SIGCSE 2018

Notes by James H. Davenport[1]

21–24 February 2018

[1]Extremely partial, given that at times the conference was 17-way parallel.

Contents

Part I

21 February 2018

Chapter 1

Cybersecurity Modules: Security Injections

http://cis1.towson.edu/~cyber4all/index.php/security-injections_home/

1.1 Introductions (5 min)

1.2 Importance of Secure Coding (15 min)

It's not (just) about firewalls. Short video clip from Obama.

Video talk from Bishop. We can't ignore the Internet (everything is being connected to it) and "feature first, security later" doesn't work. Buffer overflows are common, but trivially preventable.

At Towson, we are now a specialist school, but started with a couple of courses, then a track (currently about 12% of cohort). But only teaching secure coding to the track is a mistake. We want to teach secure coding from the start. Note that ACM accreditation requires security throughout (as Information Assurance) throughout. There are guidelines in ABETT CS accreditation (still in draft, but applied from this year).

All experts agree the importance of "create a security mindset".

We have a Learning Goal in CS0 of "Introduce the risks and mitigation methods associates with integer overflow for introductory CS students". Cartoon of counting sheep, except it goes 32767, -32768 ....

In general science labs are better structured and integrated (into the wider learning outcomes) than CS labs.

1. Facebook group "If the group reaches 4,294,967,296 it might cause an integer overflow".

2. Comair was grounded as #crew changes exceeded 32767 due to bad weather changes.

Claims that our labs take very little instructor effort (JHD would probably agree: a mix of programming and multiple choice).

1.3 Importance in the CS 2013 Curriculum and NSA CAE Accreditation (10 min)

1.4 Modules (45 min)

JHD tried CS0 C++ module from their site, and it got his attention. Some-one asked "how does integer overflow cause a real-world problem", to which JHD quoted "Gangnam Style/YouTube": `https://techcrunch.com/2014/12/03/gangnam-style-has-been-viewed-so-many-times-it-broke-youtubes-code/`.

There are modules at various levels, including "computer literacy" courses (phishing etc.).

1.5 Build-and-Share: Strategies for Including Secure Coding Outcomes (15 min)

Also cites SEEDLab at Syracuse.

1.6 Discussion/Brainstorming session (10 min)

Discussion on the difficulty of involving girls.

1.7 Splash Program

Programme for schools, especially girls.

1.8 Miscellaneous

One of their recommendations is that one should write (in C-speak)

```
int a;
int b;
int c;
```

rather than

```
int a,b,c;
```

This was apparently to guard against the naïve thinking that

```
int a,b,c=0;
```

initialised all three.

Part II

22 February 2018

Chapter 2

Opening

2.1 Introduction

One of the biggest SIGCSEs ever. "Best Paper" are now noted in ACM DL. Also some "best in category" judged by the audience.

Note that the App (Whova) is the only vehicle for communicating session changes etc.: no posted announcements.

Graph showing SIGCSE attendance: flat at 1200 until the current team, then 1450 and 1661 in 2017 and 2018.

Special slide on etiquette for sharing power points.

2.1.1 SIGCSE

SIGCSE chair spoke. 50th anniversary of first CS Educational conference. Notable that SIGCSE Board is 7 women + 1 man.

"Teaching CS is no more about creating more software developers than teaching English is about creating more novelists".

2.2 The Evolution before the Revolution: Brenda Wilkinson

CEO of AnitaB.org, and apparently a pioneer in Chicago public schools[1] for CS4All. Spent 8 years (?as/?working for) Rector for CS Education. Before started, it was only offered in highly selective High Schools. Launched K-12 in December 2013. 460 new teachers have received PD. CS coursework available in 107+ schools. Adding 50–60 schools/year, focusing on High Schools first.

In 2016 Chicago added a CS graduation requirement, for all 400,000 students, beginning with the class of '20. Federal grant to outfit every classroom with high-speed broadband +WiFi.

[1] Third largest system in the country.

Note NYC 2015. Training 4775 teachers to bring CS to everyone. Then the Obama initiative.

She got into teaching after being passed over at work (tech. firm) for a less-qualified man.

We imagine a society where those who build technology mirror the society they build it for.

Claims that the presence (at least of 5K such) women at Bletchley Park was not (widely) known until Sue Black. Claims that Grace Hopper invented the concept of a compiler. Also Gladys West, pioneer of GPS, was one of only two black women hired (?at Aberdeen Proving Grounds).

"In education, the hardest people to reach are the grown-ups". We're teaching critical skills:

- Computational thinking: literacy for the new millennium
- Problem-solving, ethics and collaboration
- The difference between consumption and creation.

One of the greatest problems in AI/ML today is bias: existing datasets are biased: "grandma" produces white women with grey hair. Hence "Facial Recognition is Accurate, If You're a White Guy".

Q Missouri "core 42" new curriculum refused to allow any CS input. What next?

A This involves (constructive) conflict with other STEM subjects.

Chapter 3

Pedagogy 1

3.1 TMOSS: Using Intermediate Assignment Work to Understand Excessive Collaboration in Large Classes: Lisa Yan

See [YMSP18].

Claims this finds excessive collaboration *during* an assignment. Typically CS1 has 400 students, and 1:1 isn't feasible. The troubled students find an online resource and submit that. "Excessive collaboration" is overly relying on an outside resource (peer, past student, online) to complete assignments. We know it's bad for them, but we can't really address it, since one file isn't enough information. TMOSS looks at the GitHub history.

Temporal Measure of Software Similarity. But this needs the fine-grained temporal data. We have developed an Eclipse plugin. 9-day assignment. Mean start day 5.22, 253 snapshots, 9.77 hours on task, and 4 hours TA.

Recall the gold standard is MOSS, developed in 1994. Similarity scores based on MOSS tokens. It's $O(N^2)$, but for MOSS this is OK, 2 minutes. TMOSS then has a problem, 86 days. So compare a snapshot with all final solutions (others and online). $O(N^2M)$ about 8 hours. MOSS found 35 students, TMOSS 61.

Shows distributions of normal and HEC students, where MOSS has a large false negative score.

$Pr(Y > k) = 1-e^{-e^{\cdots}}$: Gumbel distributions. Good data on non-HEC/HEC students for exam scores, start dates etc. TMOSS produces better data. `github.com/yanlisa/tmoss`.

Q Other IDEs?

A Sure: the only use of the IDE is forcing commits.

Q When do you run?

A End of the assignment. We have thought about online use rather than batch use.

Q Student reaction?

A Generally positive.

Q Plagiarism policy?

A We haven't been doing formal reporting.

3.2 Social Help-seeking strategies in a Programming MOOC: Arto Hellas

Author is also Arto Vihavainen! See his 2015 paper `http://dx.doi.org/10.1145/2724660.2724671` in L@S 2015. We surveyed MOOC participants, and asked about help-seeking strategies. What might higher tendency to seek help from friends by linked to? Used a Finnish MOOC, running since 2012, which is also an entrance exam to University of Helsinki. Used the spring 2015 instantiation: surveys to 982 students who completed at least one week (out of 14). 255 usable responses.

1. "Did you ask, and how helpful". Course platform: 38%, 2.7/5; 25% help from friends (2.6/5), 5% help from teachers [] (1.3/5); [1=useless; 5 very helpful]. Split friends into computer-mediated or not, and MOOC platorm into heavy/light.

2. Asked deep/surface/strategic learning. Surface learning is highest in the non-computer-mediated friends group, but that's about the only correlation.

3. Correlation with completeness. "Did not ask for help" was under 50%, then "friends f2f" 52%, then MOOC some (60%), MOOC heavy (80%) and best (90%) was friends computer-mediated. Is that prior bias?

Propose a "MOOC Buddy system" See [McCartney et al SIGCSE Bulletin 38(2007) pp. 156-160].

We didn't ask about frequency of seeking for help, nor why. Also didn't ask "why taking the MOOC".

Q I've been looking at buddies. I think you need one with the same motivation.

A True, but also one with the same deadline motivation.

3.3 Lightweight Strategies for Large Classes: Mia Minnes

Classes have average enrolment of "several hundreds". Shows growth from CRA enrolment survey. In advanced data structures, have "masterclasses" with assigned tutors, who sat with then in groups at lectures etc. Survey using "Class-

room Community Scale", evaluations + grades. Great "feel good", but trivial difference in grades.

So can't justify the cost. Baseline is peer instruction + in-class groupwork. 40:1 to 55:1 student/TA ratio.

Tried various seating strategies in large lectures[1], partly to allow tutor access. Also worked on improved connection: "personal tutors", coming to classes[2] (worked when paired with seating assignments). All variants showed better community scores than the control group. 63% of students didn't see a personal tutor, but those that did reported extremely helpful.

Q Low attendance? Data was in 40–70% range.

A We have a participation score, which can be by attendance or online review. All lectures are podcast. We have many commuters/working students. This is probably why individual seating doesn't work.

Q How large were the zones?

A Depended on layout of lecture theatre. Typically 4 zones, so 50 students with 20–30 present.

Q TA effort? Graduate TAs were 10 or 20 hours/weeks.

A Built-in, irrespective of scheme.

Q Drop rate?

A No real change, but it's low for structural reasons?

Q What do TAs do in the zones?

A Speak up on behalf of puzzled students. Encourage the active learning collaboration.

Q Plagiarism?

A Didn't study this.

Q Group?

A Pair programming certainly also group work in theory classes.

3.4 Eric Roberts: Prize Acceptance Speech

Theme was time travel.

1957 Sputnik changes American education (my generation). "The days when USA and USSR competed in science, rather than election fixing".

[1] Totally assigned seating had a backlash, but some freedom was welcomed.
[2] Bonus: tutor understanding improved.

1964 My first work, on a 402 tabulator, then 1966 has a 1401. 1968 IBM 1620. 1971 Arpanet (first ever network programming course, from BBN: "64 is a reasonable approximation to infinity, but we'll double it".

Now Demand for courses is outstripping supply, Faculty are hard to find. Majors have quadrupled, tenure line flat, teaching faculty a small increase. 2015 we have 1780 PhD awarded. 18% go into academia, which makes one every five years even if distributed evenly.

In 2007, none of the top 5 companies were IT, now all are. Also all US. Data about graduate production. Note that there has been boom/bust before.

In 1980, as a committed feminist, I started the CS department at Wellesley. Showed the 1984 Superbowl ad for McIntosh: '1984 will not be like "1984"'.

At Stanford CS is now the top major for women (over 300 declared).

Q Alternative models?

A Need to be careful. Note that there isn't a "STEM shortage", rather a "Tech shortage". Do we want Amazon and Google to teach people. One reason we look at degrees is that these data are accessible.

Q Alternative models?

A I normally say "If we had a functional government". How about "free tuition in exchange for teaching"? Education fails both the employers and the students — the only people happy are the educators. See McKinsey report.

Chapter 4

Cybersecurity 2

4.1 Cyber Range and Defence Exercises: Burská

Feedback currently is a leaderboard: attacks, injects, total points.

Typical exercise has Blue team (learners), Red team (attackers). White team (JHD would regard as umpires). Green team (maintain the exercise infrastructure). Share of exercise:

1. Blue team familiarisation: 3 hours on day 1.
2. Actual exercise: 6 hours on day 2.
3. debrief: 5 minutes on day 2.

* New features below here

4. scoring timeline interaction: 10 minutes
5. scoring timeline survey (blue team feedback): 5 minutes

Q Size of effort?

A 10 staff in each of red/while, and 5 in green. This is labour intensive, but worth it.

4.2 Teaching CyberSecurity Using Competitive Software Obfuscation and Reverse Engineering Activities: Rizwan

I became course coordinator in 2016. There was a project followed by a challenge phase. The presentation after the challenge phase was the end, and really stimulated the students.

Therefore decided to disseminate: got permission to run questionnaires in 2017.

4.3 Enhancing Security Education Through Designing SDN Security Labs in CloudLab: Park

[PHYL18] Aim: to introduce new research materials into our lecture. Use software-defined networking. SDN separates control planes from data planes. The Network Operating System runs various applications. Uses OpenFlow: "if header = X, send to port 4" .... This means that the network is programmable (as opposed to just configurable). This introduces many new security issues.

CloudLab is an open cloud platform built for research and education purposes. Utah, Wisconsin and South Carolina. This uses profiles as a specification of the experiment's requirements.

This lab should incorporate the latest results in SDN security. This should encourage students in SDN security research.

DOS Avant-Guard (CCS'13), Rosemary (CCS'14)

MitM Aegis (ACM SACMAT'16)

API Misuse

We have five security labs. Each has a problem statement, presents the objectives and outcomes.

Example 1 (Lab 2) *Flooding attacks in the SDN Data Plane. Simple topology: two hosts, switch and controller.*

Student survey. Positive feedback on all questions except usability (first experience of CloudLab).

Q Can I build my own?

A Yes, you can modify my scripts?

Q Are there data retained for forensics?

A You will need to insert tcpdump or wireshark to do this, but yes.

Q How did you measure effectiveness?

A By the survey. But it's a new course and new topic.

Q Issues with CloudLab resource allocations?

A 90% was OK.

Chapter 5

Autograders

5.1 Providing Meaningful Feedback for Autograding of Programming Assignments: Georgiana Haldeman

[HTBV+18] Rutgers: interest in CS majors has doubled in last five years. We like many use Autograders. Typically grade is based on instructors tests. Binary immediate feedback leads to bad student behaviour: trial/error submissions etc. [Ambroseetal2010a] suggests targeted feedback. [Beauboeuf2005] case for good feedback. Instructors want to know what the students are struggling with.

CSF^2:Concepts and Skills Based Feedback Framework. Extract signature /error/hint triples. Classify the submissions by feedback bit strings. Manually inspect the buckets. If necessary, add tests to disambiguate frameworks. Generated in one semester for use later on.

Our example is a conditional fee structure. Various errors, e.g. only outputting in some branches, or wrong answers. Example of feedback given in last case. Got 91.5% ad 87.5% diagnoses correct (two courseworks). Note that this relies on reuse of assignments.

Conclusions: meaningful feedback is important.

Q Did you analyse whether the hints helped?

A Comments from students on the hints were collected.

Q Is the word problem [understanding the spec.] the real challenge?

A Sometimes.

Q Wrong hints?

A Partly because of unit tests only: no static analysis. We don't have a really good analysis.

5.2 ArTEMiS: An Automatic Assessment Management System for Interactive Learning: Krusche

[KS18] We focus on interactive learning: this talk too!

Note this is SE, not programming. We want to reduce correcting effort (classes of 1500). Also fairness issues. Want people to use developer tools daily. Not just active learning, but interactive learning. Use a CI server (Travis) to run test cases automatically. Students run the ArTEMIS client in their browser, as well as the version control client. Artemis server tool, VCS, CI all connect to user management system.

Note that the tests mustn't be accessible to the students, so we need two repositories, one for the students and these then get pushed to instructor-level repository.

But do students need to understand VCS: not necessarily, we have an online editor. We also have TAs roaming the classroom.

There are also classes where we expect students to write test cases, where we write meta test cases.

Scales to over 200 submissions/minute (with enough build agents). Our systems is independent of language (provided it's command-line driven). One problem is trial/error, so we want to limit the number of attempts. And should we publish (maybe some) test cases? What level of feedback is right?

Q Like the architecture. What CI agent do use.

A We use bitbucket and its agents, but there are aspect interfaces, so in theory could be swapped.

Q

A

Q

A

5.3 MRS: Automated Assessment of Interactive Classroom Exercises: Deb

[DFEG18]. Our motivation is algorithms classes. Here there are often pen/paper diagrams to illustrate. How does the instructor mark these?

We deliver questions via mobile app., returns to server and immediately autograded. Instructor gets instant visibility. These are configurable by the instructor, via an XML file. Partial grading is possible: comparing each screen with desired results. Demonstration of selection sort app, with the student basically doing the selection.

Analyse 97 students in sophomore "Intro Hardware" and Junior "analysis of algorithms". Course grade improved 10%, with F-test $p \approx 0.004$. Student surveys were positive, fro both learning and engagement.

5.4 BoF: CyberSecurity

Intro: security education has been around for a while, but massive changes recently in the political landscape. We've been looking at security contextualised in CS (or IS or ...).

1. Most of us have done of this to some extent.
2. Actually produce a security specialist degree. If so, what are curricula, standards, accreditation etc.

NSA accredits 100+ in Cyber Defence, and 19 in Cyber Operations.

CSEC 2017 just released, as part of current generation of curriculum recommendations.

USNA We actually have cyberoffence.

?? Have a DHS degree

Houston IT Department. Various tracks. 100 on this minor. So would like a major.

College in Portland Want a minor

UNC Charlotte What do people teach at u/g; routes to Masters?

Small school Teach it every other year.

Evergreen Teach it every other year. I'd like

Georgia State Just introduced a certificate for the undergraduate programme, and a new Masters Programme. Also want security in the required classes.

near Philly Try to but some in every course. Start with CyberSecurity news at the start of the semester.

FIU Started a graduate programme two years ago. Fastest-growing degree: 200 students from 68 countries. More students from non-STEM than STEM. Many IRS and law enforcement students. Curriculum based on employer demands. We've set up online classes to deal with students who get job offers before they graduate — "can you start now".

Masaryk Bachelors' with a focus, and a full security Masters, splitting into theory and operation/legal.

? I think option 1 is more important, but in fact industry is looking for the label. Hence we've created a major, but I'm looking to hire real quick.

***** Echoes:

? We don't have one

College in Nevada Now incorporate Cisco in network specialist degree. Pressure from industry and Governor.

Pennsylvania College Want a track.

Uni. Central Missouri Bachelors in 2015.

FIU?? Been working on setting up. Advice: "connect to the feds — just got a bunch of equipment from the Army". Setting up 1500VMs (aim to offer to schools etc. as well). Why online classes? See other FIU speaker.

5.5 BoF: Plagiarism

Why do students cheat? Various answers round the room.

1. Focused on grades
2. Jobs
3. because they can
4. it's the norm
5. they feel they haven't got the background
6. Anything on the Internet is fair game.
7. CS is the only department that checks.
8. Pride stops them coming to the help room.
9. Tutors (common problem)
10. Run proctored midterms where students can search but not post, very similar to the homeworks. They know this means that being able to do the coursework matters.
11. I have more confidence in exams, hence a rule that you need a C- on the exam to get a C- or better. Hence cheating can't help.
12. Students can't appreciate that changing variable names doesn't count.
13. Peer review of an assignment post-hoc showed students that there were multiple ways to solve it, which helped with "how do you know".

14. I get very little genuine remorse, even after quoting the Volkswagen case at them.

15. A "regret" system, that they can own up within 72 hours, has brought some up. This may have helped.

16. If you can't detect it scalably, there are problems.

17. Teaching programming is like teaching composition, and why don't they have this problem? Not in classes of 500.

18. There was a debate about the utility of "at the job you'll have to . . . "

19. One teacher operated "If it's posted online, you'll all get zero, as in the corporate world, teams live and die together". This seems to have helped.

Part III

23 February 2018

Chapter 6

What's the big idea with CS Education in K-12 — Tim Bell

This SIGCSE award went to Tim Bell (Auckland), among other things the author of CSunplugged.

6.1 Introduction

Both externally, and internally, we need to know what the big idea is in CS Education in schools.

He did an audience poll (audience was c. 400) and three were in elementary. NZ has adopted a new curriculum starting this month for all grades.

The Tui is a native bird of New Zealand. In Maori the word means sew/stitch/bind.

This started when my son was 5, and parents had to explain what they did for a living. Easy for the nurse etc., but what about the algorithms specialist (me)?

One of my local schoosl got a CS award for Powerpoint! My colleague Fellowes (British Columbia) apparently invented "computer science is no more about computers than astronomy is about telescopes", appropriated by (or at least on behalf of) Dijkstra.

Demonstration of bar code check sums. Note that it's 12 digits in USA, and 13 elsewhere. Verification seems to be $3\sum \text{odd} + \sum \text{even} = 0$.

Blows up a digital photograph using his unplugged software, and ends up with RGB values. "Oh yes, I guess a digital photograph should have digits in".

Story about Janina, who "hated CS with a passion", but ended up at Cambridge, PhD (rewriting Java compiler) and now works at Google.

Q I'm dyslexic and couldn't do one exercise.

A This raises very serious cultural and other

Q–Puerto Rico Your tools are really useful for us as we try to do CS4All in an country without electricity!

A Thanks.

Q–compere How many people here are using this?

A About 2/3 of the audience stood up.

Q I use CSunplugged for algorithms, but parents complain that this isn't CS.

A Indeed. We are doing a major chance of attitude here. Media can be a help or a hindrance.

Chapter 7

CECE's Map of Informatics in European Schools

7.1 Introduction

This project (ACM Europe + Informatics Europe) ran for two years. Follow-up from a 2013 report by same bodies.

7.2 Terminology

Wide variety of terms across Europe, but those related to "Informatics" are a majority. "IT covers the foundations of cmputational sctctures, processes, artifacts and systems, and theor software designs, their applications and their impact on society". Digital literacy "covers fluency with [...] tools".

7.3 Informatics availabiity

Note that Germany is 16 Länder (Spain also has devolution, but the Spanish "delegate let us down").

Primary 6/50; UK etc

Secondary Compulsory in 11/50, Available to all: 22/50 (possibly 24, but the other 2 don't guarantee in practice)

Available to some 10/50.

Combined with DL 2/50.

Not available 3/50

91% have a "national curriculum". School autonomy in Flanders, Estonia, Finland, Ireland, Sweden.

7.3.1 Recommendations

These include "only taught by those with formal qualifications", but speaker said that these were "final goals".

7.4 Digital Literacy

First contact data.

Primary 41/53

Lower Secondary 9.53

Upper Secondary 2/53

Flanders Kindergarten.

Is it a separate subject?

Integrated 27

Separate 27

Curriculum definition

National 40/53

School Autonomy 13/53

7.4.1 Recommendations

1. Should not be confused or be a substitute for informatics.
2. Teaching should follow an agreed upon, general curriculum that is perioidically updated to reflect new developments in information technology. Not only skills, but use safely, effectively and ethically.
3. appropriate teacher traning.

Note that even "literate" has changed over the last two centuries.

Q–FH Sweden uses European "Digital Competency Framework".

A We worry that there is a view that this is sufficient.

7.5 Teacher Training

"To teach is to touch lives forever".

47/54 have informatics teaching qualifications available. 92% have an informatics curriculum available. For DL is is available in 8/43. In-servicetraining: 1-12 months 37%, 12–24 months 47% and none 3 /45 (6%). Note that the definitions vary.

In theory, 29/48 require a Masters to enter, 9/48 a Bachelors and 10/48 have no requirement. This survey didn't distinguish level of teaching.

3/46 countries allow professionals to become teachers, 26/46 require pedagogical training. In 39/47, professionalism is not allowed to replace the degree requirements.

Educational requiremets for Informatics teachers should follow the same requirements as for science teachers.

7.6 Next Steps

Joint Informatics4All initiative to be launched in Brussels 15 March. JHD has an invite, but clashes with BCS AGM and other events.

- Establish Informatics as an essential discipline fro all, a subject available at all levels throughut the educational level
- All students and teachers have to be not only digitall literate
- All citizens receive an appropriate level of Informatics education.

Q–AMcG This initiative was born out of the US "running on empty".

A Indeed: should have mentioned it.

Chapter 8

Algorithms

8.1 Quick-Sort: A Pet Peeve

[NHGW18], considering Quicksort versus Mergesort in Haskell. Quicksort naively is 1.45 times faster than Mergesort, with a one-pass partition function 2.44, and with that and accumulator 4.01 times faster.

Q In-place?

A None of these.

8.2 Map-based Algorithm Visualization with METAL Highway Data: Teresco

[TFZ+18]. Teaching algorithms and data structures for a while. Resurrected an old site and called it TravelMap. Originally used this for teaching in the conetxt of Dijkstra. Recently got funding and added visualisation.

Q

A

8.3 Student Misconceptions of Dynamic Programming: Zehra

[ZRZZ18]

1. identify subproblems
2. define the size and values of array used
3. define a recurrence.

4. Iterative code pseudocode

[EnstromKann2017a] study on Algorithms and Compilers course.

"Understanding the problem" seems to be a big issue, even when the underlying problem is constant.

Chapter 9

Software Engineering

9.1 Developing SE Skills using Real Tools for Automated Grading: Heckman

[HK18]. We want our students to be prepared for industry as soon as possible. Students are novice professionals. "Canary Framework" supports student project development process: implementation and unit testing in particular. Use Eclipse,JUnit, Jacoco, Checkstyle etc. locally. GitHub and Jenkins[1], driving JUnit, Jacoco, Checkstyle etc. We have an enterprise GitHub that the university pays for.

- Version Control is a critical skill for enterprise-ready students.

It also facilitates team work, and kills the "hard drive died" e-mails.

CI evaluates student projects. The real tools aren't quite set up for academic use cases. We have tools to generate repositories and assign students to teams. Creating Jenkins jobs automatically etc. Testing and Library checkers. Would like to do more automated data collection. Also we use time limits for algorithms/data structures problems, and so on.

1. One course at a time
2. Follow institution guidelines to support automated grading.
3. 5–10 hour setup, maintenance 5–10 hours across a 15-week semester.

Data Mining: better students start earlier and do more commits (yes), bt we also find the students who work hard but don't do well, and these are the ones I want to help. `sos-cer.github.io`.

Q Where is it used?

A A lot of courses, even behind the scenes in CS1.

[1] Looks at Travis, but too expensive. Jenkins is free.

Q If the students write the tests, how to you spot "skivers" (JHD's phrase)?

A We can spot the trivial cases `assert true true`. I'd like to explore mutation testing.

9.2 Specification-Based Testing in Software Engineering Courses: Fisher

[FJ18]. Goals

1. Teach students about specification.
2. Teach students about software testing, and its importance for quality.
3. Teach students about benefits of formal methods.

If we had a tool that would generate test, would that motivate you to write specifications? — students say "yes". [JonesSIGCSE2001: Arsenic in small doses]. JUnit[2] has shown how a better tool can improve acceptance of testing.

Has a comparison of Specification-Driven Development and Test-Driven Development: former generates the tests from the specification. Both can be Agile!

Student feedback was neutral at best. The comments were useful. Lack of IDE integration is a real problem. The tool (Spest) was not the production-quaity the students expected. Expert-written tests are more readable, but the differences weren't enormous. On a sample that was good enough code, student tests had 91% coverage; Spest 100%.

Q That 91% looks pretty amazing.

A But

9.3 Integrating Project Based Learning and Project Management for Software Engineering Teaching: An Experience Report: Fioravanti

Use IEEE2010 definition of SE. Guideliens say SE mst be addressed. Traditoally atughtthrough lectures and with static methods.

Had a "real client", with a graduate student as project manager with 8–10 undergraduate students as workers.

1. Requirements Elicitation
2. Planning and Modelling
3. Prototyping and Systems integratio

[2] It wasn't clear why, to the speaker at least.

4. Presentation

Some "hiring and firing" between stages.

"PM important" gave 55% strongly agree. "Contact with real customers" git 59% strongly agree.

Q Hiring/Firing?

A Managers evaluate team members, and worst were fired, and *had* to e rehired by other groups. Happened after stages 2 and 3. Betwen 1 and 2 the graduate students rotated.

Q Deleopment model.

A Chosen by project manager. Often scrum.

Q What were you?

A I was one of the project managers: that's why there were so many authors!

Part IV

24 February 2018

Chapter 10

CS Education Around the Globe

10.1 Bringing Computer Science Education to Secondary School: A Teacher First Approach: Neutens

[NW18]. European Council quote on importance of education. He is speaking from a Flanders point of view. Flanders 2012–16 540–630 students with 5% unemployed after one year. Other degrees are also stagnant in numbers and unemloyment rates, but the unemployment rates range from $< 1\%$ to $> 22\%$ depending on degree title.

The secondary school goals (as set by government) are all really digital literacy. Several extra-curricular activities already. We started "Progra-MEER" as a workshop for teachers: 61 over two years. Covered all grade levels of secondary school, and a wide range of backgrounds. 6 workshops across the school year, with homework. Wanted to encourage cohort-building. One session was two whole days at the university on their project with tutor support. These data are not in the paper.

But 37/61 completed, major comment was "too hard". 14/37 reported doing the project in their classroom. Survey of teachers, also of pupils. Girls were less motivated than boys by the projects, so we need to add a gender aspect to the training.

10.2 An Agile Conversion Masters Degree Programme in Software Development: Lundqvist

[LAH+18] describing a Masters programme at Wellington. Massive skill shortage: many students are grabbed part-way through the Masters. Research shows

that soft skills are lacking in the regular CS cohort. Despite the research, Government decreed "industry-driven Masters". Conversion one, 12 months: "you must have a Bachelor's". Depth aimed at equivalent to a BSc. Prior 4-week boot camp[1] if there's no programming experience (19 out of 25 did boot camp). Two groups "past experience doesn't get a job" and "want acceleration". Both sets are motivated: best I have seen. First studio is waterfall: "they realise they never want to do that again", and the remaining studios are Agile.

Try to introduce industrial problems towards end of course. Agile as the principle. Teach Monday–Thursday, and industry seminars on Friday. 75% had jobs before they graduated. Assessment largely by written report: the English majors did really well. We also think this is good preparation for analyst/designer jobs.

Q Programming problems?

A Three weeks as a group, e.g. a LinkedIn equivalent for renters. Industrial partners came in daily. These were quite difficult problems.

Q Staff.

A 5 Faculty involved. Industrials: we have a lot, probably 40. They are not compensated, but get access to the students. Often offer (paid) internships.

10.3 Language Choice in Introductory Programming Courses at Australasian and UK Universities: JHD

[SMC+18]

Q Are you assuming Python isn't object-oriented?

A *We* are not assuming that, but the questionnaire audience seem to be. It is a rather strange finding, but replicated in Australasia and the UK.[2]

Q I am from Australia, and used to teach Java, switched to Python, found the students were not as well prepared for subsequent courses, and switched back.

A I can see that if the subsequent courses didn't change at all.

[1] 1 week Lego Mindstorms then 3 weeks Java. 60% group project, with no support for group work — "that comes later".

[2] A subsequent discussion may have clarified this. By "object-oriented" JHD's interlocutor meant "subtyping and polymorphism with static typing".

Chapter 11

Ethics

11.1 Ethics Education in Context: A Case Study of Novel Ethics Activities for the CS Classroom: Skirpan

[SBB+18]. Impacts of AI in judicial sentencing, identity determination, medical judgements, flows of "public" information. Note the rôle of even sorting in terms of "top of your news feed". ABETT says 20 hours of "professionalism", but this tends to be other things than ethics. Leads to "ethics is for the lawyers" mindset. Evidence that ethics increases retention, especially amongst under-represented.

We took "Human Centred Design", and made ethics a common thread. Significant ethics (and history of decisions) material in the learning objectives. 10% of the grade was participation (manually logged): verbal or submitting an essay. 50% team project, 30% individual essays, 10% reflections. 5 weeks at 3 $\times$2.5 hours/week= lecture + workshop. Lecture is a "no technology" setting. Course contract on respect and open-mindedness etc. 5 guest lectures, including emeritus professor on "changes in last 40 years".

Started with foundations of ethical thinking: "Do Artefacts have Politics". Example: estimate wealth from social media data set. Also "from Moodle data, who deserves a scholarship", which brings up bias.

Q Summer format?

A Probably helped with attendance. But I've used the activities before.

Q Datobase courses etc.?

A Data Science course is where some came from. Also been used in a vision class.

Q How do you cope with lack of background?

A Good question. History of technology courses would be useful. A lawyer retraining described use of facial recognition technology.

11.2 Quantified Self: An Interdisciplinary Immersive Theater Project Supporting a Collaborative Learning Environment for CS Ethics: Cameron

[SCY18]. Shows a clip of immersive theater. Audience[1] from theater courses, but also Google etc. "CS students don't have a sense of what people are comfortable with".

22 students including 3 PhD project as leads, from seven departments. This was extra credits, or performance credits for theter students.

Theme: invited by "Amelia", actually a bot using your social data. 8 characters for different personalities: CEO, lawyer, hacker etc.

The technical skills were ReactJS, P5.js, and Phython, public APIs to pull social media data. Emphasis on privacy and encryption. First class had to include pronunciation lessons for the non-techies! The theater students were interacting with the audience, often as improvisation.

Tech. students changed their appreciation of how people view data sharing: "the are terrified when we ask for data, but they put it all on Facebook". Non-techies learned about what's really happening. A "terms of service" exhibit whcih got more and more ridiculous was a feature. One gain was "improved conversational ability". The non-techies (including women) wanted to learn coding/data science, not just talk about it. While the non-techies would speculate about the future, the techies wouldn't — an issue to address for the future.

Fiction is a neutral area for discussion. There's a talk at CHI'18, and a re-run in Pittsburgh in 2019.

Q "Techies won't speculate" is worrying. The are designing the future.

A That was in the context — but I did see it in the long term.

Q Other models?

A The audience wanted deeper dives: how do you structure this?

11.3 Key Concepts for a Data Science Ethics Curriculum: Saltz

I teach "Introduction to Data Science" ug and pg, and students from CS, engineering and public policy, business etc. First year was all content.

RQ1 What should he material be?

RQ2 Are current codes of conduct (Data Science Association, ACM) work.

[1] But of course these are also participants.

76% of DS professionals think that ethics should be in education, but 30 DS programs were analysed and there was no ethics in the descriptions. EU project EDISON has a Data Science Model Curriculum (MC-DS) with no ethics. Identified challenges:

1. Privacy and Anonymity: aggregation and linking bring new challenges;
2. Data Misuse;
3. Data Accuracy.

Need to look at subjectvity in model creation. Also questions of explanation/transparency.

Analysed a variety of codes: "Accenture's 12 principles" did well, probably best.

Q—ACM Committee Draft 3 (Comm ACM January 2018) covers all the boxes.

A Thanks.

Q National Academy online ethics center: post your papers there.

A Various collections, but none complete.

Chapter 12

Epilogue

12.1 Observations

1. At the closing speech, the speaker (Ruthe Farmer, Chief Evangelist, CSforAll) produced a list that's new to me: "Digital 5" — UK, South Korea, Estonia, Israel, New Zealand. Observed that the US was merely a bystander here.

2. Significant number of papers presented by graduate students who were teaching assistants on the various courses.

3. Much use of GitHub (or others) to support programming, autograding and plagiarism detection. See sections 5.5 and 9.1, but also others, and casual conversation.

4. See footnote 2.

5. Fredrik Heintz drew JHD's attention to [AVE+18], based on data from zybooks exercise submissions. In particular, it compares very similar C++ and Python exercises. Under a fairly arbitrary, but plausible, definition of "struggle", they have the surprising result that Python students are twice as likely to struggle as C++ students, even when restricted to courses with no programming pre-requisites [AVE+18, Table 5].

6. Fredrik won a "Best Paper" award for [HM18].

12.2 Denmark

JHD (+Miles Berry) met two Danish civil servants. Apparently currently "Informatics" is a gymnasium subject, optional in three tracks, and compulsory in the fourth. But in Autumn 2018 they are about to start a three-track experi-

ment in schools[1], each school following one track. After 3 or 4 years, they will evaluate.

1. Free-standing subject (the Danish title would translate to "understanding technology", but that's not appropriate, as it includes doing as well as understanding).

2. Integrated with 2–3 other subjects such as Maths.

3. Utterly pervasive, even in, say, Music.

They haven't decided whether 2018 will be the first year of teaching, or a year of teacher training. JHD strongly suggested the latter,

They also asked about CAS. JHD said it was great, but strongly urged them to join themselves.

[1] JHD assumed all schools. But he cross-checked, and in fact it'll be in 50 schools in all.

Bibliography

[AVE+18] N. Alzahrani, F. Vahid, A. Edgcomb, K. Nguyen, and R. Lysecky. Python Versus C++: An Analysis of Student Struggle on Small Coding Exercises in Introductory Programming Courses. In *Proceedings SIGCSE 218*, pages 86–91, 2018.

[DFEG18] D. Deb, M. Fuad, J. Etim, and C. Gloster. MRS: Automated Assessment of Interactive Classroom Exercises. In *Proceedings SIGCSE 2018*, pages 290–295, 2018.

[FJ18] G. Fisher and C. Johnson. Specification-Based Testing in Software Engineering Courses. In *Proceedings SIGCSE 2018*, pages 800–805, 2018.

[HK18] S. Heckman and J. King. Developing Software Engineering Skills using Real Tools for Automated Grading. In *Proceedings SIGCSE 2018*, pages 794–799, 2018.

[HM18] F. Heintz and L. Mannila. Computational Thinking for All: An Experience Report on Scaling up Teaching Computational Thinking to All Students in a Major City in Sweden. In *Proceedings SIGCSE 2018*, pages 137–142, 2018.

[HTBV+18] G. Haldeman, A. Tjang, M. Babeș-Vroman, S. Bartos, J. Shah, D. Yucht, and T.D. Nguyen. Providing Meaningful Feedback for Autograding of Programming Assignments. In *Proceedings SIGCSE 2018*, pages 278–283, 2018.

[KS18] S. Krusche and A. Seitz. ArTEMiS: An Automatic Assessment Management System for Interactive Learning. In *Proceedings SIGCSE 2018*, pages 284–289, 2018.

[LAH+18] K. Lundqvist, C. Anslow, M. Homer, K. Bubendorfer, and D. Carnegie. An Agile Conversion Masters Degree Programme in Software Development. In *Proceedings SIGCSE 2018*, pages 846–851, 2018.

[NHGW18] A. Nunes-Harwitt, M. Gambogi, and T. Whitaker. Quick-Sort: A Pet Peeve. In *Proceedings SIGCSE 2018*, pages 547–549, 2018.

[NW18] T. Neutens and F. Wyffels. Bringing Computer Science Education to Secondary School: A Teacher First Approach. In *Proceedings SIGCSE 2018*, pages 840–845, 2018.

[PHYL18] Y. Park, H. Hu, X. Yuan, and H. Li. Enhancing Security Education Through Designing SDN Security Labs in CloudLab. In *Proceedings SIGCSE 2018*, pages 185–190, 2018.

[SBB+18] M. Skirpan, N. Beard, S. Bhaduri, C. Fiesler, and T. Yeh. Ethics Education in Context: A Case Study of Novel Ethics Activities for the CS Classroom. In *Proceedings SIGCSE 2018*, pages 940–945, 2018.

[SCY18] M. Skirpan, J. Cameron, and T. Yeh. Quantified Self: An Interdisciplinary Immersive Theater Project Supporting a Collaborative Learning Environment for CS Ethics. In *Proceedings SIGCSE 2018*, pages 946–951, 2018.

[SMC+18] Simon, R. Mason, T. Crick, J.H. Davenport, and E. Murphy. Language Choice in Introductory Programming Courses at Australasian and UK Universities. In *Proceedings SIGCSE '18: Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 852–857, 2018.

[TFZ+18] J.D. Teresco, R. Fathi, L. Ziarek, M. Bamundo, A. Pengu, and C.F. Tarbay. Map-based Algorithm Visualization with METAL Highway Data. In *Proceedings SIGCSE 2018*, pages 550–555, 2018.

[YMSP18] L. Yan, N. McKeown, M. Sahami, and C. Piech. TMOSS: Using Intermediate Assignment Work to Understand Excessive Collaboration in Large Classes. In *Proceedings SIGCSE 2018*, pages 110–115, 2018.

[ZRZZ18] S. Zehra, A. Ramanathan, L.Y. Zhang, and D. Zingaro. Student Misconceptions of Dynamic Programming. In *Proceedings SIGCSE 2018*, pages 556–561, 2018.