ISSN: 2473-3857
v4 n4647

Knowledge and Skill Gaps in Programming

Wenli Wang
wangw@rmu.edu

Natalya Bromall
bromall@rmu.edu

Computer and Information Systems
Robert Morris University
Moon Township, PA 15108 USA

Abstract

This study identifies five core programming knowledge areas of *logic*, *programming syntax*, *components and relations*, *security,* and *resources and integration*, based on the observed discrepancies between the industrial demand and the academic preparation for information systems (IS) programmers, the IS model curriculum, and the feedback from programming courses' instructors. It further identifies a list of eighteen specific skills in programming under the five core knowledge areas, and develops a survey instrument as a measurement for these skills and knowledge areas. Data collected from students in basic programming courses show significant gaps between their perceived skills needed for entry-level programming jobs and their current skills across all five knowledge areas but with various gaps. Students perceive to have a small knowledge and skill gap in *logic*, medium gaps in *programming syntax* and in *components and relations*, but large gaps in *resources and integration* and in *security*. Students perceive the largest gap in the area of *security.* These results provide valuable guidelines how to redesign IS curriculum in order to shorten these knowledge and skill gaps, with an emphasis on the curriculum improvement in the knowledge areas of *security* and *resources and integration*.

Keywords: IS skill requirements, job skills, information systems curriculum, integration, security, skill development

1. INTRODUCTION

As the Internet and mobile technology have become essential elements in most of people's lives of this modern age, the demand for newly developed information systems (IS) is also continuously increasing. Correspondingly, the demand for skilled workers in IS programming and development also increases.

Application development jobs make one of the five major categories of IS jobs (Lee, Trauth, & Farwell, 1995). Such a trend has sustained over the past twenty years. The U.S. Bureau of Labor Statistics (BLS) in its most recent handbook in 2016-17 has projected a 27% increase in web developers, a 17% increase in software developers, and an 11% increase in database administrators from 2014 to 2024.

Although BLS (2016-17) has also projected an 8% decline for computer programmers from 2014 to 2024, BLS states that one major reason for the projection of declined employment in computer programmers is that "computer programming can be done from anywhere in the world, so companies sometimes hire programmers in countries where wages are lower." This may not be the case anymore for some countries; for example, it has said way back in 2008 that outsourcing in China would be no longer cheap (Maltby, 2008a). Employers who create jobs with benefits for American workers are provided tax cuts, according to the Patriot Employer Act of 2007 (Maltby, 2008b). The situation may change

even more under the new Trump administration, which is eager to bring the typically-outsourced jobs back to the U.S. soil.

Many companies already bring programming jobs back to U.S. from overseas for a variety of reasons. For instance, the communication and collaboration are a lot easier when the entire team is in the same (or close) time zones and speaks the same language. This is important in any project development, but simply crucial for some development methods such as agile. In addition, the code developed in-house typically has much better comments and documentation. Finally, maintenance and upgrading become much easier as well when the project is in-house developed.

Currently, about 40% of the programming jobs in the U.S. are filled by qualified immigrants, and about 65% of H1-B visa workers were in computer-related occupations in 2014 (DHS, 2015). But with the new Trump administration's intent of tightening H1-B visas, there will be more programming jobs open for the domestically-trained programmers.

In addition, cybersecurity is a growing concern for almost all sectors. BLS (2016-2017) projected an 18% increase for information security analysts between 2014 and 2024, and many of these jobs require high levels of security screening. This implies the further demand for home-grown programmers, especially those with the U.S. citizenship.

Hence, the programming job market for domestic programmers will grow in the next decade. The question is whether or not the IS education in the U.S. prepares its graduates with the right knowledge and skills set for this growing job market.

Knowledge and skills in programming have always been one of the major requirements for the IS profession. Lee et al. (1995) identified programming as a major "technical specialties knowledge," together with knowledge about networking, operating systems, systems analysis, data management, decision support, and artificial intelligence.

Knowledge and skills in programming are not simply referring to knowing about programming languages. BLS (2016-17) regards computer programmers as those who "specialize in a few programming languages…write and test code …turn the program design created by software developers…into instructions that a computer can follow…" and software developers as those who "develop applications" or "systems" with "strong computer programming skills" and "are the creative minds behind computer programs." Hence, it is the strong programming skills rather than the specialization in programming languages that secure a higher-paid job ($100,690 median pay for software developer vs. $79,530 for computer programmers in 2015) with a more prosperous job outlook. This may also explains why BLS has conflicting outlooks in software developer jobs (17% increase) and computer programmer jobs (8% decrease) from 2014 to 2024.

In addition, with the increasing demand for information security analyst (18% increase 2014-2024), systems analysts (21% increase), and IS management (15% increase) (BLS, 2016-17), it is important to consider what knowledge and skills in programming should prepare students to either become these specialists themselves or have effective collaborations with these analysts and managers. For instance, are the programming skills in secure programming, program integration, and technology management as important as the typical skills in programming logic and syntax?

Peslak et. al. (2018) identified and ranked the tops key words in job descriptions for programmer analysts. They found that while the programming/database languages such as SQL, .NET, C#, and Java are at the top of the list for specific technical skills, security and integration are among the top 25 most common key words in the general systems skills.

The objective of this research is to identify a list of programming skills and a categorization of knowledge areas in programming. It also measures the perceived gaps in these programming skills and the core knowledge areas for the purpose of IS curricula redesign. The key research questions are:

1. What are the discrepancies between the industrial demand and academic preparations in programming knowledge and skills?
2. What are the core knowledge areas and specific skills in programming?
3. What are the students' perceived gaps between the knowledge and skills needed for entry programming jobs and their current knowledge and skills?

The study proposes the core knowledge areas and the specific skills in these knowledge areas based on investigated possible discrepancies between

the industry and the academic preparations, the literature in IS curriculum design, and the feedback from experienced instructors of programming courses. A survey instrument based on the core knowledge areas is then designed for examining the gaps in students' perceptions between the knowledge and skills needed for the entry-level programming job and the students' current skills. These gaps then further serve as a feedback input for the design of IS education in programming.

2. ACADEMIC/INDUSTRY INVESTIGATION

Knowledge and Skills vs. Technical Change

The IS industry progresses forward at a rapid speed, with non-stop changing and evolving landscapes. Although it has long been called for even in 1990s the corresponding re-structuring of IS curricula at universities with the realignment of IS activities in the industry (Lee et al., 1995), the IS curricula have not been able to re-structure at the pace that the industry demands for.

Academics, as a traditionally much more stable and static profession, cannot, and sometimes do not need to, evolve as fast as the IS industry. Hence, academics have not fully satisfied the changing demand of the industry, not to mention to move ahead of the curve to educate students with forecasted knowledge and skills that would satisfy the anticipated changes in the industry. In other words, academics may never be able to move ahead of the industry. Typically, academics try to satisfy the industrial demand from a feedback loop. However, when the feedback from the graduated alumni and their employers reach the academic programs, the technology in question could quite possibly lag behind the new technical frontier, or even become obsolete while the academic programs are still teaching the outdated technology to students in line for graduation.

There are two possible solutions to this ill-match problem between industrial needs and IS curricula: 1) to take an adaptive agile approach that encourages more frequent industrial feedback and tighter academic-industry collaboration; and 2) to have academics focus on educating and training students the essential knowledge and skills that remain intact with the technical landscape change.

Solution one is more easily said than done because of the following reasons: a) the academic programs need to satisfy not only industrial demand, but also, and maybe more importantly, other stakeholders' requirements such as those of the accreditation agencies which are not evolving as rapidly as those of the industry and hence may result in discrepancies; and b) both the academics and the industry may not have the adequate resources for more frequent interactions and tighter collaborations.

Solution two is more feasible as it mainly involves the academics. It is therefore important to systematically identify and categorize what knowledge and skills are essential for the IS profession (such as programming logic, syntax, etc.), as well as the self-learning skills and knowledge expansion for the future growth in careers and outside the classroom settings (such as resources identifications and integration). Most programming courses focus on programming logic, syntax, and basic structures rather than the broad knowledge and skills.

Academic Preparations

The students are introduced to a programming language in an introductory programming course (mostly C++ or Java). They start programming by building simple console applications. Parallel to this, they learn some basic programming logic and the programming syntax of the particular language. The students are then introduced to Graphic User Interface (GUI) and the programming environment designed for building graphical interfaces (such as Visual Studio .NET for Visual Basic or C#, Eclipse for Java, etc.). Finally, the students learn to build desktop applications and (sometimes) connect these applications to local data sources or the basic Web. This is where the sequence typically stops. It skips such important steps of complex web and mobile development and integration with important components such as external databases and web services like Paypal or the backend credit card processing.

The current textbooks are either lacking the "integration" element, or it has come too late in the last chapters or appendices as optional or unimportant. Since there is no common curriculum structure of which programming language(s) are to be taught as required core curriculum, there is a lacking of coordination of which content has been covered prior to a programming course. For instance, it is unclear if students have learned database prior. Hence, the textbook authors try to be as comprehensive as possible for introductory content on certain subjects and such comprehension has a trade off in depth.

Industrial Demand
The current trend in the job market for programming/application development is to focus more and more on logic and integration skill sets as opposed to specific programming languages.

For the purpose of this study we reviewed a random sample of approximately 100 programming/application development job announcements posted by high-tech Fortune 500 companies such as Google, Amazon, and Facebook. The qualification requirements in every announcement typically include one line of specifying a programming language (Java, C++, C#, Python, etc.). Each announcement also includes at least five specifications of other (non-programming language) skill requirements such as integration with SQL database, strong object-oriented programming (OOP) skills, knowledge of GUI design and Human Computer Interaction (HCI), etc. We observed the same pattern in the recruitment of large brick-and-mortar companies (such as Ford, General Motors, etc.) and in the service-based companies (such as Monster, UPS, etc.).

Discrepancies: Academics Preparations vs. Industrial Demand
With the fast changing technology, academia needs to quickly adapt to the industrial demand and the employer requirements to ensure that the recent graduates have marketable skills and can find jobs up to their potential. For example, based on the reviewed positions, there is a consistent message from the job market that OOP is a necessity. The good news is that most advanced programming courses do cover OOP in depth, making this demand fulfilled.

At the same time, most companies require integration between the applications and enterprise-scaled databases (such as MS SQL and Oracle), but unfortunately, this demand is not fulfilled in most programming courses. Some of the advanced programming courses include the basics of data connection using the standalone databases such as Access, which is not preparing the students for the career of a developer.

Companies often require the knowledge of a specific application-type building, such as Windows desktop, enterprise, or web service; this requirement is also rarely fulfilled by the academic IS curricula.

The discrepancies between what industry requires and what IS education can provide has further accentuated by the lack of proper textbooks and the difficulty of setting up a functioning real-life environment, such as a connected Application Server and/or Database Server. This problem is easier to solve with the emerging new technologies, such as Amazon or Microsoft Cloud that provides various virtual servers (e.g., Windows Server with IIS, SQL Server, etc.).
In summary, current academic preparations in IS education are not sufficient for satisfying the industrial demand in the skill sets needed for application development and programming jobs.

3. KNOWLEDGE AREAS AND SKILLS IN PROGRAMMING

Based on our observations of the discrepancies between industrial demand and academic preparations for the knowledge and skills needed for programming jobs, the 2010 model curriculum and guidelines for undergraduate degree in IS recommended by the Association of Computing Machinery and Association of Information Systems (Topi et al., 2010), as well as our informal interviews with five IS faculty who have extensive teaching experiences in programming and application development courses at a private university, we identify five core knowledge areas in programming. These areas are as follows: 1) logic; 2) programming syntax; 3) components and relations; 4) security; and 5) resources and integration.

Logic
In an introductory or basic programming course, it is the logic knowledge unit and the critical thinking that should be emphasized. Such knowledge and critical thinking capacity can be easily applied to any other programming language further in the IS curriculum. Students in the study of their second programming language just need to implement their learned logic in a different programming syntax. In this way, their existing logic can be further cultivated and extended.

Programming Syntax
Programming syntax is typically well-covered in textbooks and in IS education. Otherwise, programs simply do not execute and students hence cannot learn about a programming language. Students may easily forget the specific programming syntax of a language after the course is over, in case the students no longer program in that particular language; however, their critical thinking capability such as in the similarities or differences in different syntax would have longer retention. The applications of logic in various programming syntax in different programming implementations can also help

students further develop their knowledge and skills in logic.

Components and Relations
An introductory programming course should heavily focus on logics with a minor emphasis and the exemplary explanation of the structure of the programs and their building blocks. Logic also plays an important role to demonstrate how the building blocks are related and later integrated.

The computing environments such as desktop, web, mobile, etc. are the context of programs, and these contexts have commonalities. As noted by the professional developers (exemplified by contributors in Software Engineering forums (Software Engineering, 2018)), the mastering of programming in different contexts "comes later" than the programming logics itself. Critical thinking with logics comes ahead of programming environment.

Security
Current programming textbooks typically do not cover much or emphasize security at all. Security is often an after-thought, or a knowledge taught in a specific course in security. We suggest that security is so essential and pervasive nowadays that students even in their introductory programming courses should be well-exposed in this topic and let them implement secure programming practices along with other featured functional programming.

Resources and Integration
The Resources and Integration knowledge unit includes technology management skills such as identifying resources and integrating the found resources. The knowledge given to the students in the programming courses is always limited. No matter how much information the students have learned in the process and in the IS curricula, it is typically not enough for most industrial positions that require experiences.

Therefore, one of the main goals of a programming course is to teach the students to find information and identify resources and utilize the information and resources to satisfy the demand from the industry and to grow as a developer. There is a point in every professional's career that he/she runs into a question that cannot be answered by textbooks or by other fellow developers. In fact, one of the most crucial professional skills is to identify the right information and resources that would lead to the answer to the question. The resources are not limited to just forums and tutorials, but also include off-the-shelves packages that can be integrated with in-house development for an application.

Programming Skills
Based on the suggested five core knowledge areas, a survey instrument was designed to measure the specific programming skills of each core knowledge area. There were eighteen (18) questions that covered the five core knowledge areas (see Appendix A).

The eighteen questions were designed with the current technologies in mind and were partially based on the IS 2010 curriculum guidelines the for undergraduate degree programs in IS by the Association of Computing Machinery and the Association of Information Systems (Topi et. al., 2010). These questions are listed in the Appendix A with their corresponding knowledge areas. However, in the survey issued, the respondents would not be shown the specific five core knowledge areas that are being surveyed.

The survey instrument asks the respondents to provide their perceptions on the skills needed in an entry-level programmer job position and their current skills, on a 1-5 scale (1: never, 2: rarely, 3: sometimes, 4: often, 5: always). The respondents can also answer "don't know".

4. METHODOLOGY

The survey was issued to four (4) beginning programming courses (2 courses in Visual Basic, 1 course in C#, and 1 course in C++) in an IS program at the start of the Spring 2017 semester in a private university. Since the survey was issued at the very beginning of the courses, the students have not received much teaching from the teachers yet. All of these four courses are tailored to the beginners in programming and should be the first basic programming course taken by the students. Since there are advanced programming courses (such as Visual Basic II) offered as follow-up courses of the basic courses, students typically would take the advanced follow-up courses with the same programming language rather than retake another basic programming course. Students were asked to voluntarily answer the surveys.

There were 66 responses altogether. Among them, 9 did not answer or stated "don't know" on all of the 18 questions regarding "skills needed" and 9 had left empty or claimed "don't know" on more than 6 questions regarding "current skills." These responses were not used for data analysis. For the remaining 57 responses regarding "skills needed," there were few (40 out of 57*18 =

3.8%) questions that were not answered or provided the "don't know" answer by some students and these scores were replaced with the corresponding average scores. For the remaining 57 responses regarding "current skills," there were few (18 out of 57*18 = 1.7%) questions that were not answered or provided the "don't know" answers by some students, and these scores were replaced with the corresponding average scores. Table 1 provides the summary statistics among the 57 responses to the eighteen questions.

	Skills Needed	Current Skills
Mean	3.61	2.33
Variance	0.08	0.21
Observations	18	18
df	34	
t Stat	10.16	
P	<0.001	

Table 1. Overall Programming Knowledge and Skill Gaps

Based on the average scores and the standard deviations for each question, there were no scores outside the range of the three standard deviations from the average score. Hence, there were no outlier scores among the 57 responses that need to be deleted. As shown in Table 1, the average score for students' perceived skills needed in an entry-level programmer job is 3.61 and the average score for students' perceived current skills is 2.33. There is significant difference ($p < 0.001$, $t = 10.16$) between the perceived skills needed and the perceived current skills. This result implies that there is a need for IS education for students to perceive their readiness for the entry-level programming jobs.

Appendix B shows that the students' perceptions in the skills gap (skills needed in entry programmer job minus current skills) in the five core knowledge areas are, on average, small (gap < 30%) in logic, medium (30% ≤ gap < 35%) in programming syntax and in components and relations, and large (gap ≥ 35%) in security and in resources and integration. The average perceived skills gap in all questions is medium (32%).

Interestingly, for the individual questions, it is the skill in "programming security" that has the largest perceived skills gap (40%) and it is the skills in "programming logic" that has the smallest perceived skills gap (18%). Students, even those with their first programming courses, have been exposed to the importance of secure programming and realized that there is a great demand for secure programming even in entry programming jobs; however, students' current skills in this area are perceived to be relatively low. In the components and relations knowledge area, the perceived skills gaps are also large in Web programming (38%), data programming (36%), console programming (36%), scripting programming (36%), mobile programming (35%), and cloud programming (35%). In the resource and integration knowledge area, the gap in resources is large (37%).

The difference between the students' perceived current skills and the skills required in industry is statistically significant for all 18 items (see Appendix C). The difference was from the 57 students who answered almost all questions for both needed skills and current skills. Few missing scores are replaced with the mean. Again, the results show that the students from the introductory programming courses believe that they do not currently have enough skills to start working as programmer/application developers.

5. DISCUSSIONS

Data analysis of the programming skills survey shows a significant perceived gap in knowledge and skills needed for entry programming job and students' perceived current knowledge and skills. This implies that IS education should target at shortening the gap for the proper academic preparations that satisfy the industrial demand.

The survey results show a small gap in logic, medium gaps in programming syntax and in components and relations, and a large gap in security and in resources and integration. Even at the start of the basic programming course, students believe they have somewhat sufficient logic to do programming, it is the knowledge in programming syntax that need to obtain. Hence, the basic course should not emphasize logic, especially the programming logic that students may have already been exposed in other prerequisite courses. However, how to introduce and enhance security in programming should be emphasized.

With the gradual coverage of advanced topics such as data, web, and mobile programming, etc., and their relations in higher level programming courses, more emphases could be put on educating about resources and integration that will put components and pairwise relations

into systems programming and networked relations.

This research currently only surveyed undergraduate students at the start of their introductory programming courses. Future research will also survey students pre- and post-advanced programming courses and it is hypothesized that, if proper emphases in core knowledge areas are put at the right stages of programming education, students at their last programming course before graduation should have the minimum knowledge and skill gaps between their perceived skills needed for entry-level programming jobs and their then programming skills.

Future research will also examine the graduate-level curricula in cultivating programming knowledge and skills. As task complexity increases, skills such as collaboration in a pair programming setting may be needed (Balijepally et al., 2009), which may require additional core knowledge areas to be considered.

6. CONCLUSIONS

Aligning IS education with industrial demand and building students' core knowledge and skills in programming that satisfy the ever changing industrial demand are among the high priorities in IS education. Our study suggests that in order to satisfy the industrial current and future demand for a cadre of IS professions, especially with the programming knowledge and skills, academics should systematically educate students with the core knowledge areas.

This research identify the five core knowledge areas in programming of logic, programming syntax, components and relations, security, and resources and integration. It further identify a list of eighteen specific skills in programming and develop a survey instrument as a measurement for the skills and related knowledge areas. These five core knowledge areas and a list of eighteen programming skills can serve as a blueprint for systematic development of programming knowledge and skills in the IS curriculum. Gaps found in students' perceptions in programming knowledge and skills inform how to redesign the IS curriculum. For instance, programming knowledge and skills in security has the largest gap and IS curriculum should strengthen the education in secure programming.

7. REFERENCES

Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are Two Heads Better than One for Software Development? The Productivity Paradox of Pair Programming. *MIS Quarterly* 33 (1), 91-118.

Bureau of Labor Statistics, U.S. Department of Labor, Occupational Outlook Handbook, 2016-17 Edition, Computer and Information Systems Managers. Retrieved July 15, 2018 from https://www.bls.gov/ooh/management/computer-and-information-systems-managers.htm.

Lee, D. M. S., Trauth, E. M., & Farwell, D. (1995). Critical Skills and Knowledge Requirements of IS Professionals: A Joint Academic/Industry Investigation. *MIS Quarterly* 19(3), 313-340.

Maltby, E. (2008a, August 19). In China, outsourcing is no longer cheap. CNNMoney.com, Retrieved July 15, 2018 from http://money.cnn.com/2008/08/11/smallbusiness/china_no_longer_cheap.fsb/index.htm?postversion=2008081910.

Maltby, E. (2008b, November 11). Overseas outsourcing heats up again. CNNMoney.com, Retrieved July 15, 2018 from http://money.cnn.com/2008/11/10/smallbusiness/outsourcing.smb/index.htm?postversion=2008111106.

Peslak, A., Kovalchik, L, Kovacs, P., Wang, W., Conforti, M., & Bhatnagar, N. (2018). Linking Programmer Analysts Skills to Industrial Needs: A Current Review," *Proceedings of the EDSIG Conference on Information Systems and Computing Education*, Norfolk, VA.

Software Engineering: Programming knowledge vs. programming logic, a professional forum. Retrieved July 15, 2018 from http://softwareengineering.stackexchange.com/questions/155679/programming-knowledge-vs-programming-logic.

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K. M., Nunamaker, J. F. Jr., Sipior, J. C., & de Vreede, G. J. (2010). IS 2010 Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. *Association for Computing Machinery* (ACM) and *Association for Information Systems* (AIS). Retrieved July 15, 2018 from https://www.acm.org/binaries/content/assets/education/curricula-recommendations/is-2010-acm-final.pdf

APPENDICIES

Appendix A. Programming Skills Survey

Knowledge Areas	Survey Questions	Skills needed (1–5)	Current skills (1–5)
Logic	Programming logic (pseudo code, flow charts, etc.)		
Programming Syntax	Language syntax (variables, conditions, loops, arrays, etc.)		
Components & Relations	Programming graphic user interface (text fields, buttons, menus, etc.)		
	Object-oriented programming (creating and using classes and objects, etc.)		
	Organization of a program (functions, procedures, modules, etc.)		
	Operating system programming (file management, system setup, etc.)		
	Data programming (data controls, database connection, query, etc.)		
	Desktop programming (windows application, etc.)		
	Console programming		
	Web programming (client/server, etc.)		
	Mobile programming (android, iOS, etc.)		
	Cloud programming (SaaS (software as a service), etc.)		
	Ubiquitous programming (smart appliances, etc.)		
	Interface programming (CSS, HTML, etc.)		
	Scripting programming (cookies manipulation, JavaScript, Ajax, etc.)		
Security	Programming security (input validation, encryption, etc.)		
Resources & Integration	Reuse and integrate with existing components (customized controls, etc.)		
	Programming resources (IDEs, libraries, forums, references, etc.)		

Appendix B: Programming Knowledge and Skill Gaps

Knowledge Areas	Survey Questions	Skills needed		Current skills		Skills Gap			Knowledge Gap		
		Ave	Std	Ave	Std	Ave	%	Size	Ave	%	Size
Logic	Programming logic	4.00	0.98	3.29	1.19	0.71	18%	S	0.71	**18%**	**S**
Programming Syntax	Language syntax	4.06	1.06	2.84	1.10	1.22	31%	M	1.22	31%	M
Components & Relations	Programming GUI	3.89	1.03	3.00	1.34	0.89	22%	S	1.20	32%	M
	Object-oriented programming	3.77	0.87	2.82	1.30	0.94	24%	S			
	Organization of a program	3.82	1.04	2.72	1.31	1.10	28%	S			
	Operating system Programming	3.74	0.95	2.45	1.16	1.29	32%	M			
	Data programming	3.66	0.93	2.20	1.14	1.46	**36%**	**L**			
	Desktop programming	3.72	1.01	2.47	1.23	1.25	31%	M			
	Console programming	3.42	1.15	1.98	1.14	1.44	**36%**	**L**			
	Web programming	3.51	1.14	1.98	1.01	1.53	**38%**	**L**			
	Mobile programming	3.26	1.12	1.86	1.14	1.4	**35%**	**L**			
	Cloud programming	3.16	1.15	1.75	1.12	1.41	**35%**	**L**			
	Ubiquitous programming	2.94	1.16	1.63	1.05	1.31	33%	M			
	Interface programming	3.58	1.16	2.23	1.34	1.35	34%	M			
	Scripting programming	3.53	1.22	2.11	1.22	1.42	**36%**	**L**			
Security	Programming security	3.78	1.01	2.18	1.10	1.6	**40%**	**L**	1.6	**40%**	**L**
Resources & Integration	Programming resources	3.57	1.22	2.11	1.21	1.46	**37%**	**L**	1.39	**35%**	**L**
	Reuse and integrate	3.63	0.98	2.32	1.15	1.31	33%	M			
	Average	3.61	1.06	2.33	1.18	1.28	32%	M	1.29	32%	M

Skills gap sizes: Small, gap <30%; Medium, 30% ≤ gap < 35%; Large, gap ≥ 35%. % is obtained by the average gap/4 (=5 -1). N = 57.

ISSN: 2473-3857
v4 n4647

Appendix C: Programming Knowledge and Skills Gaps in paired differences

Paired Differences		Mean	Std	SE	t	Sig. (2-tailed)
Logic	Logic	.840	1.695	.240	3.505	.001
Programming Syntax	Syntax	1.100	1.515	.214	5.133	.000
Components & Relations	GUI	.900	1.607	.227	3.961	.000
	OO	1.040	1.551	.219	4.740	.000
	Org	1.060	1.570	.222	4.773	.000
	OS	1.340	1.745	.247	5.430	.000
	Data	1.640	1.638	.232	7.078	.000
	Desktop	1.240	1.598	.226	5.487	.000
	Console	1.540	1.487	.210	7.321	.000
	Web	1.760	1.422	.201	8.750	.000
	Mobile	1.600	1.690	.239	6.693	.000
	Cloud	1.640	1.651	.233	7.025	.000
	Ubiquitous	1.500	1.515	.214	7.000	.000
	Interface	1.380	1.839	.260	5.305	.000
	Scripting	1.400	1.784	.252	5.548	.000
Security	Security	1.720	1.863	.263	6.528	.000
Resources & Integration	Resources	1.480	1.644	.233	6.365	.000
	Integrate	1.440	1.740	.246	5.853	.000

N=57, df = 56