International Journal of Multidisciplinary Education and Research
ISSN: 2455-4588; Impact Factor: RJIF 5.12
www.educationjournal.in
Volume 2; Issue 1; January 2017; Page No. 42-46

A preliminary investigation of programming in curricula outside computer science

[*1] Christopher Scaffidi, [2] Balaji Athreya

[1] School of Electrical Engineering and Computer Science, 1148 Kelley Engineering Center, Oregon State University Corvallis, OR, United States
[2] eBay, San Jose, CA

Abstract
As programming has risen in importance among professionals in diverse domains, the software engineering research and education community has contributed domain-appropriate courses and tools to aid in the training of students headed for these professions. However, data are lacking on the extent to which courses exist in other majors, as well as their nature. Therefore, this paper presents a preliminary study investigating the extent to which curricula in business, biology and engineering include training in programming, software engineering, and different kinds of programming languages. This investigation revealed that there was more uniform prevalence of programming among engineering schools than among biology (with business in between), although programming figured prominently in all three disciplines' syllabi. The results indicate research and education aimed at helping to master end-user programming, as well as software engineering practices, could impact a range of disciplines.

Keywords: end-user programming, end-user software engineering, curricula

1. Introduction

Programming plays a prominent role in the work of scientists, engineers, and business professionals [10, 12, 15, 17]. These diverse people do what is often called "end-user programming" (EUP), defined as *creating code for one's own use rather than for use primarily by others* (as is the case with what is called "professional programming") [15]. They typically use their code for work or hobbies. In response, researchers have offered domain-appropriate tools for end-user programmers in different contexts [1, 5, 7, 11, 15, 18, 19], in recognition of the fact that end-user programmers sometimes use languages and associated tools such as spreadsheets that differ from the textual languages favored by most professional software engineers [15].

An important part of education is to prepare students for disciplinary practices [8]. Future scientists, for example, learn about practices of testing hypotheses against nature, as well as sharing and critiquing scientific claims [8]. Given the rise of EUP in so many disciplines' professional practices, educators have introduced programming into K-12 and college courses. While some focus on programming [2, 19], others [4, 14, 16] incorporate coverage of validation, testing, debugging, agile, architecture, and other concerns of software engineering. This synthesis yields end-user software engineering (EUSE), defined as *methods and supporting technology to support quality in EUP* [13].

However, a missing component in the work cited above is a dearth of statistics on whether and how universities teach EUP or EUSE in disciplines outside computer science. In particular, it is unclear (a) the *extent* to which curricula include courses in EUP, (b) what *kinds* of languages are used in these courses, and (c) whether these courses also cover *EUSE*. Due to this three-fold gap in our knowledge, the applicability of the related work, cited above, remains vague. For example, when a paper presents a new course in EUP or software engineering topics at a given school (e.g., [19]), it is hard to distinguish whether that course is novel only at that given institution or, rather, innovative relative to most universities more generally. For tool papers, because we do not know what kinds of languages are used in discipline-specific curricula, we cannot judge to what extent a new tool will be compatible with existing courses.

Consequently, this paper presents a preliminary study of syllabi in disciplines outside computer science, with a focus on generating insights into the prevalence of EUP, EUSE and different kinds of languages.

2. Materials and methods

Our study proceeded in four stages. (1) We acquired a list of well-respected universities in business, biology, and engineering. (2) We used a web search engine to count mentions of programming tools and software engineering topics in each school's syllabi. (3) We normalized these data based on university size and analyzed the prevalence of search terms. (4) We performed a qualitative analysis of a sample of business syllabi, to ascertain the extent to which the syllabi were related to promotion of programming skills.

2.1 A sample of well-respected universities

Not all universities offer programs in all disciplines, so we needed to identify a sample representative of those that were well-established in a given discipline. We relied on the US News and World Report rankings of the top 100 in business, biology, and engineering. (To avoid confusion with an executable "program," we use "school" below to refer to a university program in one of these disciplines.)

Given this ranking's limitations, including for example the potential for universities to manipulate statistics [6, 9], we avoid assuming that rank indicates a school's precise level of quality. Rather, for almost all analyses below, we simply treat the

school's rank in the top 100 as evidence that the university does, in fact, offer a credible program in that discipline.[1] For a single analysis, we binned schools by quartile; so, in that one analysis alone, we assumed that schools were similar to those with roughly similar rank.

2.2 Data acquisition

For each school, we looked up university size in the Integrated Postsecondary Education Data System[2] operated by the US Department of Education. We were unable to locate one university from the biology list, so we removed it from our sample. For the remaining 299 schools, we had now obtained Rank and Enrollment (Table 1).

We wrote a script that, for each school in each discipline's list, issued five queries to the Bing search API and retrieved the number of search results (capped at 500) [3]. The first query had a set of topics covering a broad range of different kinds of programming languages (Table 1, first row of the Prevalence subtable). Based on the information available about programming activities of professionals in the disciplines [10, 12, 15, 17], this list of topics included spreadsheets, Matlab and textual languages. (We would have included "C" but worried including it would generate false matches, due to "C" having many other meanings aside from the language name.) As shown in Table 1, our script also issued queries for just spreadsheets, just Matlab, and just textual languages (specifically, Fortran, Perl, Java and Python), enabling us to disaggregate the set of search results into subcounts for each kind of language.

Finally, as shown in the last row of Table 1, the script queried for pages mentioning software engineering topics focused on testing and debugging, independent of any programming language. (Future studies could include other terms such as "requirements" and "architecture.")

2.3 Statistical analyses

For each school, we calculated a measure we call "Programming Prevalence" (PP) by dividing the number of hits for the Programming query by the Enrollment, to take into account the fact that larger universities typically offer more courses. We computed a histogram for each discipline, enabling us to visualize how the distribution of PP levels varied within and among disciplines.

In addition, for each quartile (based on US News and World Report Rank) in each discipline, we

Table 1: Variables used for statistical analyses.

Rank	**Discipline-specific school rank in *US News and World Report* (where 1 indicates best)**
Enrollment	2014-15 enrollment (in thousands), a measure from IPEDS including undergraduates and graduates that accounts for credit-hours and/or contact-hours
Prevalence	Number of results (capped at 500) divided by Enrollment; queries were of the form (*<topic-set>) AND (course AND syllabus AND <discipline>) AND site:<university>.edu*

Five different <topic-sets> were queried for each combination of <discipline> and <university>…

Programming	Excel OR spreadsheet OR Matlab OR Fortran OR Perl OR Java OR Python
Spreadsheets	Excel OR spreadsheet
Matlab	Matlab
Textual	Fortran OR Perl OR Java OR Python
Software Engineering	("software engineering" OR debugging OR testing) AND (software OR programming)

1 http://www.usnews.com/rankings
2 https://nces.ed.gov/ipeds/datacenter/
3 http://datamarket.azure.com/dataset/bing/searchweb

Computed the average PP level. This revealed how PP varied among quartiles within each discipline.

We computed what we call "Software Engineering Prevalence" (SEP) for each school by dividing the number of results for the Software Engineering query by Enrollment. We used linear regression of SEP versus PP, once per discipline, to assess if the prevalence of software engineering topics generally tracked with that of programming topics.

Finally, we used the counts from the other three queries, specific to kinds of languages, to disaggregate the overall PP according to each kind of language. This indicated, for example, the proportion of business search results that were attributable to spreadsheets, the proportion to Matlab, and the proportion to textual languages. In cases where a given page mentioned more than one kind of language, its weight was divided equally among those languages.

2.4 Qualitative analysis

We used a qualitative analysis to evaluate our study's premise that search results reflected programming-related training. For this analysis, we focused on data retrieved for "Excel OR spreadsheet" in business. One of our team's two researchers randomly selected 50 schools. For each, we examined the first webpage returned by the search.

We were not aware of any general rubric for determining whether a course teaches programming. However, we found a rubric for assessing how well a lesson plan covers computational thinking, which encompasses programming and other related issues (such as problem decomposition) [3]. This rubric assigned a 0, 1 or 2 for each of 10 dimensions:

- Data Collection
- Data Analysis
- Data Representation
- Problem Decomposition
- Abstraction
- Algorithms and Procedures
- Automation
- Parallelization
- Simulation
- Connection to Other Fields

A score of 0 meant "Not Incorporated," a 1 meant the teacher demonstrated that dimension but students had little active role in applying it, and a 2 meant the student played an active role in applying it. For example, a score of 1 for the Simulation dimension would indicate that the teacher demonstrated a simulation, but students did not create simulations of their own.

We filtered the 10 dimensions to include those with good reliability. Specifically, the "Data Representation" dimension was too vaguely specified for us to interpret precisely, so we dropped it. We wrote more specific rules for ourselves about how to interpret the others. Then, we independently scored 20% of the syllabi and computed inter-rater agreement and Spearman correlation. We found 4 dimensions had an inter-rater agreement of at least 80% and an average Spearman coefficient of 0.68, which compared well to an average inter-rater agreement of 62% and -0.04 for the other 5 dimensions. Thus, we retained these 4 dimensions: Algorithms and Procedures, Automation, Simulation, and Connection to Other Fields.

As a final reliability check, we computed the average score between the two of us for each dimension on each of the test web pages, then calculated Cronbach alpha. The result, 0.71, indicated we could combine each page's 4 scores for the dimensions into a single overall average scale.

3. Results

3.1 Statistical results

Statistical results of the three disciplines, engineering had the highest PP. Specifically, for any given level of PP, the proportion of engineering schools at or above that level exceeded the proportion of biology or business schools at or above that level (Fig. 1). Approximately 75% of engineering schools had a higher PP than 75% of biology schools did (i.e., PP ≥ 4). Looking horizontally across Fig. 1, biology had a median PP of 2.3, business 4.29, and engineering 6.45. Averaging by quartile showed better-ranked schools generally had higher PP levels than less well-ranked schools (Fig. 2). The sole notable exception was that engineering's PP rose slightly in the final quartile.

SEP closely tracked PP (Fig. 3). Its ratio to PP was highest among engineering schools, averaging 75%, versus 62-64% in biology and business. Thus, not only did engineering have the highest level of PP (Fig. 1), but it also had the highest level of SEP.

Breaking down the total PP among kinds of languages revealed some surprises (Fig. 4). Though business schools mostly used spreadsheets, they also used textual languages. And despite Matlab's reputation as an engineering language, it appeared for the other disciplines, as well.

3.2 Qualitative results

The qualitative analysis of 50 sample business results for "Excel or spreadsheet" indicated half described courses involving assignments requiring students to automate procedures/algorithms specified by the teacher (Fig. 5): for example, some directed students to automate company-valuation calculations, and several courses provided data to students (such as hypothetical company sales or operations data) for use in revenue or cost projections. The other half of the pages mentioned spreadsheets only in general terms, not in the context of programming demonstrations or assignments.

Overall, 16% of courses had an average score of 0 across dimensions (i.e., no meaningful instruction of computational thinking at all), 58% scored 0.75 or higher (e.g., having a 1 on at least 3 of the 4 dimensions), and only 8% had an overall scale score of at least 1.5 (e.g., having a 2 on at least 3 of the 4 dimensions).

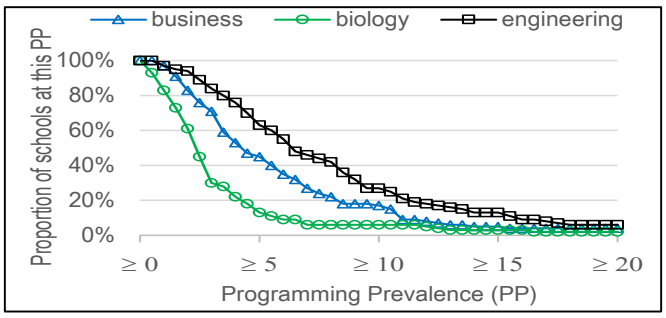

Fig. 1: Distribution of schools' programming prevalence, for each discipline.

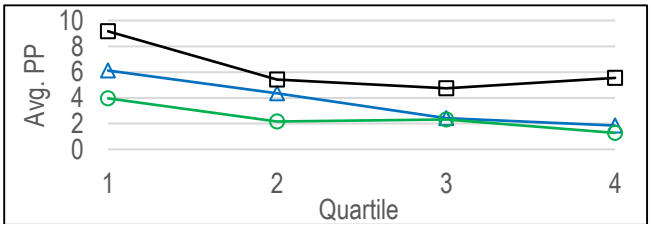

Fig. 2: Average Programming Prevalence (PP) for schools in each quartile (1=best) for each discipline.

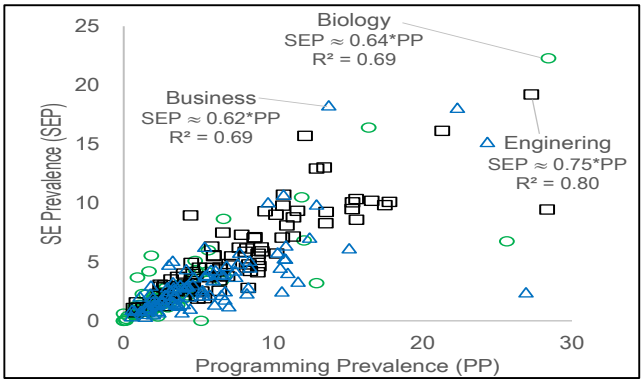

Fig. 3: SE Prevalence (PSE) versus Programming Prevalence (PP), with linear fit shown for each discipline.

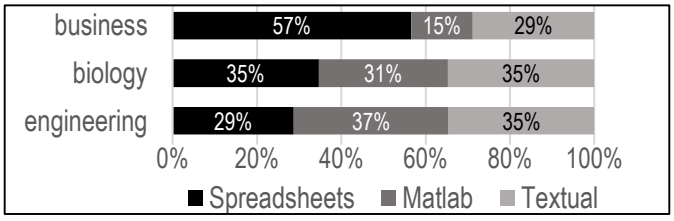

Fig. 4: Distribution of web pages returned for pp among types of programming languages/tools

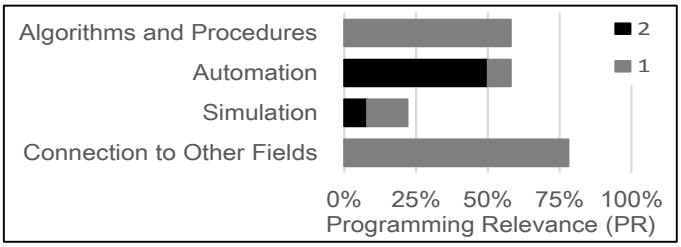

Fig. 5: Distribution of sample pages' scores on computational thinking dimensions, where, 1=teacher demonstrates, 2=student creates.

4. Discussion

The results illuminate three aspects of the threefold-gap in our knowledge that motivated this work:

(a) Although virtually all schools mentioned EUP in syllabi, the prevalence varied widely. The distribution of EUP's prevalence dropped most sharply among biology schools and least sharply among engineering schools; or, equivalently stated, there was more uniform prevalence of EUP among engineering schools than among biology (with business in between). Business courses were often focused on demonstrating computation that the students then implemented rather than having to create new kinds of algorithms or abstractions of their own.

(b) Spreadsheets, Matlab and textual languages made an appearance in all three disciplines. Only in one discipline (business) did one kind of language (spreadsheets) greatly predominate over the others. This language diversity presents an opportunity for EUSE tool designers to aid a broad range of populations with a new tool, but integrating a given tool into a given course may prove challenging if that course uses a different kind of language than what the tool supports.

(c) Software engineering topics related to testing and debugging occurred at a non-trivial rate in close proportion to the mentions of programming languages. Biology and business trailed engineering in terms of the prevalence of software engineering topics relative to programming.

5. Conclusion

Overall, this study highlights the opportunity for EUSE research and education to generate a broad impact on education in a wide range of disciplines. Future work could build on this research in several areas.

First, the sharp drop of EUP prevalence among schools in biology, the somewhat less sharp drop in business, and the lower correlation between EUP and EUSE in these disciplines (compared to engineering), all together raise the question of how to more effectively broaden training in these topics for those disciplines. Future work could investigate whether this challenge is merely a matter of increasing adoption of existing approaches or if, instead, perhaps an opportunity to develop new programming tools and/or teaching strategies tailored to those disciplines.

Second, this preliminary study illustrated that syllabi may serve as a useful resource for understanding education in multiple disciplines. Further studies could focus more deeply on engineering and biology, and/or more broadly on a larger range of software engineering topics such as requirements analysis, architecture and/or design issues, refactoring, documentation, and reuse.

Finally, with PP and SEP scores for each school in hand, it is now possible to assess whether those scores have any relationship to other key variables pertaining to the concerns of different stakeholders. For example, do students who graduate from high-PP or high-SEP schools tend to have higher starting salaries than those from low-PP and low-SEP schools, all else being equal?

Future investigation on syllabi and other sources of data on educational practice could aim to address these and other issues illuminated by the current study, thereby revealing opportunities to more effectively serve students in a broad range of disciplines.

6. References

1. Abraham R, Erwig M. Inferring templates from spreadsheets. International Conference on Software Engineering (ICSE). 2006; 182-191.
2. Ahamed S, *et al*. Computational thinking for the sciences: A three day workshop for high school science teachers. ACM Technical Symposium on Computer Science Education, 2010; 42-46.
3. Bort H, Brylow D. CS4Impact: Measuring computational thinking concepts present in CS4HS participant lesson plans. ACM Technical Symposium on Computer Science Education, 2013; 427-432.
4. Cain A, Babar M Reflections on applying constructive alignment with formative feedback for teaching introductory programming and software architecture. International Conference on Software Engineering (ICSE) Companion, 2016; 336-345.
5. Chowdhury S. Assisting end-user development in browser-based mashup tools. International Conference on Software Engineering (ICSE), 2012; 1625-1627.
6. Dill D, Soo M. Academic quality, league tables, and public policy: A cross-national analysis of university ranking systems. Higher Education, 2005; 49(4):495-533.
7. Fisher M, Cao M, Rothermel G, Cook C, Burnett M. Automated test case generation for spreadsheets. International Conference on Software Engineering (ICSE), 2002; 141-151.
8. Ford M, Forman E. Redefining disciplinary learning in classroom contexts. Review of Research in Education, 2006; 30:1-32.
9. Gioia D, Corley K. Being good versus looking good: Business school rankings and the Circean transformation from substance to image. Academy of Mgmt. Learning & Ed. 2002; 1(1), 107-120.
10. Hannay J, *et al*. How do scientists develop and use scientific software? Workshop on Software Engineering for Computational Science and Engineering, 2009; 1-8.
11. Hermans F, Sedee B, Pinzger M, Deursen A. Data clone detection and visualization in spreadsheets. International Conference on Software Engineering (ICSE), 2013; 292-301.
12. Jones M., Scaffidi C. Obstacles and opportunities with using visual and domain-specific languages in scientific programming. IEEE Intl. Symp. On Visual Lang. and Human-Centric Computation. 2011; 9-16.
13. Ko A, *et al.* The state of the art in end-user software engineering. ACM Computing Surveys (CSUR). 2011; 43(3).
14. Missiroli M, Russo D, Ciancarini P. Learning agile software development in high school: An investigation. International Conference on Software Engineering (ICSE) Companion, 2016 293-302.
15. Nardi B. A Small Matter of Programming: Perspectives on End User Computing, MIT Press, 1993
16. Papaevripidou M, *et al*. 2007 Modeling complex marine ecosystems: An investigation of two teaching approaches with fifth graders. Journal of Computer Assisted Learning. 2007; 23(2):145-157.
17. Rashed G, Ahsan R. Python in computational science: Applications and possibilities. International Journal of Computer Applications, 2012; 46(20):26-30.

18. Scaffidi C, Myers B, Shaw M. Topes: Reusable abstractions for validating data. International Conference on Software Engineering (ICSE), 2008; 1-10.
19. Stolee K, Elbaum S. Refactoring pipe-like mashups for end-user programmers. International Conference on Software Engineering (ICSE). 2011; 81-90.
20. Tjaden B. A multidisciplinary course in computational biology. ACM Journal of Computing Sciences in Colleges. 2007; 22(6):80-87.