-oriented: C# and Java; functional: F# and Haskell; scripting:	Statistical analysis shows that functional and scripting languages considered more brief than procedural and object-oriented languages. Scripting languages become competitive to C language in case of moderate input sizes otherwise C is hard to beat in terms of execution speed for large sized inputs. Moreover, strongly-typed languages are less vulnerable to runtime failures than weakly-typed languages as defects are possibly caught during compile time.
		[35]	Questionnaire based survey. 200 computer science programs across the USA were surveyed	Factors for choice of language are: ease of learning, tradition and jobs, as well as feature of language. Java was taught by most programs as IPC. This is followed by python and C++. It is further indicated by this survey that colleges with name "Engineering" mostly teach C-language while colleges with name "Science" teach python most.
	Pedagogical benefits dealing with syntax barrier	[24]	F-test, ANNOVA test, post-hoc Tukey tests and Mauchlys' test for sphericity	Identification of intuitive and non-intuitive words and symbols in a general purpose language. These show that perceived intuitiveness varies across word choices and language constructs, between programmers and non-programmers and across language as a whole. It is also found that novice accuracy varies across language constructs.
	Based on industry relevance	[4]	Scoring function to compute quantified suitability score of each language	Java scored highest whereas python appeared at second highest.
		[67]	Questionnaire	Survey indicates a dominance of Java, 800 members over 100 institutions of UK Universities, 80 IPC had been taught.
		[36]	To keep factors such as developer competence or software process uniform, open source applications written in a combination of C and C++ were statistically analysed and empirically validated.	A novel methodology was introduced for quantifying the impact of programming language on software quality and developer productivity. It is found that using C++ instead of C results in improved software quality and reduced maintenance effort, and that code bases are shifting from C to C++.
		[50]	521 job postings data were collected over an eight-month period for computer science associated job advertisements for major US cities. Correlation among job specification and language prevalence is detected using X^2-test.	Java, C++ and C# appeared to be frequently requested by industry. Developing a practice to consider industry based relevancy for language choice possibly benefits computer science education community, which in turn will help developing curriculum.
		[52]	Empirically validated. Experiment on 44 introductory programming courses in 28 Australian universities	Language comparison based on courses and students, showed java is first choice and python is second. Comparison of frequency for language selection based on use in industry/marketable shows 56.1% for the year 2001 and 48.8% for the year 2010, indicating impact of industry relevancy factor on the selection of programming language has been decreased between 2001 and 2010.
		[67]	Survey of IPCs (N = 80) taught at UK universities as part of their first year computer science (or related) degree programmes, conducted in the first half of 2016.	Dominance of Java has been observed in the results of first UK survey at a time when universities were still teaching novice programmers, although Python was perceived by that time.
	Based on pedagogical benefits	[19]	Questionnaire	C programming language was taught most, 59% students believe language choice facilitated their learning, 62% considered this language to be one of the most challenging.
		[67]	Questionnaire	Survey indicates a dominance of Python, 800 members over 100 institutions of UK Universities, 80 IPC had been taught.
Language selection for an IPC	Based on pedagogical benefits	[25]	Quantitative evaluation using formative assessment	Findings indicate that use of syntactically simple language (such as: Python) improves students' learning of programming in comparison to using a more complex one (like: Java) . Moreover, significant improvement in students' performance has been observed when problem solving was taught before programming.
		[26]	Reliability ratings were obtained using formulas: K-R21, Cronbachs' alpha	This study reveled that students performance has been significantly improving who were learning programming in Java and C++ as compared to those who were learning programming in Visual Basic.
		[27]	Research covered current state of 1019 programming subjects in 715 study programs at total of 218 faculties and 143 universities in 35 European countries	It was concluded that most IPC subjects are based on imperative paradigm while various programs highly support object-oriented paradigm of programming. It is further noticed that most often taught introductory programming language in the 1st semester in European tertiary is C, followed by C++, Java, Pascal and Python.
		[51]	Empirically validated using t-test and Mann-Whitney. Experiment conducted on two study groups: control group taught using C++ and experimental group using Java as programming language	It is concluded that no statistically significant impact of language choice found on the effectiveness of an online IPC when two groups were taught C++ and Java, and their performances were measured for achieving course intended learning outcomes (ILOs). Analysis was made based on four effectiveness indicators: students' understanding of multiple online course delivery metrics, levels of achieving the ILOs, and degree of students' satisfaction with the course.
		[52]	Empirically validated	Impact of pedagogical benefits for the selection of programming language change in percentage as 33.3% for the year 2001 to 53.5% for 2010. Languages that are seen as particularly beneficial for learning purposes (rather than for industry use) are becoming more popular, such as Python, Alice and Processing.
		[56]	Questionnaire based survey. Solutions of problems are compared for 30-60 students in a lab work and grades are then compared after implementing new introductory sequence	According to new sequence programming is first done in python switching to Java to achieve object-oriented feature. After introducing new introductory computer science courses, this study found that students are better served by focus on problems and their solutions rather than programming. Students reported liking the problem focus of courses, however the distribution of grades remained similar to those in previous years.
	Based on students' cognitive level/ ease of learning	[34]	Delphi method using an expert panel applied to 26 students for EPL course	It is suggested in this survey that educational programming language (EPL) for novices is not suitable for higher-grade students.
		[59]	Survey is conducted among 45 students based on open ended questions. C++ and Python courses were taught with a 90 minutes lecture twice a week for 14 weeks in separate semesters.	Results reveal that more student select Python as an introductory programming language compared to C++ due to the factors: simple and pseudocodish syntax, higher abstraction and easy to learn environment. Students exhibit more satisfaction while learning Python and consider conditional structures, loops and other features easier compared to C++.
		[60]	QBasic, C and Python programming languages were taught to the students of 7 colleges of Croatia in this survey to assess which one is best fit for an IPC.	Although Python seemed promising, the results of this survey indicate students' did not approach python seriously enough. Students' solution success records show that C-solutions were as much as three times lengthier than same solution coded in python.
Curriculum content design	Support for non-majors for learning programming	[18]	unpaired Welchs' t-tests	Helped students to develop better programming approaches preferably in their own academic discipline
	Standards for programming objectives	[49]	Report	Problem solving strategies and program development courses come under Software fundamentals domain. ITE-SWF-03, and ITE-SWF-04 are subdomain for ITE-SWF Domain (p 51, 59, 98)
	Topic level comparison	[53]	Topic wise contents are compared of all volumes using digraph structure visually, Graph Trans-morphism between different volumes 2001, 2008, 2013 of ACM/IEEE CS Curriculum	Algorithm and Visualisation of the Ratios for Difference Comparison and Knowledge Unit-Level Equivalence Comparison and Topic-Level Comparison Numbers,. ACM/IEEE curriculum volumes have changed rather significantly from CC2001 to CS2013 in content and structure. Programming fundamentals and languages topics also being compared in this study (I/C) comparisons are made.
	Program by design curriculum	[64]	Program by design curriculum has been implemented in many settings such as: secondary schools and universities.	This study proposed programming language concepts by designing abstractions based on existing programs for the improvement of advanced students' program design skills. This curriculum starts with a limited version of Java (FunJava) to reduce problems novices face.

Languages'' topics (P31). Some courses use functional programming while others adopt object-oriented programming or platform-based development. Findings and evaluation methods of other studies reviewing curriculum presented in Table 10 indicating there are only few studies present at the moment addressing standards for programming objectives for majors and non-majors, topic level comparison and program by design curriculum. According to the reviewed studies it is concluded that ACM/IEEE CS Curriculum considered as widely accepted standard.

C. ASSESSMENT OF RQ3: WHICH TEACHING AND LEARNING APPROACHES HAVE BEEN REPORTED TO ADDRESS PROBLEMS FOR NOVICE STUDENTS?

This systematic study aims to examine the existing knowledge in IPC from finalized 60 papers. In order to gain the overview, we investigate teachers' and students' perspective of IPC trends and approaches to address novice problems. Teaching/pedagogy an IPC refers to the techniques, tools, approaches and methods developed for teaching programming in an effective way. These developments in teaching methodologies contributed in finding solutions for a lot of challenges related to the nature of programming. Whereas, learning/cognition requirements is the students' ability to think and act in their academic life which help them acquire programming skills and overcome difficulties in learning by adopting any specific model, notional machine or games.

Teaching and learning an IPC by applying pedagogical skills or through tools are summarized and discussed in this section. Pedagogy enables instructors to impart knowledge and skills in such ways where students understand, remember and apply these skills. In our SLR, pedagogy is divided into two sub-levels i.e, teaching and learning. Existing surveys targeting teaching and learning IPC are evaluated in the following sub-sections:

1) TEACHING

Impact of various teaching approaches on students' performance for IPC has been surveyed by several studies [16], [20], [27], [31], [38], [42], [45], [47], [64], [65], [70]. Different pedagogical approaches are compared in terms of passing rate percentages by [27] where as some pedagogical and motivational strategies have been surveyed in [31], [38], [65] to promote students' motivation. Significant factors to predict students' attitude towards computer programming have been identified as three variables i.e, (performance in programming courses, perceived learning and programming self-efficacy) by [20]. ''Action research insights and outcomes ADRI approach'', ''active learning technique'', ''innovative programming environment approach'', ''models for teaching programming'' and ''program by design approach'' are various approaches discussed, compared and validated in studies [16], [42], [45], [47], [64]. Another study [71] proposed agile process for teaching an IPC and concluded positive impact on students' learning. These studies examined various factors that contribute to students' failure in IPC. Results from these studies highlight significance to measure the effect of curricular changes.

2) LEARNING

Existing studies investigated learning requirements and pointed out need of basic reasoning and mental skills for learning IPC. Surveys/literature targeting learning an IPC through methods/models, notional machine and games are evaluated as follows:

i) Methods/Models: Different types and models of learning styles have been reviewed in an SLR [37] concluding learning style can not be used as an instrument to predict student success. Students have been introduced to programming using different introductory programming languages in another study [29] and performance has been analyzed statistically. Another analysis of programming language education has been conducted by a study [34] for a pre-introductory CS course. According to the results of this survey, students recognized that education programming language based course framework helped their understanding of basic programming concepts and algorithm design. Objective of both of these studies is to possibly improve the selection of suitable programming language for novice programmers. Another recent study [70], explores the effect of incremental mindsets in individuals and found an increases in effort during programming activities but not in performance.

ii) Notional Machine:
An informal literature review to examine knowledge bodies in conceptual relation to a ''notional machine'' has been presented in study [39]. Notional machine serves as an abstract computer for the execution of particular kind of programs and fulfills the purpose of understanding program execution stages. One or more programming languages/paradigms with one specific programming environment are possibly associated with notional machine. several aspects on the functionality of notional machines in introductory programming education (IPE) have been discussed together in this study that concludes notional machines as a prominent challenge in IPE.

iii) Games/Robots:
A game-based learning (GBL) model is presented by [41] and [68] developed a robot-based programming tool. Major contributions and approaches used for the developments in domain of IPC are presented in Table 9.

Table 11 elaborates the criterion, evaluation methods, participant details and findings of selected studies addressing IPC teaching and learning methods and techniques. Four evaluation criterion were identified for teaching aspect during review that includes seven studies. These studies looked at factors effecting students' learning, pedagogical and motivational strategies, use of active learning techniques and

TABLE 11. Teaching and Learning: Findings of the reviewed studies.

Aspect	Criterion	Ref.	Evaluation Method	Findings
Pedagogy/ Teaching	To reveal the factors effecting students' learning	[20]	Linear regression	**1)** students had moderately high attitude towards computer programming (ATCP), **2)** their ATCP had significant correlations with their achievement in computer programming courses, computer programming self-efficacy, and perceived learning, **3)** three variables (achievement in computer programming courses, computer programming self-efficacy, and perceived learning) were significant predictors of their ATCP
		[29]	Questionnaire based survey consisting of 27 questions. Statistical analysis performed based on MWU-test and KS-test.	A total of 288 students from three universities responded to this survey. It is concluded that instructors have a great impact on students' learning whereas selection of programming language has less or no impact. As main learning goals of an IPC is to support students in learning problem solving skills as well as designing simple algorithms.
	Pedagogical and motivational strategies	[31]	Interview based survey. 18 teachers belong to the University of Coimbra and Polytechnic Institute of Coimbra were interviewed	Aspects such as the student-teacher relationship, the motivational strategies used, the assessment method or the materials presented in class were pointed out. Student-teacher relationship, class competitions, challenges, continuous assessment, strategies in which the student is forced to engage actively are considered to have better results.
		[42]	Survey comparing traditional teaching approach and action research insights and outcome (ADRI) teaching approach. Nine different activities were conducted during three cycles of this action research. Different tools such as surveys and the focus group were used to collect the feedback.	Four stages of ADRI teaching approach were integrated into teaching material. The impact of ADRI approach was examined from final grades which showed improved outcomes as well as positive impact in students' retention during use of ADRI approach.
		[45]	28 face-to-face interviews were held with information systems degree program students. A conceptual model is proposed for understanding the influence of social context on students learning of programming.	The findings identified lack of intrinsic motivation and future expectation, anxiety, peer influences, and poor lecturer skills and behavior as the challenges resulting in a high failure rate in IPCs. Lecturers should involve students in activities such as:pair programming, video tutorials, 3D environment to teach programming, teaching through fun, non-competitive and interactive activities and with plenty of support towards fixing of errors in programming.
	Use of active learning techniques	[47]	Comparative analysis for the effectiveness of various active learning techniques	Possible methods for incorporating active learning techniques into teaching an IPC are: **1)** Write, compile and run first computer program **2)** Tutorial: Picture Viewer or Maze Program **3)** Learn how to cook spaghetti. **4)** Flowchart Creation: Telephone Call **5)** Visual Studio programming environment
	Application based teaching technique	[65]	The sample consisted of 22 postgraduate computer science students. Their responses to the instruments administered, their scores and their LMS access logs were statistically analyzed.	It is found that students appeared to look at the programming concepts from a broader view and may be able to correlate these concepts with one another when a programming course is taught using one or two applications and developed the application by distributing the development of the features throughout the semester.
Cognition	Considering students' cognitive level	[34]	Delphi method using an expert panel applied to 26 students for EPL (educational programming language) course	An EPL selected based on important factors has been taught to the participants using RUR-PLE (virtual game like environment). This survey indicates that, considering the characteristics of the learner, a GUI-based quick-and-easy programming environment is needed and selection of EPL should be based on students' cognitive level.
		[39]	Bodies of work evaluated in relation to the notional machine	The idea that novice programmers must learn programming using one or more notional machines is supported by several perspectives such as: theories of mental models and constructivism, misconceptions catalogs, theory of threshold concepts, and learners' ways of experiencing programming.
		[62]	Qualitative responses were gathered from academics as part of a study of 44 introductory programming courses in 28 Australian universities.	It is suggested that for many low-performance students learning fails due to cognitive overload.
	Types and models of learning styles	[37]	Most widely used learning types out of total 71 are addressed in this survey like: Active, reflective, sensing, intuitive, visual verbal, sequential, global etc. Four learning models: Soloman-Felder Index, VARK Modalities, Perkin's model, and Gregorc Style Delineator (GSD) were explored to find impact on students' performance.	The Soloman-Felder appeared to be the most used model according to this study. Moreover, active method of teaching considered effective for the encouragement of a variety of learners. The results point out that learning style cannot be considered the only factor to assess a students' ability to learn.
Tool	Learning by doing and with minimum pre-requisites	[22]	Quasi-experimental design along with Bayesian technology was adopted to investigate the efficacy of this system	This system has proved to be effective **1)** in students' problem-solving improvement during a 4-week experimental study, **2)** for students with a lower prior knowledge level as compared with others.
	Learning through games/ API/ GUI/ Robots	[41]	Implementation of proposed approach was processing at e-learning centre of the University of Greenwich at the submission time of this study	Learning of programming specifically in relation to procedural knowledge and associated skills development were focused by the proposed game-based learning (GBL) model. GBL model also provides theoretical knowledge to novice on computer science concepts. In order to transfer skills into learning programming, some strategies need to be applied including: modelling, scaffolding and coaching.
		[54]	IPC taught using SplashKit framework for several years at both postgraduate and undergraduate levels by adopting objects-first and objects-later curriculum design	While teaching with SplashKit, students have been able to use these tools to create a wide range of fun and engaging games, as well as small business-like applications all with an easy to approach API design
		[58]	Three different courses, developed independently at three different colleges, using Visual Logic with completely different approaches have been evaluated in this paper.	Using this visual tool generates far more student enthusiasm and understanding than the traditional approach of explaining syntax and fixing errors.
		[68]	Some programming examples created with LearnBlock using various robots. Specifically, three different programs tested in different pairs of robots: Square trajectory, Reaction to tags and Line follower.	This desktop application proposes a progressive learning process that leads to the use of a general-purpose programming language. This is achieved by extending the robot-agnostic property to the generated code.

application based teaching techniques. Approaches developed considering pedagogical and motivational strategies appeared to be more effective towards students' learning and programming behaviour. On the other hand, active learning and application-based teaching techniques showed promising results when used during an IPC. It is important to note that four studies contributed in students' cognitive level meeting two evaluation criterion. Findings of these studies suggesting adoption of such learning models that consider the characteristics of the learners.

D. ASSESSMENT OF RQ4: HOW EFFECTIVE IS TEACHING IPC THROUGH TOOLS IN IMPROVING THE LEARNING EXPERIENCE OF NOVICE PROGRAMMERS?

Teaching and learning an IPC through tools is divided into following categories:

1) IMPLEMENTED TOOLS

GUI based tools have been developed for teaching and learning an IPC and empirically validated in studies [22], [58], [63] with the aim of improving problem solving ability

of novice programmers. These tools allow the programmers to interactively develop executable flow charts. Introduction of abstract programming concepts visually allow students to make transition itno a complex programming language easier, concluding visual tools are relevant in attaining better results.

2) FRAMEWORK

A development framework has been designed by [54], called Splashkit, for motivating and engaging students in introductory programming for tertiary education. This is an open-source, cross-platform development and language-agnostic framework that supports a wide range of programming education approaches. Objects-first/ objects-later curriculum design, a range of fully-featured game engines like programming tools, databases, programming language choices, and web-servers together with easy API design techniques are the basic programming education approaches included in this framework. Objective of this study to empower students for developments of important software engineering skill that is reading API documentation. Students were able to design games when taught using this framework.

As can be seen in Table 11 four studies have been categorized addressing learning of an IPC using tools based on two evaluation criterion. Proposed approaches proved to be effective compared to traditional learning styles as these helped students for problem solving improvement and to develop a wide range of applications. It is concluded that teaching with SplashKit considered as more beneficial as this framework is empirically evaluated on large scale of participants for several years.

E. ASSESSMENT OF RQ5: WHICH METHODOLOGIES HAVE BEEN ADOPTED TO ANALYZE AND VALIDATE PERFORMANCE OF NOVICE PROGRAMMERS?

Surveys performed to assess teaching and learning process of IPC are categorized into following sub-levels.

1) STUDENTS' PERFORMANCE ANALYSIS

Several methodologies have been adopted to analyze and validate students' performance in IPC by various studies [19], [21], [28], [32], [33], [40], [43], [44], [48], [61], [62], [66]. Objective of these studies is to predict factors effecting students' performance in an IPC by observing programming errors made by students using multiple testing techniques: ADRI model based on six categories of Blooms' Taxonomy, regression and t-tests, formative and summative approach, and bayesian network classifier. Few studies suggested that for many low-performance students learning fails due to cognitive overload, while others identified students' main problem was to divide activity into functions and classes and to find errors in ones' own programs. One survey observed that implicit theories of programming-aptitude and programming-efficacy are interrelated and positively correlated with effort, performance, and previous failures in the course. Repeaters favor fixed programming aptitude and have lower programming efficacy which increases further possibility of failure.

2) ASSESSMENT TOOL

A web-based students' response platform has been developed in study [57] for instructors to ask free-form short response questions to students, while impact of different teaching approaches and languages on students' learning an IPC has been assessed by study [26] using an assessment tool. objective of both of the studies is to assess teaching and learning an IPC. Programming language learning is assessed in another study [23] based on peer code review model EduPCR2. The EduPCR system and its PCR-based assessment process have significantly improved student learning outcomes in many areas including programming skills.

Custom-based assessment tools, self-assessment tools and tools on giving student feedback have been reviewed in another recent SLR [38]. This includes ProgTest, UML testing, industrial based testing suite, Bluefix, NoobLab and PRAISE. Another recent study [72] build a comprehensive automated programming assessment system named: Edgar, that deals with various programming languages and can be deployed on all major operating systems.

3) ASSESSMENT DESIGN

Various factors have been reviewed in studies [17] [55] to improve student academic self-efficacy and problem solving and in learning programming by applying Spearmans' rank correlation coefficient technique. It was identified in [55] that performance in formative assessment and problem-solving-skill are weakly correlated. These studies aim to discover correlation between various assessment method, student's participation and their final performance.

Table 12 further elaborates findings and evaluation methods of each reviewed study which are evaluated on criterion like: students' learning and programming behaviour, auto recognition of required features in students' code, students' course performance, students' programming errors investigation and correlation among perceived and academic performance.

First most addressed evaluation criteria was students' course performance, which was used to evaluate nine studies. Most of these studies gather data through assessment which is then statistically analyzed. These studies attempt to predict students' pass rates as well as evaluated different factors effecting students' performance. It is concluded that such assessment tasks/tools considered more effective which tend to assess students' problem solving skills and logical errors in their code. Moreover, considering factors such as: structural design of assessment, time management, auto feature detection and balanced weights of labs, quizzes and, midterm/final exams may significantly improve methodologies for assessing students' performance in an IPC.

TABLE 12. Assessment: Findings of the reviewed studies.

Aspect	Criterion	Ref.	Evaluation Method	Findings
Assessment Tool	Students' learning and programming behaviour	[23]	Several programs were inspected and evaluated for around 100 participants for Java and C programming courses.	Developed assessment system has significantly improved **1)** student learning outcomes in programming skills, **2)** compliance with coding standards, **3)** collaborative learning and **4)** time management. **5)** Students' satisfaction rate with developed assessment system exceeded 80%.
		[26]	Reliability has been evaluated on the basis of internal consistency measures with two different formulas: K-R21, Cronbachs' alpha	This study found that factors such as instructional approach and language do not affect students' performance addressing object-oriented concepts. Whereas format of assessment is considered as such a factor that possibly can affect students' performance (Like MCQ's type questions etc). Proposed assessment tool supports imperative and object-oriented programming.
	Automatic recognition of features in students' code	[57]	The system was deployed in an IPC with approximately two dozen students in each of two sections.	The analysis presented in this paper focuses on five questions about forming method headers and the 231 responses provided by students to those questions. The algorithm described in this paper can only learn limited models with a disjunction or wildcard operator.
Assessment design	Correlation among various assessment methods	[17]	Marks variation and average deviation	It is suggested to **1)** reduce the weights of labs, assessments and quizzes, **2)** and increasing the wights of final and midterm exams.
	Perceived vs academic performance	[55]	Correlation among problem solving skill (PSS) and student performance has been measured in an experiment on 200 students from one semester in 2016 and Spearman's rank correlation coefficient technique is used for empirical validation	This study found weak correlation among students' performance in formative and summative assessment tasks and PSS. However, assessing student PSS levels may help instructors to develop instructional interventions and assessment tasks and to improve student academic self-efficacy and problem solving and in learning programming.
Students' performance analysis	Students' need for assistance	[19]	Formative and summative assessment approaches were used to evaluate results	**1)** 71% students need assistants for help with the subject **2)** 65% students feel comfortable working under a research project process
	Programming aptitude and course arrangement	[21]	Theory of constraints	**1)** The pass rate had been 44% in 2004 but, after five years of study and course improvements, the pass rate reached 68% in 2009, a 55% improvement. **2)** The low pass rate for the studied course resulted mainly from three factors: programming as a discipline, course arrangements, and student behavior.
		[28]	Mindset for intelligence and for programming aptitude was measured by drawing items from Dwecks' mindset scale	This study finds co-relation between students' mindset for programming aptitude with their mindset for general intelligence. It concludes some evidence that mindset for programming aptitude is not only distinct from mindset about intelligence, but that it may also have a stronger relationship with programming practice.
		[33]	Impact and correlation of programming aptitude and self efficacy on course performance is evaluated through regression and t-test. Participants/Institutes (193 students / 3 sections / same course and university) That include regular and repeating students.	The results show that students develop domain-specific implicit-theories or self beliefs. So the students who believe in improving have higher programming efficacy therefore they study more and get better grades. While the ones who re-enroll tend to persist and think that programming is difficult. While some of these students (who think that programming is difficult) can be detected using some measures in this study. Thus they can be intervened and counseled to fight their implicit theoretical mindset and self-efficacy issues.
		[40]	Survey is empirically validated based on 7 phone interviews conducted with 24 registered students of Columbus State University to evaluate students' performance in two online computer science courses: Fundamentals of computer science and IPC.	This study suggests to establish the task structure of novice problem solving in programming with respect to time. Material re-use by instructors may largely influence structural aspects of instructional design. Decrease in individual tutoring workload may be considered as a compensation to provide time for individual problems.
		[44]	66000 snapshots of six-week Java programming work from 152 participants were gathered to predict students' performance by utilizing a non-parametric bayesian network classifier B-Course.	This study predicts whether the student is a high-performer, passes the course, or fails the course with a 78% accuracy.
		[48]	ADRI model is used for assurance of learning programming tasks tested based on six categories of Bloom's Taxonomy, empirically validated by collecting students' grades from registration dept.	Results show that 93% students passed the course where 7% are high achievers, 63% are medium achievers whereas 23% appeared as low achievers. The results show that students did not perform well in the basic structure of C++, designing a program, and predefined functions topics of the IP course.
Students' performance analysis		[66]	The participants were 158 students that had attended a first year course in programming of Reykjavik University in Iceland	This study shows that it is important to bear in mind that students' problems can have different roots and therefore it is not wise to concentrate on one reason of difficulties. Students' main problem was to divide activity into functions and classes and to find errors in one's own programs.
	Students' programming errors investigation	[43]	Frequency rankings gathered from 76 educators for 18 students' mistakes over 100,000 students. A conversion to Spearmans' correlation for ranked data is used for interpretation.	This study found that educators formed only a weak consensus about which mistakes are most frequent in Java among programming novices, that their rankings bore only a moderate correspondence to the students in the Blackbox data. It is possible that each educator is correct for their own students making these frequencies contextual. No correlation found among experience level of educators to how close their frequency rankings with those from Blackbox data.
		[61]	Students attending the lectures in weeks 3 and 5 were given two short written tests, the results then examined based on X^2 Test.	It is concluded that current pedagogical practice does not help novice programmers think relationally. Early detection and treatment of students who do not think relationally may improve failure rates.
		[32]	Error categories have been developed and checked for inter-coder reliability. Error frequencies are recorded in a table after collecting student error data	Two university students participated in this study for introductory java programming courses. It is found that ,assessment methodology can be improved by analysing logical errors instead diagnostic errors. It further results in human reliability for diagnosing errors more than compilers do.

V. DISCUSSION AND FUTURE DIRECTIONS

This section summarizes and discusses the results related to the systematic literature review.

A. TAXONOMIC HIERARCHY

The goal of this systematic literature review was to examine the current knowledge in IPC by selecting 60 papers. To achieve this goal, We build taxonomic hierarchy of selected studies shown in Figure 7, excluding those studies (SLRs) which have not validated their methodologies empirically. We have investigated developments and challenges for the aspects such as: IPC curriculum, teaching and learning and assessment. However, these aspects are then further divided into many sub-levels showing depth of each aspect and their role in the performance improvement of novice programmers.

B. GENERAL OBSERVATIONS AND FUTURE DIRECTION

Several observations possibly can be made in the findings of this SLR. Various RQs were developed in order to determine methods/approaches/tools while teaching an IPC and to provide an exclusive overview of subject matter. Many trends and findings can be noted as a result in regards to the challenges for teaching and learning IPC. These include following observations together with future directions:

1) CURRICULA

The motivational factor for big proportion of selected studies (33%) involves designing curriculum contents and selection of appropriate programming language for IPC. Early programming curricula attempts focused mainly on the relatively simple challenge. Various factors for the

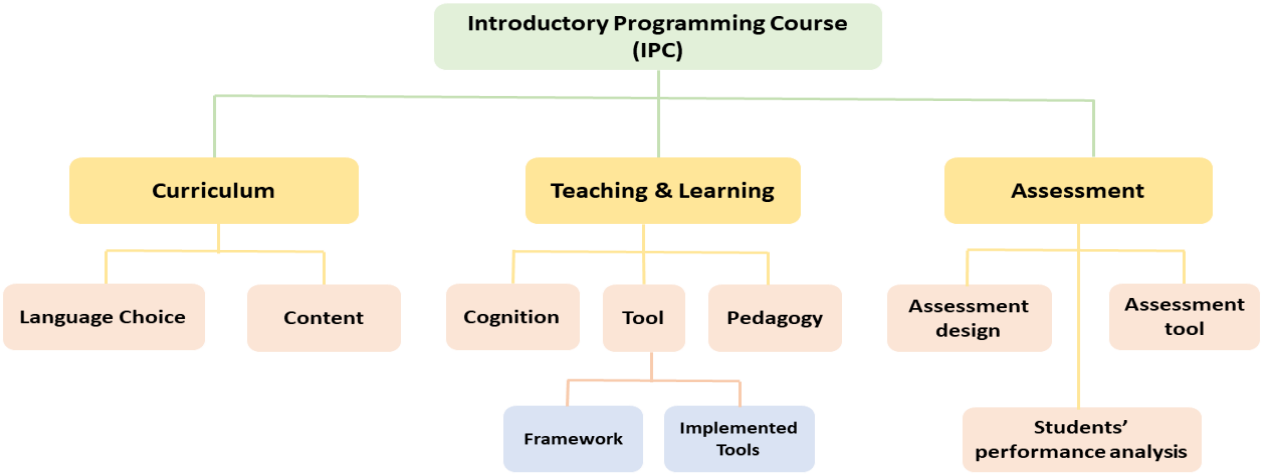

FIGURE 7. Taxonomy of introductory programming education perspectives.

selection of language for IPC have been discussed by many researchers, concluding that using a syntactically simple language facilitated student's learning of programming concepts. Therefore, Python as first programming language considered as first choice of instructors, whereas, Java leads where industry relevancy becomes core requirement. Curriculum developers, on the other hand, have been focused on the development of course contents for computer science as major. Little has been done to tackle challenges in IPC course contents for non-majors. Moreover, Lack of curricula standardization appears as a limitation as the industry cannot rely on teachers' subjective attitude towards programming. Students might be able to learn IPC well if they have been taught IPC starting from school years instead of graduation level and by including algorithm visualization tools in curricula. In addition, coding standards and year wise focus of each study must be considered while developing IPC curricula in future.

2) TEACHING AND LEARNING

A large number of publications that we have explored focus on methods/tools designed for teaching and learning IPC. Where most of the work has clearly help students learn programming through building tools. Comparatively less attention has been paid to analyzing and evaluating those learning tools, according to our observations. No/less distribution of these learning tools beyond the institution where they were designed also been observed. Students can improve their problem solving skills and learn C++ programming language using tool developed in [22] whereas multiple programming languages are supported by another recent learning tool: SplashKit [54]. It is further noticed that web-based programming learning tools were developed in early years while trend recently shifts towards serious games using mobile phones.. In future, more modern tools should be developed for learning IPC such as serious games.

3) ASSESSMENT

The vast majority of literature assumed that formative/summative assessment approaches are enough to measure student's performance. Little has been done to focus on assessment design. Code conventions are important to programmers to improve the readability and maintainability of code which needs to be assessed while measuring performance of novice programmers. We have noticed a language dependant web-based assessment tool [23] that assesses students' performance for object oriented programming in Java course and C programming for freshmen. Whereas, a language-independent assessment tool [26] has been found during this review which assesses students' grasp of all fundamental and object-oriented concepts. Another domain independent practical system designed in [57] for collecting and labeling student responses to open ended questions has also been identified by our study. Associated rubric must be included while designing assessments containing functional correctness scheme, technical correctness scheme, high quality code sample files and code convention restrictions. Tester files must also be included while assessing novice programmers. To enhance learning an IPC, web based assessments and strong assessments need to be developed.

VI. CONCLUSION

This study has been conducted to build an understanding of research trends in the field of Introductory Programming Education. To ensure thorough coverage of challenges and their solutions we followed a systematic literature review. We have performed search using as many terms as we know associated with IPC then results have been evaluated accordingly. We concluded our search in June 2020, so studies conducted after that date would have not been included. Eleven main digital repositories were explored for

this search and around 60 articles appeared to contribute even better students' performance out of 500k searched articles.

The results show that more than half of the selected studies appeared in recognized journals and only a few had published in workshops or as reports. ''Solution proposal'' and ''evaluation research'' have been reported as two main research types adopted by these studies. Majority of selected studies were evidence based and possibly helping the instructors to gain maximum benefits of IPC teaching and learning. Pedagogy, language choice and students' performance analysis were found as most frequently addressed aspects of IPC, whereas curricula contents, assessment tool/design and teaching/learning through tools have been appeared as less addressed aspects of IPC.

Limitations of any SLR are mainly related to search strategy, inaccurate extracted data or misclassification. However, our search strategy reduced the risk of selection bias by conducting this study with different keywords from all common digital repositories. By applying rigorous inclusion/exclusion criteria and asking two independent reviewers for assessment of all extractions, other two risks were addressed.

For future research on IPC, more attention should be paid to curricula design for non-majors, tools design; particularly serious games, and web-based assessment design with rubric. More evaluation research should be carried out in order to evaluate existing IPC curriculum contents.

REFERENCES

[1] H. Topi, J. S. Valacich, R. T. Wright, K. M. Kaiser, J. Nunamaker Jr, J. C. Sipior, and G. De Vreede, ''Curriculum guidelines for undergraduate degree programs in information systems,'' in *Proc. ACM/AIS Task Force*, 2010, pp. 56–57.

[2] M. Sahami *et al.*, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2013.

[3] R. P. Medeiros, G. L. Ramalho, and T. P. Falc ao, ''A systematic literature review on teaching and learning introductory programming in higher education,'' *IEEE Trans. Educ.*, vol. 62, no. 2, pp. 77–90, May 2019.

[4] 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.

[5] L. Major, T. Kyriacou, and O. P. Brereton, ''Systematic literature review: Teaching novices programming using robots,'' *IET Softw.*, vol. 6, no. 6, pp. 502–513, 2012.

[6] A. Alammary, ''Blended learning models for introductory programming courses: A systematic review,'' *PLoS ONE*, vol. 14, no. 9, Sep. 2019, Art. no. e0221765.

[7] Y. Qian and J. Lehman, ''Students' misconceptions and other difficulties in introductory programming: A literature review,'' *ACM Trans. Comput. Edu.*, vol. 18, no. 1, pp. 1–24, Dec. 2017.

[8] J. Sorva, V. Karavirta, and L. Malmi, ''A review of generic program visualization systems for introductory programming education,'' *ACM Trans. Comput. Edu.*, vol. 13, no. 4, p. 15, 2013.

[9] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, ''Lessons from applying the systematic literature review process within the software engineering domain,'' *J. Syst. Softw.*, vol. 80, no. 4, pp. 571–583, Apr. 2007.

[10] B. Kitchenham, *Procedures for Performing Systematic Reviews*, vol. 33. Keele, U.K.: Keele Univ., 2004, pp. 1–26.

[11] A. Fernandez, E. Insfran, and S. Abrahão, ''Usability evaluation methods for the Web: A systematic mapping study,'' *Inf. Softw. Technol.*, vol. 53, no. 8, pp. 789–817, Aug. 2011.

[12] S. Ouhbi, A. Idri, J. L. Fernández-Alemán, and A. Toval, ''Requirements engineering education: A systematic mapping study,'' *Requirements Eng.*, vol. 20, no. 2, pp. 119–138, Jun. 2015.

[13] (2018). *CORE Conference Portal*. Accessed: Jan. 6, 2020. [Online]. Available: http://portal.core.edu.au/conf-ranks/

[14] (2018). *Scimago Journal & Country Rank*. Accessed: Jan. 6, 2020. [Online]. Available: https://www.scimagojr.com/

[15] C. Wohlin, ''Guidelines for snowballing in systematic literature studies and a replication in software engineering,'' in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng. EASE*, 2014, p. 38.

[16] V. Aleksić and M. Ivanović, ''Introductory programming subject in European higher education,'' *Informat. Edu.*, vol. 15, no. 2, pp. 163–182, 2016.

[17] R. A. Alturki, ''Measuring and improving student performance in an introductory programming course,'' *Informat. Edu.*, vol. 15, no. 2, pp. 183–204, Nov. 2016.

[18] J. Q. Dawson, M. Allen, A. Campbell, and A. Valair, ''Designing an introductory programming course to improve non-Majors' experiences,'' in *Proc. 49th ACM Tech. Symp. Comput. Sci. Edu.*, Feb. 2018, pp. 26–31.

[19] E. D. Canedo, G. A. Santos, and L. L. Leite, ''An assessment of the teaching-learning methodologies used in the introductory programming courses at a brazilian university,'' *Informat. Edu.*, vol. 17, no. 1, pp. 45–59, Apr. 2018.

[20] M. D. Gurer, I. Cetin, and E. Top, ''Factors affecting students' attitudes toward computer programming,'' *Informat. Edu.*, vol. 18, no. 2, p. 281, 2019.

[21] U. Nikula, O. Gotel, and J. Kasurinen, ''A motivation guided holistic rehabilitation of the first programming course,'' *ACM Trans. Comput. Edu.*, vol. 11, no. 4, p. 24, 2011.

[22] D. Hooshyar, R. B. Ahmad, M. Yousefi, F. D. Yusop, and S.-J. Horng, ''A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers,'' *J. Comput. Assist. Learn.*, vol. 31, no. 4, pp. 345–361, Aug. 2015.

[23] Y. Wang, H. Li, Y. Feng, Y. Jiang, and Y. Liu, ''Assessment of programming language learning based on peer code review model: Implementation and experience report,'' *Comput. Edu.*, vol. 59, no. 2, pp. 412–422, Sep. 2012.

[24] A. Stefik and S. Siebert, ''An empirical investigation into programming language syntax,'' *ACM Trans. Comput. Edu.*, vol. 13, no. 4, p. 19, 2013.

[25] T. Koulouri, S. Lauria, and R. D. Macredie, ''Teaching introductory programming: A quantitative evaluation of different approaches,'' *ACM Trans. Comput. Edu.*, vol. 14, no. 4, p. 26, 2015.

[26] W. M. Kunkle and R. B. Allen, ''The impact of different teaching approaches and languages on student learning of introductory programming concepts,'' *ACM Trans. Comput. Edu.*, vol. 16, no. 1, pp. 1–26, Feb. 2016.

[27] G. Silva-Maceda, P. D. Arjona-Villicana, and F. E. Castillo-Barrera, ''More time or better tools? A large-scale retrospective comparison of pedagogical approaches to teach programming,'' *IEEE Trans. Educ.*, vol. 59, no. 4, pp. 274–281, Nov. 2016.

[28] M. J. Scott and G. Ghinea, ''On the domain-specificity of mindsets: The relationship between aptitude beliefs and programming practice,'' *IEEE Trans. Educ.*, vol. 57, no. 3, pp. 169–174, Aug. 2014.

[29] S. Xinogalos, T. Pitner, M. Ivanović, and M. Savić, ''Students' perspective on the first programming language: C-like or pascal-like languages?'' *Edu. Inf. Technol.*, vol. 23, no. 1, pp. 287–302, Jan. 2018.

[30] S. Nanz and C. A. Furia, ''A comparative study of programming languages in rosetta code,'' in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, May 2015, pp. 778–788.

[31] A. Gomes and A. Mendes, ''A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations,'' in *Proc. IEEE Frontiers Edu. Conf. (FIE) Proc.*, Oct. 2014, pp. 1–8.

[32] D. McCall and M. Kolling, ''Meaningful categorisation of novice programmer errors,'' in *Proc. IEEE Frontiers Edu. Conf. (FIE)*, Oct. 2014, pp. 1–8.

[33] F. B. Tek, K. S. Benli, and E. Deveci, ''Implicit theories and self-efficacy in an introductory programming course,'' *IEEE Trans. Educ.*, vol. 61, no. 3, pp. 218–225, Aug. 2018.

[34] I. Yoon, J. Kim, and W. Lee, ''The analysis and application of an educational programming language (RUR-PLE) for a pre-introductory computer science course,'' *Cluster Comput.*, vol. 19, no. 1, pp. 529–546, Mar. 2016.

[35] O. Ezenwoye, ''What language?—The choice of an introductory programming language,'' in *Proc. IEEE Frontiers Edu. Conf. (FIE)*, Oct. 2018, pp. 1–8.

[36] P. Bhattacharya and I. Neamtiu, ''Assessing programming language impact on development and maintenance: A study on c and c++,'' in *Proc. 33rd Int. Conf. Softw. Eng. ICSE*, 2011, pp. 171–180.

[37] M. Carelli Oliveira Maia, D. Serey, and J. Figueiredo, "Learning styles in programming education: A systematic mapping study," in *Proc. IEEE Frontiers Edu. Conf. (FIE)*, Oct. 2017, pp. 1–7.
[38] A. Luxton-Reilly, J. Sheard, C. Szabo, Simon, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, and M. J. Scott, "Introductory programming: A systematic literature review," in *Proc. Companion 23rd Annu. ACM Conf. Innov. Technol. Comput. Sci. Edu. ITiCSE Companion*, 2018, pp. 55–106.
[39] J. Sorva, "Notional machines and introductory programming education," *ACM Trans. Comput. Edu.*, vol. 13, no. 2, pp. 1–31, Jun. 2013.
[40] K. Benda, A. Bruckman, and M. Guzdial, "When life and learning do not fit: Challenges of workload and communication in introductory computer science online," *ACM Trans. Comput. Edu.*, vol. 12, no. 4, p. 15, 2012.
[41] C. Kazimoglu, M. Kiernan, L. Bacon, and L. MacKinnon, "Understanding computational thinking before programming: Developing guidelines for the design of games to learn introductory programming through game-play," *Int. J. Game-Based Learn.*, vol. 1, no. 3, pp. 30–52, 2011.
[42] S. I. Malik, "Improvements in introductory programming course: Action research insights and outcomes," *Syst. Pract. Action Res.*, vol. 31, no. 6, pp. 637–656, Dec. 2018.
[43] N. C. C. Brown and A. Altadmri, "Investigating novice programming mistakes: Educator beliefs vs. Student data," in *Proc. 10th Annu. Conf. Int. Comput. Edu. Res. ICER*, 2014, pp. 43–50.
[44] A. Vihavainen, "Predicting Students' performance in an introductory programming course using data from Students' own programming process," in *Proc. IEEE 13th Int. Conf. Adv. Learn. Technol.*, Jul. 2013, pp. 498–499.
[45] S. Dasuki and A. Quaye, "Undergraduate Students' failure in programming courses in institutions of higher education in developing countries: A nigerian perspective," *Electron. J. Inf. Syst. Developing Countries*, vol. 76, no. 1, pp. 1–18, Sep. 2016.
[46] T. Crow, A. Luxton-Reilly, and B. Wuensche, "Intelligent tutoring systems for programming education: A systematic review," in *Proc. 20th Australas. Comput. Edu. Conf. ACE*, 2018, pp. 53–62.
[47] J. L. Duffany, "Application of active learning techniques to the teaching of introductory programming," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 12, no. 1, pp. 62–69, Feb. 2017.
[48] S. I. Malik, "Assessing the teaching and learning process of an introductory programming course with Bloom's taxonomy and assurance of learning (AOL)," *Int. J. Inf. Commun. Technol. Edu.*, vol. 15, no. 2, pp. 130–145, Apr. 2019.
[49] M. Sabin *et al.*, *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology*. New York, NY, USA: Association for Computing Machinery, 2017.
[50] J. Polack-Wahl, S. Davies, and K. Anewalt, "A snapshot of current languages used in industry," in *Proc. Frontiers Edu. Conf. Proc.*, Oct. 2012, pp. 1–6.
[51] W. Farag, S. Ali, and D. Deb, "Does language choice influence the effectiveness of online introductory programming courses?" in *Proc. 13th Annu. ACM SIGITE Conf. Inf. Technol. Edu. - SIGITE*, 2013, pp. 165–170.
[52] R. Mason, G. Cooper, and M. de Raadt, "Trends in introductory programming courses in australian universities: Languages, environments and pedagogy," in *Proc. 14th Australas. Comput. Edu. Conf.*, Darlinghurst, Australia, Australian Computer Society, vol. 123, 2012, pp. 33–42.
[53] L. Marshall, "A topic-level comparison of the ACM/IEEE CS curriculum volumes," in *Proc. Annu. Conf. Southern Afr. Comput. Lecturers' Assoc.* Springer, 2017, pp. 309–324.
[54] J. Renzella, A. Cummaudo, A. Cain, J. Grundy, and J. Meyers, "SplashKit: A development framework for motivating and engaging students in introductory programming," in *Proc. IEEE Int. Conf. Teaching, Assessment, Learn. Eng. (TALE)*, Dec. 2018, pp. 40–47.
[55] A. K. Veerasamy, D. D'Souza, R. Lindén, and M.-J. Laakso, "Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses," *J. Comput. Assist. Learn.*, vol. 35, no. 2, pp. 246–255, Apr. 2019.
[56] J. Heliotis and R. Zanibbi, "Moving away from programming and towards computer science in the cs first year," *J. Comput. Sci. Colleges*, vol. 26, no. 3, pp. 115–125, 2011.
[57] C. Heiner, "Mining student responses to learn answer models: A case study using data from an introductory programming course," *J. Comput. Sci. Colleges*, vol. 29, no. 2, pp. 17–25, 2013.
[58] D. Gudmundsen, L. Olivieri, and N. Sarawagi, "Reducing the learning curve in an introductory programming course using visual logic," *J. Comput. Sci. Colleges*, vol. 27, no. 6, pp. 10–12, 2012.
[59] M. Ateeq, H. Habib, A. Umer, and M. U. Rehman, "C++ or Python? Which one to begin with: A Learner's perspective," in *Proc. Int. Conf. Teach. Learn. Comput. Eng.*, Apr. 2014, pp. 64–69.
[60] D. Krpan and I. Bilobrk, "Introductory programming languages in higher education," in *Proc. 34th Int. Conv. MIPRO*, May 2011, pp. 1331–1336.
[61] M. Corney, R. Lister, and D. Teague, "Early relational reasoning and the novice programmer: Swapping as the hello world of relational reasoning," in *Proc. 13th Australas. Comput. Edu. Conf.*, Darlinghurst, Australia: Australian Computer Society, vol. 114, 2011, pp. 95–104.
[62] R. Mason and G. Cooper, "Why the bottom 10% just can't do it: Mental effort measures and implication for introductory programming courses," in *Proc. 14th Austral. Comput. Edu. Conf. (ACE)*, Melbourne, VIC, Australia, vol. 123, 2012, pp. 187–196.
[63] E. de Jesus, "Teaching computer programming with structured programming language and flowcharts," in *Proc. Workshop Open Source Design Commun. OSDOC*, 2011, pp. 45–48.
[64] V. K. Proulx, "Introductory computing: The design discipline," in *Proc. Int. Conf. Informat. Schools, Situation, Evol., Perspect.* Springer, 2011, pp. 177–188.
[65] G. Venugopal-Wairagade, "Study of a pedagogy adopted to generate interest in students taking a programming course," in *Proc. Int. Conf. Learn. Teach. Comput. Eng. (LaTICE)*, Mar. 2016, pp. 141–146.
[66] Á. Matthíasdóttir and H. J. Geirsson, "The novice problem in computer science," in *Proc. 12th Int. Conf. Comput. Syst. Technol. CompSysTech*, 2011, pp. 570–576.
[67] E. Murphy, T. Crick, and J. H. Davenport, "An analysis of introductory programming courses at UK universities," 2016, *arXiv:1609.06622*. [Online]. Available: http://arxiv.org/abs/1609.06622
[68] P. Bachiller-Burgos, I. Barbecho, L. V. Calderita, P. Bustos, and L. J. Manso, "LearnBlock: A robot-agnostic educational programming tool," *IEEE Access*, vol. 8, pp. 30012–30026, 2020.
[69] M. S. Farooq, S. A. Khan, and A. Abid, "A framework for the assessment of a first programming language," *J. Basic Appl. Sci. Res.*, vol. 2, no. 8, pp. 8144–8149, 2012.
[70] J. G. C. Rangel, M. King, and K. Muldner, "An incremental mindset intervention increases effort during programming activities but not performance," *ACM Trans. Comput. Edu.*, vol. 20, no. 2, pp. 1–18, May 2020.
[71] B. Isong, "A methodology for teaching computer programming: First year students' perspective," *Int. J. Modern Edu. Comput. Sci.*, vol. 6, no. 9, p. 15, 2014.
[72] I. Mekterovic, L. Brkic, B. Milasinovic, and M. Baranovic, "Building a comprehensive automated programming assessment system," *IEEE Access*, vol. 8, pp. 81154–81172, 2020.

ERUM MEHMOOD was born in Pakistan. She received the M.Phil. degree in computer science from NCBA&E, Lahore, Pakistan, in 2017. She is currently pursuing the Ph.D. degree with the University of Management and Technology, Lahore.

She is also working as a Lecturer of computer science with the Government Degree College Lahore, Pakistan. Her research interests include programming language education and design, big data analytics, stream processing, and ETL and real-time data warehousing. Her M.Phil. dissertation is in the area of stream processing for real-time data warehousing.

ADNAN ABID (Member, IEEE) was born in Gujranwala, Pakistan, in 1979. He received the B.S. degree from the National University of Computer and Emerging Science, Pakistan, in 2001, the M.S. degree in information technology from the National University of Science and Technology, Pakistan, in 2007, and the Ph.D. degree in computer science from the Politecnico Di Milano, Italy, in 2012. He spent one year in EPFL, Switzerland, to complete his M.S. thesis. He is currently an Associate Professor with the Department of Computer Science, University of Management and Technology, Pakistan. He has almost 70 publications in different international journals and conferences. He has served as a reviewer in many international conferences and journals. His research interests include computer science education, information retrieval, and data management. He is a member of the IEEE Education Society and the IEEE Education Society. He is also an Associate Editor of IEEE Access journal.

MUHAMMAD SHOAIB FAROOQ (Member, IEEE) was born in Lahore, Pakistan. He received the M.Sc. degree from Quaid e Azam University, Pakistan, in 1995, the M.S. degree in computer science from the Government College University, in 2007, and the Ph.D. degree from Abdul Wali Khan University, Pakistan, in 2015. He possesses more than 23 years of teaching experience in the field of computer science. He is currently an Associate Professor with the Department of Computer Science, University of Management and Technology, Pakistan. His research interests include theory of programming languages, big data, the IoT, and computer science education. He is a member of the IEEE Systems, Man, and Cybernetics Society.

NAEEM A. NAWAZ received the B.Sc. degree from the University of Punjab, Pakistan, in 1997, the M.Sc. degree in computer sciences from Hamdard University, Pakistan, in 2000, the M.S. degree in computer engineering from Mid Sweden University, Sweden, in 2008, and the Ph.D. degree from International Islamic University Malaysia, in 2018. He received different diploma (DCS) and certifications. He is currently a Lecturer with the Department of Computer Science (CFY), Umm Al-Qura University, Mecca. He has published research articles in renowned journals and conferences. He is also serving as a reviewer for many journals, conferences, and books. His teaching and research interests include WSN, the IoT, crowd management, computer networks, and programing language.