A Snapshot of Current Practices in Teaching the Introductory Programming Sequence

Stephen Davies
University of Mary Washington
Department of Computer Science
Fredericksburg, VA 22401
540-654-1317
stephen@umw.edu

Jennifer A. Polack-Wahl
University of Mary Washington
Department of Computer Science
Fredericksburg, VA 22401
540-654-1318
polack@umw.edu

Karen Anewalt
University of Mary Washington
Department of Computer Science
Fredericksburg, VA 22401
540-654-1362
anewalt@umw.edu

ABSTRACT
We present results from a nationwide survey of undergraduate computer science departments regarding languages and techniques taught in CS0, CS1, and CS2. This snapshot of 371 schools provides an intriguing look into the state of computing education today in the U.S., quantifying which practices are actually in common use. Among other things, the study reveals the great variety in CS0 approaches, the relative uniformity of CS1 and CS2 approaches, the dominance of Java as a language for the introductory major sequence, and the tendency for departments to teach CS1 and CS2 in a consistent manner, rather than exposing students to different ideas in each.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms
Languages, Measurement.

Keywords
Survey, Programming Languages, CS0, CS1, CS2.

1. INTRODUCTION
Computing Curricula 2001 provides the most recent comprehensive curricular recommendations for undergraduate programs in Computer Science [10]. This document combined with the 2008 update [9] provides guidelines and suggestions for undergraduate computer science programs and represents joint work done by the two most widely recognized organizations in the computing discipline, the ACM and IEEE. While the recommendations are widely known, much less is known about the state of computing education today. We present results from a nationwide survey of undergraduate computer science departments regarding languages and techniques taught in CS0, CS1, and CS2. We undertook this project in an effort to quantify practices in common use today. Our hope is that through understanding more about current practices the discipline will better be able to contextualize innovations and trends in computer science education.

SIGCSE'11, March 9–12, 2011, Dallas, Texas, USA.

2. RELATED WORK
Several surveys have been done related to various aspects of computer science education.

In 2001, a group of professors from the University of Southern Queensland, Australia conducted a census on all thirty-nine Australian universities about the language trends in introductory programming courses. The census asked approximately 19,900 students taking an introductory programming course at one of the 37 Australian universities which languages were taught in their courses. The most widely taught language was Java, followed by Visual Basic, although, if C and C++ were combined (as they are often taught interchangeably), then the number of students receiving instruction in C/C++ surpassed the number of students receiving instruction in VB [4]. The census further broke down language in terms of discipline. Students indicated a discipline for which their course was designed; the options were: computer science/information technology (CS/IT), engineering, business, and other. Java was the most widely used language in all of the categories. VB was used frequently in business disciplines, and C++ and Haskell were used in CS/IT and engineering disciplines. Students were further asked about the paradigm used in the course. While 81% of the respondents reported using an object-oriented language in their course, most students reported that they did not use an early objects approach in their course. Instructors were asked to identify reasons for selecting a language for their course and 56% indicated industry relevance of the language. A separate study showed that the languages thought to be relevant by the instructors were actually the most in demand languages by industry [5]. The census project was repeated in 2003 for Australian and New Zealand universities. Again the results showed a dominance of Java in programming courses [6].

In spring of 2004, a survey of 351 faculty members was conducted concerning the content in the first course in computing [2,3]. The survey participants were either faculty who had contacted a medium-sized publisher of Computer Science textbooks or were members of the SIGCSE email list. Survey results showed that Java was used twice as often as C++ with members of the SIGCSE list; however, C++ was used more than Java with the publisher list. Scheme, C, and other languages were available answers but the number of people selecting them was small (5, 7, and 19 respectively). 66.9% of the survey respondents reported using the object-oriented paradigm for at least part of the course [2]. Referencing comments received with the survey, the author observed "Those who were using the object-oriented paradigm seemed to be struggling to find a

balance between object oriented concepts and the more traditional constructs that appear in the general programming category [3]."

In 2006 Schulte and Bennedsen conducted a survey concerning what is taught in introductory programming [11]. Because no international list of all instructors who teach programming exists, the online survey was sent to the authors of articles for the 5th Koli conference, 37th SIGCSE, the 11th ITiCSE, two OOPSLA Conferences, the two issues of SIGCSE Bulletin, the members of the Danish Association of Datamatician Educators, the participants of workshops on computer science education in Germany, and recipients of the SIGCSE mailing list. 242 fully completed surveys were received. The results of interest to this research are as follows:

- Object oriented principles were covered by 85% of respondents. Of these respondents, 60% indicated using an objects first approach.
- Java was used by 58.3% of the respondents, followed by C++ 18%, and Pascal 9%. 20% of the respondents indicated that they used a language other than the three receiving the most responses.

Our study adds to these results in part simply by updating them, since the most recently published survey of this kind was four years ago[11]. Also, our survey solicited a broader set of information than most of the works above, covering aspects of the entire CS0-CS1-CS2 sequence. Additionally, we include information about institutional characteristics, so that differences between practices at different types of schools become apparent.

3. PROCEDURES AND METHODS

We sent our online survey to 785 institutions in the USA that have a Computer Science (CS) or Computer Information Systems (CIS) major. The aim was to survey as wide a spectrum of schools as possible with respect to geographical distribution, private versus public institution, and school size, so as to have a measure of the uniformity of languages used within the introductory sequence in computer science. The schools were originally selected from the Carnegie Classification List [1]; every 10th school in alphabetical order that had a CS or CIS program was selected. In order to find every tenth school, we started with the first school in the Carnegie Classification List and did an Internet search on that school to verify that that school had a CS/CIS department. If the school did not have a CS/CIS major the next school in the list was selected and verified that it had a CS/CIS major. After successfully finding a school with a CS/CIS major, we then counted ten schools down and repeated the same process. It became apparent that this procedure was cumbersome and the yield was low since the Carnegie Classification List includes every school within the USA and does not allow the list to be filtered based on major programs offered. Upon further research we found and used the US News University Directory because it listed all the universities in the USA and can generate a list of universities filtered to include only those with a CS/CIS major program [12]. From the list of all universities with a CS/CIS major, we selected every 5th school in alphabetical order. If a selected school was already selected from the Carnegie Classification List then the next school in the search results was selected. The overall result of this procedure is that 785 schools out of the estimated 1350 total institutions with a computing program (see [12]) were solicited. This represents about 58% of the total, and our alphabetical approach prevented any obvious bias towards any particular demographic.

Schools were contacted via email containing a link to an online survey hosted at Survey Monkey (www.surveymonkey.com). The survey contained the following information and questions.

During this survey, please use the following definitions:

- **CS0** – an introductory course with no prerequisites, involving at least some programming, that does not count towards the major
- **CS1** – the first required course in the major programming sequence
- **CS2** – the second required course in the major programming sequence (traditionally called "Data Structures")

1. What is the name of your institution?
2. Does your department regularly offer a CS0 course?
3. What language(s)/environment(s) are covered in your CS0, CS1, CS2 courses? Check all that apply: Alice, C, C++, Greenfoot, Java, Java Applets, JavaScript, Lisp/ Scheme, Python, Ruby, Scratch, Visual Basic, Other
4. Do your students develop programs using a command-line environment, a graphical IDE, or both in CS0, CS1, and CS2? Command-line, Graphical IDE, Both
5. What programming paradigm best describes the programming portion of your CS0, CS1, CS2 courses? Check all that apply: Imperative/procedural, Object-oriented, Functional, Other
6. Do you teach the use of an interactive debugger in at least some sections of any of the following: CS0, CS1, CS2? Yes, No
7. Approximately what percentage of class time do your CS0 courses spend on teaching algorithmic and programming skills? (as opposed to broader, holistic topics about computer science in general?) (If this varies considerably by instructor, semester, or section, try to give a reasonable average.) 0-10%, 10-20%, 30-40%, 40-50%, 50-60%, 70-80%, 80-90%, 90–100%.
8. What degrees do you offer? Associate's CS, Associate's CIS, Associate's SE, Bachelor's CS, Bachelor's CIS, Bachelor's SE, Master's CS, Master's CIS, Master's SE, Ph.D. CS, Ph.D. CIS, Ph.D. SE, Other

4. RESULTS AND ANALYSIS

With one follow-up email request, we received 371 responses from the 785 schools contacted, or 47.2%. Based on the US News University Directory[12] there are 1350 institutions with CS/CIS majors; therefore, this represents 27.5% of these institutions.

For simplicity, we use the most conservative "universal" margin of error in computing the confidence interval for all items, using 0.5 as the sample proportion $\hat{p}$. We apply the finite population correction $\sqrt{(N-n)/(N-1)}$ to the standard margin of error formula $1.96\times\sqrt{\hat{p}\times(1-\hat{p})/n}$, where N is the population size and n is the sample size. In this case, N represents the total number of computer science degree programs in the United States. We estimate this number as 1350 based on [12]. With n=371, then, our (conservative) margin of error for all items on the survey is ± 4.3%.

4.1 CS0

4.1.1 Frequency of offering

The definition of CS0 we provided on the survey was "an introductory course with no prerequisites, involving at least some programming, that does not count towards the major." Only 60.1% of respondents (223 total) indicated that their department regularly offered such a course. This was lower than we expected, and apparently indicates that at over a third of institutions, a student's earliest exposure to programming concepts will be the major sequence.

Note that the Computing Curricula 2001 Task Force endorsed the concept of moving to a three-course introductory sequence in order to make it easier to cover the growing amount of material included in introductory courses and to provide students with more time to assimilate course material[10]. This can be achieved in at least two ways, one of which would be to "front load" the traditional CS1/CS2 sequence with a CS0 course. The other approach would be to "backload" CS1/CS2 with a third course, a strategy which perhaps accounts for the relatively low number of institutions offering CS0.

We hypothesized that larger institutions, with more faculty and resources, would offer CS0 courses more often, but this did not turn out to be the case. The likelihood did depend on graduate degrees offered, however: 75% (±4.3%) of PhD-granting institutions regularly offered CS0, compared with only 57% (±4.3%) of other institutions (by χ^2 this was significant to $\alpha<0.05$.)

Table 1. Number of respondents regularly offering a CS0 course that use certain languages in at least some sections. (Note the percentages do not total to 100% since some schools regularly offer more than one CS0 language.) As with all items on the survey, the margin of error is ± 4.3%.

Language in CS0	# respondents (%)
Ada	1 (0.4%)
Alice	70 (31.4%)
C	13 (5.8%)
C++	17 (7.6%)
C#	5 (2.2%)
Greenfoot	2 (0.9%)
Java	41 (18.4%)
JavaScript	38 (17.0%)
Lisp/Scheme	5 (2.2%)
Python	45 (20.2%)
Ruby	1 (0.4%)
Scratch	3 (1.3%)
Visual Basic	45 (20.2%)
Other	39 (10.5%)

4.1.2 Languages

The programming languages in CS0 vary widely (see Table 1 and Figure 1.) Alice was the most common in our survey (31.4% of respondents), but Python (20.2%), Visual Basic (20.2%), Java (18.4%), and JavaScript (17.0%) appear in almost equal numbers. Alice is by far the most common novice-focused environment in common use: Scratch (1.3%) and Greenfoot (0.9%) are not yet widely adopted. Many other languages – from Lisp to Ruby to C# to ActionScript – were also mentioned, testifying to the great degree of innovation surrounding making programming accessible to non-majors.

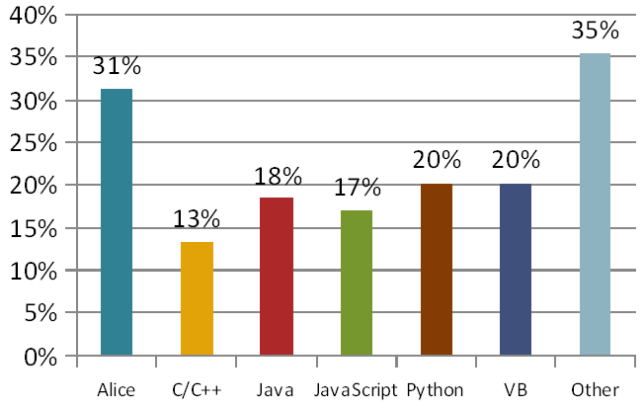

Figure 1. The most popular languages used in CS0 courses. (The "other" bar combines respondents who literally answered "other" with those who chose an item on the survey other than the most popular six.)

4.1.3 Degree of programming content

Also widely varying among CS0 courses is the percentage of time devoted to programming and algorithmic development, as opposed to broader, holistic topics about computer science in general. Respondents were asked to estimate the percentage of overall class time that was spent on programming, in increments of 10% (see Figure 2.) As is apparent, there is no consensus, although the most common strategy is to spend between 80 and 90 percent. Clearly, most educators feel it is important to devote at least some time to breadth-first topics, though the relative emphasis on this is highly variable.

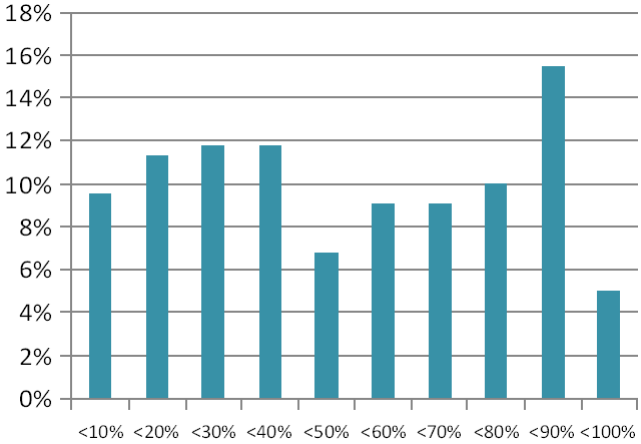

Figure 2. Percentage of CS0 class time spent "teaching algorithmic and programming skills."

4.2 CS1 and CS2

CS1 was defined on our survey as "the first required course in the major programming sequence," and CS2 as "the second required course in the major programming sequence (traditionally called

'Data Structures.')" We concede that there is widespread disagreement among institutions about the precise meanings of these two designations (see, e.g., [8]), and also that many schools spread the programming "core" across three courses. For simplicity and ease of comparison, however, we only surveyed two major courses.

4.2.1 Language choice

Table 2 and Figure 3 depict the relative frequencies of language use reported for CS1 and CS2. There is far more consensus here than in CS0, with Java and C++ being the driving factors (reported by a combined 73% (CS1) and 85% (CS2) of all respondents.)

Table 2. Number of respondents offering CS1 and CS2 in various languages. (Note the percentages do not total to 100% since some schools indicated more than one.) The margin of error is ± 4.3%.

Language	CS1: # (%)	CS2: # (%)
Ada	4 (1.1%)	3 (0.8%)
Alice	16 (4.3%)	2 (0.5%)
C	27 (7.3%)	18 (4.9%)
C++	107 (28.8%)	134 (36.1%)
C#	7 (1.9%)	6 (1.6%)
Greenfoot	2 (0.5%)	0 (0.0%)
Java	179 (48.2%)	207 (55.8%)
JavaScript	12 (3.2%)	10 (2.7%)
Lisp/Scheme	7 (1.9%)	1 (0.3%)
Python	48 (12.9%)	12 (3.2%)
Ruby	2 (0.5%)	1 (0.3%)
Scratch	6 (1.6%)	0 (0.0%)
Visual Basic	36 (9.7%)	27 (7.3%)
Other	39 (10.5%)	29 (7.8%)

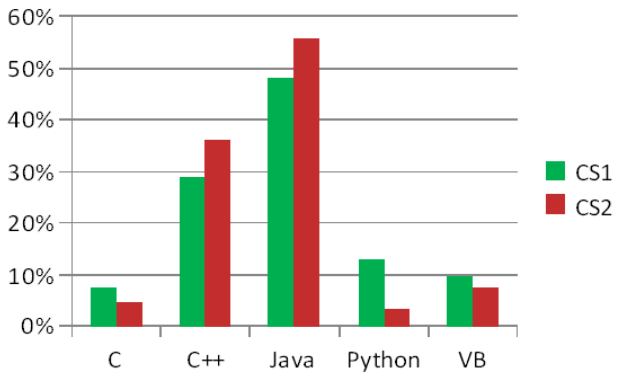

Figure 3. The most popular languages used in CS1 and CS2.

Note that the dominance of Java in our survey responses confirms previous results [2,4,6,11] collected over a span of nearly ten years. Despite what else may be happening in computer science education, the preference of Java as an instructional language is remarkably stable.

It is interesting to examine the correlation between languages used and school ranking. U.S. News and World Report[13] annually ranks institutions within four broad categories – National Universities, Liberal Arts Colleges, Regional Universities, and Regional Colleges – based on the finer-grained Carnegie Basic classification[1]. It then assigns each school to one of four "tiers" in its category, with tier 1 being the highest ranked, and tier 4 the lowest. U.S. News further ranks the schools within the top two tiers more precisely, but for purposes of this comparison we use only the "tier number" for each school.

Java and C++ are of course used broadly across all four tiers. But there is a strong correlation between ranking and certain other languages. Python, for instance, is used far more frequently in the higher-ranked tiers (for CS1, the percentages in each tier are 20.9%, 14.6%, 10.6%, and 1.6%), while Visual Basic is more frequent in tiers 3 and 4 (for CS1: 8.2%, 1.0%, 12.9%, 18.0%.) Both of these results are significant to $\alpha<0.05$. The Visual Basic trend continues in CS2 – tier percentages are 4.5%, 1.0%, 8.2%, and 14.8% – though Python does not, with 87.5% of the Python CS1 schools moving to either Java or C++ for CS2. (See Figures 4 and 5 for a summary of these results.)

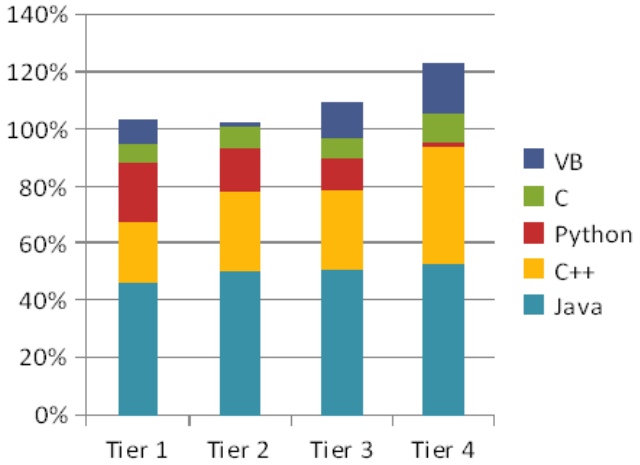

Figure 4. Percentages of schools in each ranking "tier" (as defined by U.S. News) who use each language (in at least some sections) for CS1. (Note the totals do not add to 100% since some schools reported more than one language used.)

Rankings are, of course, not a conclusive indicator of a school's strength, reputation, or anything else. But these trends are worth considering. Perhaps tier 1 and 2 schools choose Python because they feel it is more theoretically sound; perhaps tier 3 and 4 schools choose Visual Basic because they believe it is more vocationally relevant. These are speculations. What is certain is that Java and C++ (in that order) are currently dominant, and that the ways in which institutions differ from the mainstream is somewhat dependent on their type.

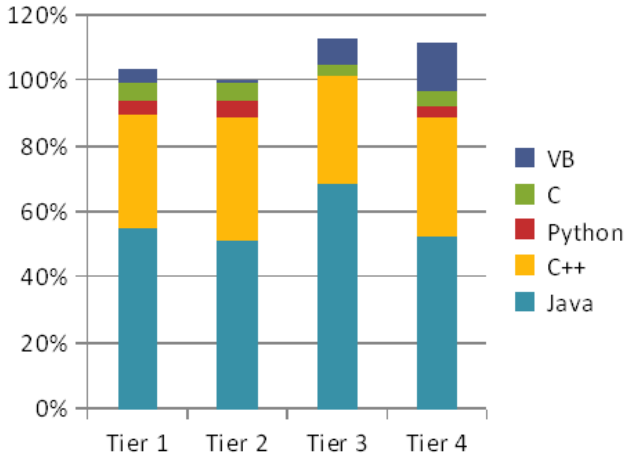

Figure 5. Language use by tier for CS2.

4.2.2 Language continuity

This gives rise to another question: how many schools use a *different* language for CS1 than they do for CS2? There are arguments to be made both for switching languages and for not switching. On the one hand, there may be pedagogical advantages to different languages in the different courses: perhaps Java or Python is seen as easier for novices to learn the basics of programming with, while C++ lends itself to a transparent view of memory usage appropriate to CS2. Too, it is advantageous for students to know more than one language – for diversity of skill set, as well as broader comprehension – and the CS1/2 sequence may be as good a time as any to introduce the second one. (Note that in Computing Curricula 2001, the task force recommends that computer science programs teach students to become competent in languages that make use of at least two programming paradigms[10].) The implications of switching languages received attention a decade ago, for instance in [7].

In any event, the answer to the question is summarized in Table 3. A total of 244 respondents said they use either Java for both courses or C++ for both, and of the "other/other" group 18 used Visual Basic for both. This means that in our survey, at least 70.6% of schools (262/371) teach CS1 and CS2 in the same language.

Table 3. To switch or not to switch? The number of respondents using various combinations of languages in their CS1/CS2 sequence.

CS1	CS2	# respondents (%)
Java	Java	151 (41%)
Java	C++	37 (10%)
Java	other	7 (2%)
C++	Java	26 (7%)
C++	C++	93 (25%)
C++	other	3 (1%)
Other	Java	42 (11%)
Other	C++	18 (5%)
Other	other	30 (8%)

It is worth calling attention to the fact that according to this data, at least half of schools teach only Java and/or Python in the CS1/2 sequence. (We compute this as 184/371=50.0%, since 30 of the "other/Java" responses were for Python/Java and 3 of the "Java/other" responses were for Java/Python.) Note that unlike C and C++, both of these languages use automatic garbage collection and (normally) do not compile natively. While we do not condemn the practice of using them exclusively in the sequence, we note that certain topics traditionally associated with CS2 may be more difficult to illustrate in Python and Java, such as: pointer arithmetic, memory leaks, explicit memory deallocation, and the tradeoffs of passing parameters by value or by reference. Further we note that in Computing Curricula 2008 the topic of "pointers and references" is identified as a core topic in the Programming Fundamentals/Data Structures component [9]. Computing Curricula 2001 also included these topics in the Fundamental Data Structures core area [10]. Both documents also listed "parameterization mechanisms (reference vs. value)" as a core topic for the Programming Languages component. While this topic could be deferred until after the introductory sequence, it seems appropriate to present it at this time.

4.2.3 Other aspects

4.2.3.1 Paradigm

We asked respondents what programming paradigm best described the programming portion of their CS1 and CS2 courses, offering as choices "imperative/procedural," "object-oriented," "functional," and "other." Based on the languages that were specified, we believe there may have been some confusion about the term "functional": it seems many respondents interpreted "functional" to be the same as "imperative/procedural" (presumably because "functions" are employed). Hence, we do not believe the numbers reported for these two categories are meaningful. "Object-oriented," however, resulted in no ambiguity, and these figures are summarized in Table 4.

Table 4. Percentage of respondents who indicated that the "object-oriented programming paradigm" was used.

OO in CS1	OO in CS2	# respondents (%)
Yes	Yes	234 (63%)
Yes	No	9 (2%)
No	Yes	106 (29%)
No	No	22 (6%)

In summary, the object-oriented paradigm is very popular. A strong majority of schools use it for both courses in the intro sequence, with around a third switching from procedural to OO in CS2. Only six percent of schools did not teach OO at all.

4.2.3.2 Tools

Finally, we asked about tools employed: whether students used a command-line environment, an Integrated Development Environment (IDE), or both; and whether they received instruction in using an interactive debugger. The results on the environment question are summarized in Table 5. IDEs clearly carry the day, though a surprising number of schools claim to use both environments within a single course.

Table 5. Types of development environments used.

Environment	CS1	CS2
Command-line	56 (15%)	55 (15%)
Graphical IDE	173 (47%)	149 (40%)
Both	138 (37%)	157 (42%)

Interactive debuggers, too, are popular, but by no means universal. They were reported as being taught in 228 (61.5%) of CS1 courses, and 244 (65.8%) of CS2 courses. Only 70 schools (18.9%) answered this question differently for CS1 and CS2, perhaps indicating that stability is the general rule for tools as well as for languages.

5. CONCLUSIONS

There is a tremendous variety in approaches to CS0. This could be interpreted as a sign of pedagogical innovation, or as a field struggling to identify best practices. Over one-third of schools don't regularly offer a CS0 course at all, and those that do use an enormous variety of languages, and spend a widely differing amount of class time on programming vs. other topics.

CS1 and CS2, by contrast, are far more uniform across schools. Combining several results from the survey, we can say that the most popular approach seems to be an object-oriented pedagogy from start to finish, the Java programming language, and a development environment including IDEs and interactive debuggers. This composite sketch describes perhaps about 2/3rds of schools in the country.

Interestingly, institutions tend to stay with the same approach between CS1 and CS2, rather than introduce a new paradigm, language, or environment. This pedagogical consistency presumably reduces some aspects of cognitive demand, although it means that if students are exposed to variety, it will typically not be in the introductory sequence.

As a final note, we remark that it may be beneficial to the computer science education community to develop a practice of polling institutions about their practices on a regular basis. This would help track trends in what our field is actually implementing in everyday classrooms, and provide a larger context to the noteworthy innovation our leading educators are developing.

6. ACKNOWLEDGMENTS

We are grateful to Debra Hydorn, Megan Martin, and Amy Sams for their help with very time-consuming data processing and analysis.

7. REFERENCES

[1] Carnegie Foundation. 2009. Carnegie Classifications of Institutions of Higher Education. Retrieved December 2, 2009 from http://classifications.carnegiefoundation.org/.

[2] N. Dale. 2005. “Content and emphasis in CS1,” *SIGCSE Bulletin*, vol. 37, pp. 69-73.

[3] N. Dale. 2006. “Most difficult topics in CS1: results of an online survey of educators,” *SIGCSE Bulletin*, vol. 38, pp. 49-53.

[4] M. De Raadt, R. Watson, and M. Toleman. 2002. “Language Trends in Introductory Programming Courses” *Proceedings of Informing Science and IT Education Conference*.

[5] M. De Raadt, R. Watson, and M. Toleman. 2003. “Language Tug-Of-War: Industry Demand and Academic Choice” *Proceedings of the fifth conference on Australasian computing education*-Volume 30.

[6] M. De Raadt, R. Watson, and M. Toleman. 2004. “Introductory programming: what's happening today and will there be any students to teach tomorrow?,” *Proceedings of the sixth conference on Australasian computing education*-Volume 30, p. 282.

[7] A. Dingle and C. Zander. 2000. “Assessing the ripple effect of CS1 language choice,” in *Proceedings of the second annual CCSC on Computing in Small Colleges Northwestern conference*, pp. 85–93.

[8] M. Hertz. 2010. “What do CS1 and CS2 mean?: investigating differences in the early courses,” in *Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE 2010)*, pp. 199–203.

[9] Interim Review Task Force. 2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001. Retrieved December 2, 2009 from http://www.acm.org//education/curricula/ComputerScience2008.pdf.

[10] Joint Task Force on Computing Curricula. 2001. Computing Curricula 2001: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Retrieved December 2, 2009 from http://www.acm.org/education/education/education/curric_vols/cc2001.pdf.

[11] C. Schulte and J. Bennedsen. 2006. “What do teachers teach in introductory programming?,” *Proceedings of the second international workshop on Computing education research*, pp. 17-28.

[12] U.S. News and World Report. 2010. University Directory. Retrieved May 3, 2010 from http://www.usnewsuniversitydirectory.com/.

[13] U.S. News and World Report. 2010. College Rankings. Retrieved June 19, 2010 from http://www.usnews.com/rankings.