College of Engineering
Undergraduate Computing Education Committee (UCEC)
Final Report — Feb 3, 2015

Introduction

The Undergraduate Computing Education Committee was charged to study the computational education provided for College of Engineering students at the University of Illinois at Urbana-Champaign. The committee was asked to answer five specific questions, as follows, which led to the committee making six specific recommendations as detailed below.

Responses to questions

Question 1. *What type of computational education should we require for engineers at Illinois?*
Engineering students at Illinois (except CS and ECE) should take an early introductory computing course (CS 101) that parallels the math and physics courses that are also taken in the first year (see Recommendations 2 and 5). This establishes the early importance of computing and provides a baseline on which to build. Subsequent to this, the required computing education should be primarily within each major, preferably integrated into existing required classes in a "computational thread" within the major so that students take at least one course per year with a substantial computing component (see Recommendation 6).

Question 2. *What core competencies should every student possess?*
We should expect that all engineering graduates from Illinois are competent at using computational tools and methods within the scope of their individual majors. This means that all engineers should be able to understand which tools (e.g., finite element simulations, plotting software) are appropriate for engineering tasks including design and analysis, understand how to use these tools appropriately, including their limitations, and be comfortable and confident in tool selection and application. Additionally, all students should be able to write simple computer programs in a high level language (e.g., Python, MATLAB) and to be able to integrate such programs in the context of solving engineering problems. Finally, all students should be proficient in "computational thinking", meaning that students should be able to conduct major-appropriate abstraction, modeling, representations of information, and algorithmic thinking related to solving engineering problems. See Recommendations 2 and 6 for specific competencies related to particular classes, and Recommendation 4 for evaluation and assessment.

Question 3. *How should we modify or enhance the existing curriculum to provide a better computational education for engineering students?*
The introductory computing course (CS 101) should be revised in line with the core competencies outlined in Question 2 and with the needs of the courses in each major (see Appendix A). Details of the revision of CS 101 are given in Recommendation 2. Each major should have a required 200-level course that has CS 101 as a prerequisite and that includes

a substantial computing component. This 200-level course should serve as the first in a "computational thread" running through the major, as outlined in Recommendation 6. In addition to the required courses (CS 101, followed by a major-specific "thread"), there should also be optional CS courses designed to expose students to computing topics without the need for a major or minor in CS. Details and suggestions for possible CS courses are given in Recommendation 3. Coordination between these different components in CS and within each major should be provided by a liaison committee (see Recommendation 1), and there should be integrated program evaluation (Recommendation 4).

Question 4. *Are there ways that computational components can be woven across the curriculum?*
Most computing education for engineers at Illinois should occur within existing classes in each major as part of a "computational thread" (see Recommendation 6). This provides authentic pedagogy, motivation, and assessment for the use of computing skills within an engineering context. To provide a baseline computing education for all majors, the introductory CS 101 course should be retained (Recommendations 2 and 5) but should be established as a prerequisite for the first course in the "thread" in each major. CS 101 and the major "threads" should be closely coordinated (see Recommendations 1 and 4).

Question 5. *What equipment, staff and facilities are needed to carry out any proposed changes?*
Current equipment and facilities are sufficient for the proposed changes, although this should be routinely re-evaluated by a standing committee (Recommendation 1). Staff will be needed within CS for the changes to CS 101 (Recommendation 2) and potential new CS courses (Recommendation 3). It is anticipated that existing CS staff will be sufficient for the CS 101 reform, but additional personnel will be required to create new courses. Within each department necessary course changes (Recommendation 6) can be enacted by existing faculty and staff. Extra college-level support staff within CARE or CSE may be necessary for additional student support (Recommendation 6.2). Evaluation plans (Recommendation 4) are much more likely to be implemented thoroughly with additional staff resources, and it is recommended that this be provided at the college level, perhaps via AE3 or CSE.

Recommendation 1: Reactivate CS liaison standing committee

1.1. Reactivate the Computer Science liaison standing committee to oversee implementation of recommendations in this report and continue to evaluate and monitor the computing education of undergraduate engineering programs.

- 1.1.1. Appoint a chair for the CS liaison committee.
- 1.1.2. Ensure committee members are the key implementation people within each department, with a department mandate to coordinate changes within departments.
- 1.1.3. Charge the committee to meet regularly to oversee the implementation of the new CS 101 format and the development of computational material in subsequent courses within majors.

Recommendation 2: Reform introductory computing (CS 101)

2.1. The introductory computing course for engineering undergraduates (except CS and ECE) should remain as CS 101.

2.2. The objectives of CS 101 are to enable students to:

- 2.2.1. solve problems algorithmically,
- 2.2.2. be confident about using computation as a standard tool, on the same level as math and physics,
- 2.2.3. program in a high-level language (e.g., Python or MATLAB),
- 2.2.4. access data sources, process data, and create outputs including plots,
- 2.2.5. use MATLAB proficiently.

2.3. CS 101 should be normally taken in the freshman year.

2.4. Each engineering major should have one or more required courses that have CS 101 as a pre-requisite. These courses should be taken immediately following CS 101 and should include substantial use of computing, for example by teaching computational modeling of engineering systems. To enable this, the CS department must be able to ensure that all engineering freshman can be accommodated in CS 101.

2.5. There is only one version of CS 101, but per-major specialization should be enabled to a limited extent, for example by having a final project for which students can choose a topic based on their major.

2.6. A proficiency exam for CS 101 should be established that all students must take if they have not taken CS 101 at UIUC. This should ensure basic competence in MATLAB, as well as programming. A CSE introductory MATLAB workshop may be sufficient for AP CS students to reach the required level of competence in MATLAB.

Recommendation 3: Optional additional computing courses

3.1. There is a need for optional courses following on from CS 101 that cover applied data structures, data processing and analysis (including big data), graphics and visualization, compiled languages, low-level hardware programming, object-oriented programming, numerical analysis, user interfaces and user experience, mobile apps, and web development.

3.2. One possibility for covering many of the topics listed above is to group them into three new CS courses:

- 3.2.1. **CS 205 (high level data):** applied data structures and data analysis and visualization (including big data and machine learning) in Python (or possibly R), with a focus on application domains.
- 3.2.2. **CS 206 (low level control):** low-level programming in C/C++ for hardware interaction on Arduino, Raspberry Pi, and similar platforms, emphasizing machine-level understanding for applied systems. This could potentially be an ECE or joint CS/ECE course.
- 3.2.3. **CS 207 (human interaction):** web and mobile programming, including HTML/CSS, JavaScript, and mobile apps with some combination of HTML5, Java for Android, or

ObjC/Swift for iOS, focusing on user interfaces, user experience, communication, and visualization.

3.3. Existing upper-division courses in CS, ECE, and CSE (including cross-lists) are currently sufficient to provide a thorough computational education for students who wish to take them. The CS liaison committee should monitor the appropriateness and implementation of these courses.

Recommendation 4: Evaluate computing education

4.1. The CS liaison committee should be charged to periodically evaluate the computing expertise of undergraduate engineering students, especially during the development of the new CS 101 and subsequent courses.

4.2. One potential strategy for evaluation would be to develop a set of standardized items that can be integrated into a small number of immediately-post-CS-101 courses (e.g., TAM 210/211, MSE 206, etc) and collected each semester, both before and after new course content is developed. These items should include:

4.2.1. Survey questions to measure student familiarity, confidence, and desire for computational tools. For example, "I feel comfortable using a computer to plot data". See Appendix D for example baseline data. These questions could be integrated into existing course feedback surveys.

4.2.2. Assessment items to measure particular student skills, such as plotting data, loading and analyzing data from online sources, etc. These could be integrated into course assessments such as homeworks, lab exercises, project-based learning, etc.

Recommendation 5: Alternative "proficient" pathways

1. Students who are already proficient in programming should be offered alternative pathways for computing education, without having to take the full CS 101 introductory course, but while ensuring that they meet the objectives listed in Section 2.2 (especially 2.2.5: MATLAB proficiency). These students include:

   1.1. AP proficiency (typically in Java or some other language).

   1.2. CS 101 transfer credit from another college.

   1.3. Proficiency exam at UIUC.

2. The CS Undergraduate Office should work to remove any equivalencies for CS 101 transfer credits which do not ensure sufficient MATLAB proficiency.

3. A 1-credit hour course should be provided by CS (possibly cross-listed with CSE or ENG) to provide MATLAB proficiency. This could either be a separate course to CS 101, or CS 101 could be split into 101A and 101B, or similar mechanisms. CS should coordinate with other departments and the college to ensure that post-CS-101 course prerequisites mandate either the 1-credit hour MATLAB course or CS-101.

4. A proficiency exam should continue to be provided for CS 101, and should ensure that students passing the proficiency exams meet all the requirements in Section 2.2.

Recommendation 6: Ensure integrated computing education within engineering majors

6.1. Engineering majors should ensure that all undergraduate develop competence in using computational tools and methods within the scope of the major.

6.2. Integration of computing education in non-computing engineering courses needs to be facilitated by additional support, including assistance with: grading and assessment logistics, lab logistics, development of effective computing pedagogy, and extra support for students at the start of semester (e.g., CARE or CSE). This should be coordinated by the CS liaison committee as these course changes are implemented.

6.3. One way for departments to ensure a comprehensive undergraduate computing education is:

6.3.1. Have a well-established computing "thread" in the curriculum, starting with a sophomore-level course that includes authentic use of computation and has CS 101 as a prerequisite.

6.3.2. Have a small set of courses in a prerequisite chain that follow the sophomore course and all use computation as an integral part of the learning experience.

6.3.3. Coordinate with the CS liaison committee to implement regular evaluation of computing education, both for internal and ABET purposes.

Dept	First course with CS 101 prereq	Computational course chain
MechSE	TAM 210/211	TAM 212, TAM 251, ME 340, ME 370
MatSE	MSE 206 (MSE 201 has co-req)	MSE 304, MSE 401, MSE 406, MSE 498
BioE	BIOE 205	?
AE	AE 202	AE 370, AE 410, AE 420
NPRE	NPRE 247	NPRE 455, NPRE 448

Appendix A: Survey of computing needs in UG programs

A survey was conducted in all Engineering departments (except ECE and CS) about their current and desired needs for introductory computing education (see the table below). All departments currently require CS 101 for their majors, but only some have courses that list CS 101 as an explicit prerequisite and most of these are upper-division courses. A variety of computing languages and platforms are currently in use within courses, with the most common being MATLAB. Some departments are interested in standardizing on a single computing platform, particularly for 200-level courses, with MATLAB being the most common choice. Almost all departments want to establish CS 101 as a prerequisite for some of their 200-level courses.

Dept	CS 101 req. for major?	Courses requiring CS 101	Current platform needs in major courses	Want to standardize on platform?	Want CS 101 as 2XX prereq?

AE	No	None	MATLAB is used for the primary junior-level required courses. AE 483 has elements of C/C++, and some Ada.	Yes, standardize on MATLAB.	Yes
ABE	Yes	None	MATLAB C-like lanaguages R GAMS Python LabView Excel	Probably not, although the topic has come up with regards to lower level (2XX) required courses.	Yes
BioE	Yes	BIOE 205 BIOE 302	BIOE 205: MATLAB BIOE 302: MATLAB BIOE 310: R	Yes, standardize on MATLAB.	Yes
CEE	Yes	CEE 320	MATLAB - 12 courses EXCEL - 10 courses C/C++/C# - 2 courses FORTRAN - 2 courses AMPL/GAMS scripts - 1 course	Yes, there was consensus on standardizing on one platform (e.g., MATLAB).	Yes
ChBE	Yes	CHBE 221 CHBE 440	CHBE 421: Comsol CHBE 424: Polymath CHBE 431: Chemcad CHBE 440: Simulink (MATLAB)	No. Right now each professor chooses their own computational tools.	Yes
ISE	Yes	GE 310 GE 320 GE 413	MATLAB in GE320, GE420, and GE424, GE413, GE310, GE320, IE310, IE360 C/C++ IE420 R: IE300 Java: IE420 Excel: IE310, IE360 Should have competence in OOP	MATLAB and Python	Yes
MatSE	Yes	None	MATLAB in MSE 307/308 (junior labs). MSE 498 uses shell and MATLAB. In the future, probably MATLAB around the computation in sophomore and junior classes.	Not yet.	Yes

MechSE	Yes	None	ME 170: Creo and aPriori ME 320: MATLAB or Mathematica ME 340: MATLAB ME 350: Excel and Creo or Solidworks ME 360: MATLAB ME 370: MATLAB and Working Model ME 471: MATLAB	Some discussion, but no firm conclusion. MATLAB would be the leading contender.	Yes
NPRE	Yes	None	MATLAB (or C, C++) in NPRE 100, 247, 448, 451, 455	Not really. We want both MATLAB and at least one language.	Yes
Physics	Yes	None	MATLAB, C, C++, Mathematica, LabVIEW, Python.	No. Want students to know something about programming, but don't particularly care what language.	No

Appendix B: Survey of peer introductory CS courses

Data collected from publically available information by Wade Fagen.

University / Course	Target Audience	Languages / Technologies	Student Experience
Harvard CS 50	All majors (CS included)	Introduction: Scratch Major focus: C Minor focuses: PHP, JavaScript, HTML	7 MPs Weekly labs Multi-week final project
Illinois CS 105	Non-engineering	Introduction: Scratch Major focus: JavaScript Minor focuses: HTML, Excel	8 MPs Weekly labs Multi-week final project
MIT 6.01	Engineering majors (CS included)	Programming: Python Multi-Week Topics: - Circuit Theory - Mathematical representation of programs	8 MPs Weekly labs Multi-week final project

Berkeley CS 10	Non-CS-majors (engineering included)	Introduction: Snap! Major focus: Python Also includes weekly readings and discussions on ethical/social issues surrounding computing.	3 MPs Weekly labs Multi-week final project
CMU 15-104	Non-CS-majors (engineering optional)	Introduction: Processing Major focus: Python Minor focuses: JavaScript, HTML	7 MPs Weekly labs Multi-week final project

Appendix C: Example desired tasks for post-CS-101

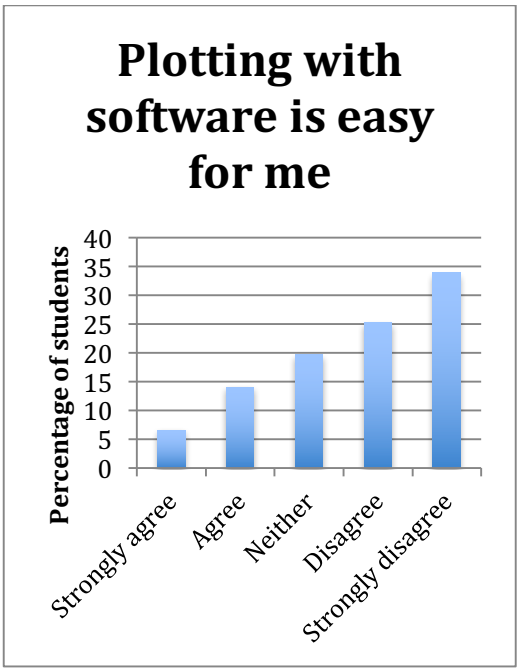
As a guide to the reform of CS 101, follow-on courses from other departments have contributed example assessment items that require the use of computation. These are intended to guide the development of the course content for CS 101.

Course	Assessment item	Computing topics
TAM 212	Worksheet 8	MATLAB or Excel, tabular computation, plotting
AE 199	C Example	C
AE 199	MATLAB Example	MATLAB
BIOE 302	Homework 2	MATLAB, Simulink, plotting

Appendix D: Baseline evaluation information

D-1. TAM 212, Fall 2014, mid-semester evaluation: 59% of students disagreed or strongly disagreed with the statement “Plotting with software is easy for me”. Just 21% agreed or strongly agreed with this statement.

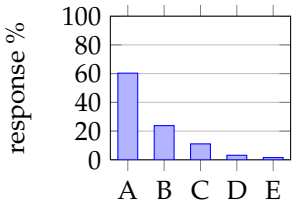
Response	N	Fraction
Strongly agree	11	6.7%
Agree	23	14.2%
Neither agree nor disagree	32	19.8%
Disagree	41	25.3%
Strongly disagree	55	34.0%
Total	162	



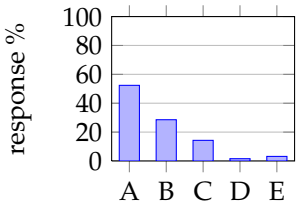
D-2. MSE 406, Fall 2014, end-of-semester evaluation: 84% of students were comfortable with using a computer to plot data, with only 5% being uncomfortable. This is after a semester with a strong inclusion of computational material throughout the course. This course is primarily taken by Juniors in MatSE.

I feel comfortable using a computer to plot data
Average: 1.619 ± 0.916
A: 60.32%
B: 23.81%
C: 11.11%
D: 3.17%
E: 1.59%

Strongly Agree - A B C D E - Strongly Disagree



I think computational materials science skills are important for my post-graduation career.
Average: 1.746 ± 0.975
A: 52.38%
B: 28.57%
C: 14.29%
D: 1.59%
E: 3.17%



I would like to use computation in my MatSE classes. . .
Average: 2.175 ± 1.032
A: 30.16%
B: 34.92%
C: 25.40%
D: 6.35%
E: 3.17%

Much More - A B C D E - Much Less