The New Normal of Teaching Computer Science

In my previous column [9], I discussed whether Computer Science (CS) educators could act as role models for other fields, because we have a long tradition in building advanced educational tools and rich interactive learning content to support online and blended learning practices in our field. When authoring the column text, I had no clue how extensively this tradition would be tested in 2020, when almost suddenly all education in universities had to be turned online due to COVID-19 pandemic. My university campus was closed in the middle of March and all education and coordination had to be arranged from home. My other colleagues had the same challenge, as well as most national and international colleagues whom I know.

For myself and my closest colleagues, this transition, at least in the first- and second- year courses, was surprisingly smooth. For some colleagues elsewhere, working apparently seemed chaotic for a while and I have pondered some reasons behind these differences. Our experience in developing tools and pedagogical methods to support blended learning provided us great advantage when we faced the challenge of turning all education online. On the other hand, the situation has also emphasized that we have to reconsider the process of creating and maintaining learning content, which is the main theme of this column.

At Aalto University, we have a long experience to support students' learning process in terms of online tools and resources. Online guidance for exercises was started in late 1980's using Unix news channels and IRC. Automatic assessment of algorithm simulation exercises was started in 1991 [8],

For myself and my closest colleagues, this transition [to online teaching], at least in the first- and second- year courses, was surprisingly smooth. For some colleagues elsewhere, working apparently seemed chaotic for a while and I have pondered some reasons behind these differences.

and the practice was adopted to programming exercises in 1994 using Ceilidh [4], which was replaced by other locally developed systems later. When WWW emerged in higher education after 1995, and especially when it became possible to build interactive Java applets and later on use Javascript tools, we invested a lot of work in building learning tools and rich interactive learning content, also called *smart learning content* [5]. In several courses, students are working with an interactive locally built e-book. In general, our goal has been to activate students to work online independently or in teams and give them rich online feedback automatically. This has reduced the role of lectures, as well as closed labs. Voluntary classroom sessions, guided by teaching assistants, have an important role in assisting students to solve weekly exercises. The role of final examination has also been reduced, and some courses do not have an exam at all, because students have to demonstrate

their learning in project demonstrations or because we rely on students' motivation to learn (this is not an issue for CS majors; in service courses, exams seem to be more important). Projects are graded by teachers and teaching assistants with supporting tools, such as Rubyrac [3].

Thus, most students' learning activities were already available online before the pandemic began. The additional work to turn the whole courses online included mainly video recording the remaining lectures, turning closed labs and individual guidance meetings into video meetings, and giving good instructions to students. These were organized with typical tools like PanOpto or Zoom. Organizing exams remained a challenge—and no fully satisfying solution was available. Due to legal limitations related to personal privacy, we could not organize proctored online exams. Some teachers gave up exams totally, some replaced them with personal assignments. In large courses with many

The New Normal of Teaching Computer Science

hundreds of students, Moodle exam was used, even though some students might not solve the tasks on their own.

In summary, the big picture in the transition to online was very positive. No chaos emerged and most students were coping well with the new situation. All this is fantastic. However, there is another side of the coin, which causes worries for the future: *how to cope with the increasingly complex development and maintenance of such rich learning resources?* Moreover, these resources cannot be research prototypes; they are tools targeted to production use for thousands of students. These challenges are well discussed in a recent paper by my colleagues [6], and I summarize and augment some experiences here.

E-books and CS learning environments have more and more incorporated tailored software. Some resources have been developed locally, some are open source software developed elsewhere, and some could be commercial tools. Several years ago, our programming courses employed half a dozen different systems, each of which had their own user interface, login access and reporting. It was clumsy for teachers to combine results from several tools for calculating the overall grades, as well as for students to learn to use several systems where they had to login during the courses. We therefore developed the A+-environment [1,7] which includes student identification, provides a common user interface, and has a data base for recording submissions, student activities and results. The architecture is based on micro services, which integrate external tools or services with the core system, such as various exercise types or grading services, by using LTI or tailored protocols.

From the students' point of view, the system works generally well, despite occasional bugs and service breaks. The teacher's view is much more complex. Source code for the course material is presented as reStructuredText or Latex, which incorporate locally tailored tags to access external services. The code is maintained in local GitLab version control system and compiled into HTML pages that present the static and smart learning content in iframes within the A+ user interface. Creating new content, such as new auto-

matically tested exercises, includes coding, debugging and testing. Unit tests need to be coded and tested to ensure students get the right number of points when a test passes or fails. Installing the exercises into the system means writing scripts that enable running the tests with the student's

are customers for using the system and associated content. Managers are interested in requested system resources as well as recruitment of competent developers.

The availability of these resources was a huge advantage when we had to make the transition into pure online learning,

I foresee that this—international collaboration—is the direction to proceed. Commercial educational services provide partial solutions for using smart learning content. ... Building bridges to overcome the integration challenges with other tools and environments is crucial.

programs in Docker containers. We are using Scala as the main language on first year CS courses, and therefore few external learning tools have been available.

In reality, content creation has turned into a software project with several stakeholders. Development of the A+-system itself and integrating any new services with it, require considerable expertise, and many teachers are not familiar with how everything works. Our CS department has had to recruit a separate software team to take responsibility of the environment, because the university IT support team did not have the resources and expertise for the task. We need scheduled service breaks which are, of course, shared with students and teachers. We cannot afford a 24/7 service team, which has led into some problems when something goes wrong, for example, during a weekend. New features that teachers would like to have will be scheduled in future releases of the software. We summarize this to show that the original project to enable better pedagogy through providing smart learning content for students has become a full-fledged software project, where software engineering methods have to be applied. There are several different types of stakeholders. Teachers are skilled customers creating new content, as well as end-users for the teacher front-end services. Students

because they were already in place and continued operations as normally. Students are generally happy of the fast feedback available on their work.

One could argue here: *"What is new with this? You have decided to build a tailored learning environment. It is a big software project which needs software engineering approaches to be successful. It is your choice and you need to pay the costs."* This is true in some respect. However, this is not a normal software project. The growth of the system has taken a long time and a lot of research to test the use and impact of various types of smart learning content and pedagogical practices. There was no initial draft specification agreed with a customer; the development has been agile and slow because courses are mostly run once a year. Therefore, the next versions are tested a year later, when the evaluation study has been completed and/or the teachers' and students' experiences have been analyzed.

What could be the alternatives? There are a limited number of commercial software available, for example, for automatic grading. They have different types of licenses and costs. It is unclear, if they can be integrated with other tools we are using and how flexible they are for new types of exercises. There are lots of different types

of educational software that have been developed for research purposes. Some of them are open source, some are not, or there is insufficient documentation. Regardless of this, integrating them with other tools is not straightforward. Therefore, some international collaborative projects have been launched, which seek to build better interoperability of existing tools, such as the SPLICE project (Standards, Protocols, and Learning Infrastructure for Computing Education) [11] or its subproject PEML (Programming exercise markup language). Another type of solution is the ACOS content server that provides as resources different types of exercises and interactive content that can be integrated with learning management systems using different protocols [2,10]. I foresee that this—international collaboration—is the direction to proceed. Commercial educational services provide partial solutions for using smart learning content. On the other hand, CS teachers often seek innovation

and flexibility which are hard to match with existing software, and therefore tailored solutions are quite common. However, they should not remain as isolated local custom solutions. Building bridges to overcome the integration challenges with other tools and environments is crucial. ✦

References

1. A+ LMS. The extendable learning management system; <https://apluslms.github.io/>. Accessed 2020 October 31.
2. ACOS Content Server; <http://acos.cs.hut.fi/>. Accessed 2020 October 31.
3. Auvinen, T., Rubryc. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, (ACM, 2010), 102–106.
4. Benford, S. D., Burke, E. K., Foxley, E., & Higgins, C. A., The Ceilidh system for the automatic grading of students on programming courses. In *Proceedings of the 33rd annual on Southeast regional conference*, (1995), 176–182.
5. Brusilovsky, P., Edwards, S., Kumar, A., Malmi, L., Benotti, L., Buck, D., Ihantola, P., Prince, R., Sirkkiä, T., Sosnovsky, S. and Urquiza, J., Increasing adoption of smart learning content for computer science education. In *Proceedings of the ITICSE Working Group Reports*, (ACM, 2014), 31–57.
6. Haaranen, L., Mariani, G., Sormunen, P., Lehtinen, T., Complex Online Material Development in CS Courses. To appear in the *Proceedings of 20th Koli Calling*

Conference of Computing Education Research, (ACM, 2020).

7. Karavirta, V., Ihantola, P., and Koskinen, T., Service-oriented approach to improve interoperability of e-learning systems. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*, (IEEE, 2013), 341–345.
8. Korhonen, A., and Malmi, L., Algorithm simulation with automatic assessment. In *Proceedings of the 5th annual SIGCSE/SIGCUE ITICSE conference on Innovation and Technology in Computer Science Education*, (ACM, 2000), 160–163.
9. Malmi, L., COMPUTING EDUCATION RESEARCH. Could we be role models in online/blended education? *ACM Inroads*, 10, 3 (2019), 12–14.
10. Sirkkiä, T., & Haaranen, L., Improving online learning activity interoperability with acos server. *Software: Practice and Experience*, 47, 11 (2017), 1657–1676.
11. SPLICE: Standards, Protocols, and Learning Infrastructure for Computing Education, <https://cssplice.github.io/>. Accessed 2020 October 31.

Lauri Malmi
Department of Computer Science
Aalto University
P.O.Box 15400
00076 AALTO Finland
Lauri.Malmi@aalto.fi

DOI: 10.1145/3433692

Copyright held by author.

ACM Computing Surveys (CSUR)

2018 JOURNAL
IMPACT FACTOR:
6.131

Integration of computer science and engineering knowledge

ACM Computing Surveys (CSUR) publishes comprehensive, readable tutorials and survey papers that give guided tours through the literature and explain topics to those who seek to learn the basics of areas outside their specialties. These carefully planned and presented introductions are also an excellent way for professionals to develop perspectives on, and identify trends in, complex technologies.

For further information and to submit
your manuscript, visit csur.acm.org