

Assessing Learning In Introductory Computer Science

Edited by

Michael E. Caspersen¹, Kathi Fisler², and Jan Vahrenhold³

1 Aarhus University, DK, mec@cs.au.dk

2 Worcester Polytechnic Institute, US, kfisler@cs.wpi.edu

3 Universität Münster, DE, jan.vahrenhold@uni-muenster.de

Abstract

This seminar discussed educational outcomes for first-year (university-level) computer science. We explored which outcomes were widely shared across both countries and individual universities, best practices for assessing outcomes, and research projects that would significantly advance assessment of learning in computer science. We considered both technical and professional outcomes (some narrow and some broad) as well as how to create assessments that focused on individual learners. Several concrete research projects took shape during the seminar and are being pursued by some participants.

Seminar February 14–19, 2016 – <http://www.dagstuhl.de/16072>

1998 ACM Subject Classification D.2 Software Engineering, D.3 Programming Languages, F.2 Analysis of Algorithms and Problem Complexity, F.3 Logics and Meanings of Programs, K.3.2 Computer and Information Science Education

Keywords and phrases Assessment, Learning Objectives

Digital Object Identifier 10.4230/DagRep.6.2.78

Edited in cooperation with Mirko Westermeier

1 Executive Summary

Jan Vahrenhold

Michael E. Caspersen

Kathi Fisler

License  Creative Commons BY 3.0 Unported license
© Jan Vahrenhold, Michael E. Caspersen, and Kathi Fisler

The goal of the seminar was to focus on several broadly applicable learning outcomes for first year university computer science courses, looking at what it would take to understand and assess them in multiple pedagogic contexts.

In preparation for the seminar, we surveyed participants to get an understanding of a what could be a common denominator of CS1/2 learning outcomes, using the outcomes from the ACM CC 2013 curriculum as a starting point. We asked participants to (a) identify ones that are covered in their institution’s CS1/2 courses, and (b) to identify ones that they have either experience or interest in investigating further. Participants also suggested objectives that were not included in CC 2013.

Of these candidate outcomes, we studied a subset during the seminar, as voted by the participants. We used breakout sessions to get small groups of participants to focus on individual outcomes, reporting on what is known about each outcome, its underlying challenges and/or relevant underlying theory, how to best assess it, and what sorts of research questions should be asked to advance educational research on that outcome. We had three



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Assessing Learning In Introductory Computer Science, *Dagstuhl Reports*, Vol. 6, Issue 2, pp. 78–96

Editors: Michael E. Caspersen, Kathi Fisler, and Jan Vahrenhold



DAGSTUHL REPORTS Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

separate sets of breakout sessions, so each participant had the chance to work on three outcomes in detail during the week. The discussion of some sessions was continued in a following session.

Rather than have most individual participants give talks, we ran three speed-dating poster sessions on the first afternoon: each person got to put up a poster on some outcome that they have studied, so others could see the research of other attendees.

In addition, we had three invited presentations focussing on workload and determinants of study success (Schulmeister), types of prior knowledge and their relation to study success (Theysen), and Concept Inventories (Kaczmarczyk and Wolfman). The abstracts of these presentations are included in this report.

2 Table of Contents

Executive Summary

<i>Jan Vahrenhold, Michael E. Caspersen, and Kathi Fisler</i>	78
---	----

Overview of Talks

A Concept Inventory Crash Course <i>Lisa C. Kaczmarczyk and Steven A. Wolfman</i>	82
ZEITLast-Project (workload project): List of research activities with reference to computer science <i>Rolf Schulmeister</i>	83
Types of prior knowledge and academic success in biology and physics <i>Heike Theysen</i>	85

Working groups

Decomposition and Algorithm Selection <i>Notes by Mirko Westermeier</i>	85
Notional Machine <i>Notes by Mirko Westermeier</i>	86
Processes and Decomposition <i>Notes by Mirko Westermeier</i>	86
Social and Professional Issues <i>Notes by Mirko Westermeier</i>	87
Spatial Thinking <i>Notes by Mirko Westermeier</i>	87
Tracing and Debugging <i>Notes by Mirko Westermeier</i>	87
Assessments with Learners in Mind <i>Steven A. Wolfman</i>	88

Panel discussions

Assessment Techniques Brainstorming <i>Notes by Mirko Westermeier</i>	89
Next Steps Discussion <i>Notes by Mirko Westermeier</i>	90

Poster abstracts


Programming Education for Novices <i>Michael E. Caspersen</i>	90
KETTI – Competence Development of Student Teaching Assistants in Computer Science <i>Holger Danielsiek</i>	91

A Method to Analyse Computer Science Students' Teamwork in Online Collaborative Learning Environments <i>Katrina Falkner</i>	91
Plan Composition <i>Kathi Fisler</i>	92
Critiquing CS Assessment from a CS for All Lens <i>Mark Guzdial</i>	92
Communicating through Sketches and Diagrams <i>Geoffrey L. Herman</i>	92
Neo-Piagetian Theory and the Novice programmer <i>Raymond Lister</i>	93
Educational Data Mining <i>Andreas Mühling</i>	93
Teaching and Learning a First Programming Language <i>Anthony Robins</i>	94
Novice Programmers' Difficulties with Program Dynamics and Misconceptions of the so-called Notional Machine <i>Juha Sorva</i>	94
Qualitative Feedback of Program Code and Programs <i>Martijn Stegeman</i>	94
Potential Factors Indicating Success in an Algorithms and Data Structures Course <i>Jan Vahrenhold</i>	95
(Mis)conceptions Surrounding Exponential Growth in the Foundations of Computing Concept Inventory <i>Steven A. Wolfman</i>	95
Participants	96

3 Overview of Talks

3.1 A Concept Inventory Crash Course

Lisa C. Kaczmarczyk (San Diego, US) and Steven A. Wolfman (University of British Columbia – Vancouver, CA)

License  Creative Commons BY 3.0 Unported license
© Lisa C. Kaczmarczyk and Steven A. Wolfman

Concept Inventories (CIs) are short, low-stakes, often multiple-choice assessments used to gauge students' learning surrounding specific concepts within and across courses. Their ease of administration—both having students write the CI and analysing results—makes them ideal as sustainable instruments for measuring the impact of changes in teaching, particularly pedagogical shifts. CIs have been widely used in science fields, most famously and earliest the physics Force Concept Inventory. Well-designed CIs aim to not only assess whether students have expert-like conceptions of course material but also to identify commonly occurring misconceptions through their questions' distractors. By providing evidence of difficulty learning fundamental concepts and of widespread misconceptions, CIs can be instruments of change in teaching practice. Designing a CI requires a careful, multi-step process to identify the often-narrow scope of the assessment, solicit common misconceptions among students, craft assessment items targeting these concepts and misconceptions, and validate the items with both experts and students. While progress has been made on CIs for some topics in computing, significant challenges remain including the diversity of programming languages, the rapid pace of change in the field, fundamentally different programming paradigms, and codification of processes such as design and debugging as concepts.

Partial List of Concept Inventory Publications of Interest to Computing Education Researchers

- W. K. Adams and C. E. Wieman. Development and validation of instruments to measure learning of expert-like thinking. *Int'l. J. of Science Ed.*, 33(9):1289–1312, 2011.
- V. L. Almstrum, P. B. Henderson, V. Harvey, C. Heeren, W. Marion, C. Riedesel, L.-K. Soh, and A. E. Tew. Concept inventories in computer science for the topic discrete mathematics. In *Working Group Reports on Innov. and Tech. in CS Ed.*, pages 132–145, 2006.
- Goldman, Ken, Gross, Paul, Heeren, Cinda, Herman, Geoffrey, Kaczmarczyk, Lisa, Loui, Michael, Zilles, Craig. (2010, June). “Setting the Scope of Concept Inventories for Introductory Computing Subjects”. *ACM Transactions on Computing Education (TOCE)*.
- Goldman, Ken, Gross, Paul, Heeren, Cinda, Herman, Geoffrey, Kaczmarczyk, Lisa, Loui, Michael, Zilles, Craig. (2008). “Identifying Important and Difficult Concepts in Introductory Computing Courses using a Delphi Process”. Presented at the 39th Annual Technical Symposium on Computer Science Education (SIGCSE 2008), Portland, Oregon.
- R. R. Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 66(1):64–74, 1998.
- Herman, Geoffrey L., Loui, Michael C., Kaczmarczyk, Lisa, Zilles, Craig. (2012, March).

- “Describing the What and Why of Students’ Difficulties in Boolean Logic”. *ACM Transactions on Computing Education (TOCE)* 12(1), 3:1–28.
- Herman, Geoffrey, Kaczmarczyk, Lisa, Loui, Michael, Zilles, Craig. (2008). “Proof by Incomplete Enumeration and Other Logical Misconceptions”. Presented at the Fourth International Computing Education Research Workshop (ICER 2008), Sydney, Australia.
 - D. Hestenes, M. Wells, and G. Swackhamer. Force concept inventory. *The Physics Teacher*, 30(3):141–158, March 1992.
 - Kaczmarczyk, Lisa, Petrick, Elizabeth, East, J. Philip, Herman, Geoffrey L. (2010). “Identifying Student Misconceptions of Programming”. Presented at the 41st Annual Technical Symposium of Computer Science Education (SIGCSE 2010), Milwaukee, Wisconsin
 - E. Mazur. Qualitative vs. quantitative thinking: are we teaching the right thing? *International Newsletter on Physics Education*, 32, April 1996.
 - K. Karpierz and S. A. Wolfman. 2014. Misconceptions and concept inventory questions for binary search trees and hash tables. In *Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14)*. ACM, New York, NY, USA, pp. 109–114.
 - R. A. Streveler, R. L. Miller, A. I. Santiago-Roman, M. A. Nelson, M. R. Geist, B. M. Olds “Rigorous methodology for concept inventory development: Using the ‘assessment triangle’ to develop and test the thermal and transport science concept inventory (TTCI)”, *International Journal of Engineering Education* Vol. 27, No. 5, pp. 968–984, 2011
 - C. Taylor, D. Zingaro, L. Porter, K. C. Webb, C. B. Lee, and M. Clancy. Computer science concept inventories: past and future. *Computer Science Education*, 24(4):253–276, 2014

3.2 ZEITLast-Project (workload project): List of research activities with reference to computer science

Rolf Schulmeister (Universität Hamburg, DE)

License  Creative Commons BY 3.0 Unported license
© Rolf Schulmeister

The ZEITLast-project aimed to find out more about the real workload of our bachelor students by measuring the time needed to attend lectures and courses, for self study, preparing for exams, as well as used for extracurricular activities. Students used a web-based time budget or diary method to enter data daily during five months of a semester in order to cover the lecture period, the exam phase, and the self study activities during lecture free weeks. Data entered were controlled every day.

Time diary studies (time use, time budget) have been conducted in following IT curricula or computer science courses:

- B.Sc. Engineering Informatics, TU Ilmenau, no innovation, Summer 2010, 4th term: Workload 21.9 hours/week
- B.Sc. Engineering Informatics, TU Ilmenau, partly blocked modules, Summer 2011, 4th term: Workload 24.5 hours/week
- B.Sc. IT Security, St. Pölten, completely blocked modules, Winter 2011/12, 1st term: Workload 32 hours/week

- B.Sc. Informatik, Universität Paderborn, no innovation, Summer 2015, 2nd term: Workload 18.95 hours/week
- B.Sc. IT Security, St. Pölten, completely blocked modules, Winter 2015/16, 1st term: ongoing
- M.Sc. IT Security, St. Pölten, completely blocked modules, Winter 2015/16, 1st term: ongoing

The data of further 30 time diary studies in subjects other than computer science may be compared with the results of the computer science studies, e.g. architecture, mechanical engineering, culture, economy, education, mathematics, media science, media technology, mechatronics, ship building, medicine, electrical engineering, physics, etc. The mean workload in all 35 subjects tested so far is 23 hours per week. Even if the only data of the 14 weeks of lecturing are calculated, the mean workload is less than 30 hours per week and much less than expected by the Bologna system. An important finding is that the inter-individual variance and the intra-individual variance are enormously big, meaning that no student manages his or her time, attendance, and self study phases day for day or week for week in the same way. This finding throws a critical light at surveys that ask for data of one week and generalize them for a whole semester. Then mean values and percentages are misleading.

Some of the time use studies were followed up by a test on learning motivation and study habit based on an Integrated Model of Learning Activity and Motivation (according to Deci & Ryan, Julius Kuhl, and Thomas Martens). These tests were applied to students in education, economy, and computer science. The test generates motivational profiles of students focussing on learning behavior: anxiety, responsibility, distraction vs concentration, persistence vs procrastination, control of negative emotions etc.

Two meta-studies have been written in order to generalize the results of the empirical analyses:

1. “Auf der Suche nach Determinanten des Studienerfolgs”. in J. Brockmann/A. Pilniok (Hrsg.), *Studieneingangsphase in der Rechtswissenschaft*, Nomos: Baden-Baden 2014, S. 72–205.
 - 300 international studies dealing with determinants of learning success have been analyzed and summarized. This essay includes a paragraph on time diary analysis and an overview over international studies reaching comparable workload data (21–28 hours per week). The studies on study habits prove that traditional variables e.g. gender, social status, family etc. do not influence the learning outcome but motivational study habits, endurance, concentration, attendance etc. are determinants of GPA and learning success.
2. “Abwesenheit von Lehrveranstaltungen Ein nur scheinbar triviales Problem. Eine Meta-Studie von 300 empirischen Arbeiten.” Hamburg 2015.
 - An analysis of 300 international studies dealing with the attendance in lectures or absence of students from lectures and courses and the effect on the GPA. The degree of attendance is a good determinant of success in exams. Students missing three or more times a lecture or a seminar have lesser chances to reach an A or B.

References

- 1 Schulmeister, R. & Ch. Metzger (Hrsg.) (2011): *Die Workload im Bachelor: Zeitbudget und Studierverhalten. Eine empirische Studie*. Münster [u.a.]: Waxmann (report about the first 12 time use studies)
- 2 Schulmeister, Rolf, Christiane Metzger & Thomas Martens (2012): *Heterogenität und Studierenerfolg. Lehrmethoden für Lerner mit unterschiedlichem Lernverhalten*. Paderborner Universitätsreden Heft 123. Paderborn.

- 3 Christiane Metzger, Rolf Schulmeister & Thomas Martens (2012): Motivation und Lehrorganisation als Elemente von Lernkultur. Zeitschrift für Hochschulentwicklung. ZFHE Jg.7 / Nr.3 (Juni 2012).
- 4 Rolf Schulmeister, Christiane Metzger: Zur Rolle der Lehrorganisation bei der Gestaltung des studentischen Selbststudiums In: Brigitte Kossek / Markus F. Peschl (Hg.): Digital Turn? Zum Einfluss digitaler Medien auf Wissensgenerierungsprozesse von Studierenden und Hochschullehrenden. V&R unipress Vienna University Press 2012, S. 77–92.

3.3 Types of prior knowledge and academic success in biology and physics

Heike Theyssen (Universität Duisburg-Essen, DE)

License  Creative Commons BY 3.0 Unported license
© Heike Theyssen

High dropout rates in science, mathematics and engineering programs of study give reason to investigate predictors for academic success. Academic success is influenced by a variety of general and subject-specific factors. So far, most studies in this field have focused on general, non-specific factors like grade point average (Dochy, Segers & Bühl, 1999). The project presented in this talk is part of the research unit “Academic learning and study success in the entry phase of science and technology study programs” (https://www.uni-due.de/zeb/alster/index_engl) funded by the German Research Foundation. In our project we focus on students’ subject-specific prior knowledge that has proven to be a good predictor of learning success. According to Hailikari and her colleagues (2007, 2009, 2010) we distinguish different types of prior knowledge: knowledge of facts, knowledge of meaning, integration of knowledge and application of knowledge. In the talk, the theoretical background is introduced and the test instruments used to assess the different types of prior knowledge are presented.

4 Working groups

4.1 Decomposition and Algorithm Selection


Notes by Mirko Westermeier

License  Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

The process of decomposition plays a central role while problem solving. The skills required to approach problems and split them into subtasks is essential not only for programmers but also for most (technical) problem solvers in life, regardless of their discipline. So it seems to be easy to motivate teaching these skills. Open questions are when to teach them, how they could optimally sequenced and whether decomposition should be taught explicitly.

4.2 Notional Machine

Notes by Mirko Westermeier

License  Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

While trying to understand what hinders students by solving their programming tasks in a (for the teacher) straight-forward way, it sometimes turns out that their mental model of the machine on which the program runs is the problem. The biggest misconception here seems to be that the computer is somehow trying to understand what the student is doing.

The interesting part of this problem is, that most of these mental models (“Notional Machines”) are consistent and therefore their behaviour is reasonable even if it can not be synchronized with reality when going into depth. The notional machine can be a collection of consistent misconceptions.

Attendees agreed that it is important to find out something about the teacher’s own misconceptions and how they (if existent) match with the ones of the students. It seems to be important to know whether students have gaps in their mental models which lead to misconceptions and how to design the teaching such that it is covering these gaps. The open research question here seems to be how to find and assess gaps in notional machines.

4.3 Processes and Decomposition

Notes by Mirko Westermeier

License  Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

This breakout group discussed assessments in the context of understanding processes and decomposition of complex (programming) problems. An obvious starting point is debugging: while trying to fix a broken component or problem, students have to understand how the parts of the component work together and how the system changes over time processing different inputs. It seems that program comprehension and tracing of programs are prerequisites while becoming a successful programmer, but it is not so clear how to teach these skills. Attendees also were concerned that it could be dissatisfying to have students learn something that does not feel like coding to them. It could be necessary to do extra work to make these activities make sense to students. Therefore, teachers should have a clear awareness for these skills and need to motivate teaching them, which seems not to be easy in an obvious way. It was mentioned that these activities need an amount of creativity and somehow collide with usual computer science matters. Attendees are unsure how to assess the understanding of decomposition strategies.

4.4 Social and Professional Issues

Notes by Mirko Westermeier

License © Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

In this breakout group, attendees started with discussing what skills would be needed, required and useful in an industry position. It was stated that this is of course not the only important measure. Communities of Practice share these skills and a central question was what the intersections between common professional practices was. Another problem was the difference between these common professional practices and student's expectations and how to synchronize these: To eventually accomplish this and establish it in education, one not only needs to identify these professional practices but also to convey authenticity before even beginning to teach required skills.

Following questions would be to find out how to design curricula and tasks addressing identified communities of practice and how much of this would be able to teach and assess in student's first year.

4.5 Spatial Thinking

Notes by Mirko Westermeier

License © Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

In this breakout session attendees discussed the connection of spatial thinking with computer science, its possibilities and what it implies. For example was asked whether spatial reasoning skills of different types of teachers were also different and if different sub-disciplines in computer science lead to different spatial reasoning. Considering students, a main touch point of spatial thinking with computer science is data structures, e.g. different expressions of binary, balanced trees and how the intuitive balancing model possibly translates into code. Also applied to mental models of heaps (graphical and text-based) the special manifestation of spatial thinking obviously makes a difference.

4.6 Tracing and Debugging

Notes by Mirko Westermeier

License © Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

To define the skills necessary for tracing and debugging a specific program, students have to develop a line-by-line understanding because they have to identify and leave out not-so-important parts. To follow the control flow and variable states, they have to use some kind of a notional machine and mentally execute parts of the program. Tracing therefore also requires a good understanding of the program's object graph, which object is doing what and how they change state during execution. This can be hard to accomplish depending on the complexity of components. In a linear program, students just have to follow the variables while in complex nested object interaction even using polymorphism, misconceptions

in the mental model can become a real problem. Several tools to support better overall understanding were discussed. It was agreed that visualization tools (on different levels) help at first, but then a correct mental model is crucial.

The question how to assess tracing and debugging skills however remains hard. Attendees stated that it mostly never happened or failed because it was too hard. It seems important to watch students while trying to solve a problem and identify specific solving strategies in debugging. A defined set of these strategies should be testable. A belt system like in martial arts to state on which competence level a student is able to solve a task and what she has to master before was discussed.

Most ideas from this breakout session were discussed later in the *Notional Machine* breakout.

4.7 Assessments with Learners in Mind

Steven A. Wolfman (University of British Columbia – Vancouver, CA)

License  Creative Commons BY 3.0 Unported license
© Steven A. Wolfman

The design of learning assessments – widely construed to include both research instruments and formative or summative instructional instruments – often focuses on qualities like validity and reliability that speak to properties of the data produced by the assessment. Both the process and results of assessment also have a powerful impact on learners' journey through their course of study. We believe that instructors and researchers will benefit from becoming more mindful of the impact of their assessments on the assessed learners' affinity for their course of study.

To change this situation, we propose a new dimension which – for the time being – we call affinity with which to design and consider assessments. Affinity is conceptually orthogonal to existing dimensions like validity and reliability. We anticipate that an assessment with high affinity empowers and motivates students to engage in their course of study and conversely, that an assessment with low affinity disengages and isolates students from their course of study. The affinity dimension will explicitly reflect the student voice in the assessment process, structured by established, research-based frameworks [Resnick and Shaffer, Ryan and Deci, Bandura, and Engle] exploring the fit and effect of instructional processes on students' mindset, motivation, and perceptions within a course of study.

We intend to explore, explain, and validate the nature of this affinity dimension and – in the process – develop a pair of instruments, one usable by researchers investigating this new affinity dimension to guide and frame the data they collect, and another related instrument usable by instructors to improve the affinity of their designed assessments. We hope that students will benefit from both of these through assessments with higher affinity that improve their learning process and trajectory.

This agenda will involve collecting and analysing a corpus of existing assessments embedded in their contexts; observing and interviewing students as they journey through an assessment process; observing and interviewing instructors and possibly researchers as they design and manage assessments; iteratively designing the instruments described above; and, of course, exploring the validity, reliability, and affinity of the assessment instruments and practice we design.

At Schloss Dagstuhl, over the course of three consecutive breakout sessions, we began this process by discussing the impact of assessments on students and their utility for students.

In search of concrete data to frame our conversation, we collected from the educators and researchers at the seminar a wide set of examples of assessments (assessments with no special expectation that they are exceptional) and selected a small set of exemplars of assessments (assessments where there is reason to believe they display best practices with regard to affinity). We brainstormed a set of questions representative of the student voice in the assessment process; and collected research-based frameworks to guide our analysis of the impact of assessment on students' mindset, motivation, and perceptions of their course of study. We are now engaged in iterative adaptation of an existing instrument for exploring users' journeys through use of a service to the context of assessment, using the student voice questions, research frameworks, and concrete examples described above.

Our next steps include two ongoing agendas and a set of precise goals. On an ongoing basis, we will continue to expand and organize our corpus of assessment examples. We will also begin the design of a “dashboard” learners can use to integrate the process and results of assessments into their ongoing course of study. This dashboard creates a frequent series of touchpoints between the learner and the instructor/researcher, which represent opportunities to both improve and explore the affinity of assessments.

5 Panel discussions

5.1 Assessment Techniques Brainstorming

Notes by Mirko Westermeier

License  Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

To synchronize efforts across all participants, we discussed established assessment techniques and related open research questions.

Regarding the teacher's perspective, rapid diagnostics were mentioned to be an important tool to determine at what level instruction should start with beginners. For a continuous adjustments to student's abilities and progress, however, the focus should be on formative assessments. The question whether teachers are assessing what students actually had a chance to learn, directly relates to the concept of Constructive Alignment. The importance of measuring the process of student's work instead of just expecting the right answer was also highlighted: teachers should also look at intermediate steps and evaluate the process as such.

Most attendees agreed that a focus on the value of the assessment for the learner is of high importance while designing assessments. This could be achieved by using assessments and contexts which are relevant to the students' personal or professional life. Observational techniques, including time tracking, could be used to measure the students' workload, the dynamics of motivation, and – if applicable – the point of giving up.

Regarding programming assessment, there was a wide consensus that coding itself is only one of many important skills for students to learn. The ability to explain concepts to get to deep understanding as well as the ability to figure out simplest ways to test a program, an algorithm, or a component for correctness should also be considered a target for assessments.

Since developing soft skills like communication with colleagues and experts of different fields are very important for computer scientists and programmers, several attendants highlighted the importance of assessing these skills. Collaborative work, e.g., in programming, might be used to achieve this even in exams.

It was agreed upon that there is a high demand for validated assessment tools across all areas of computer science education research.

5.2 Next Steps Discussion

Notes by Mirko Westermeier

License  Creative Commons BY 3.0 Unported license
© Notes by Mirko Westermeier

In the closing session, several topics regarding the seminar and research implications were discussed.

The attendees considered this first Dagstuhl seminar on computer science education research a most valuable addition to the current conferences, workshops, and meetings. Bringing people together and having time for focused work was appreciated as well as the informal meeting format which was used during the seminar several times. For example, some groups used a discussion format in which they concentrated on one researcher's work at a time. It was agreed upon that having mentoring workshops at all computer science education research venues would help to stay connected.

The attendees liked that the atmosphere in all breakout groups were very open and open to critique. To make it easier for newcomers to the computer science education research community to dive into it and get to know researchers with similar topic, Andrew J. Ko offered to write a document about the internal structure of the community and their body of knowledge (after the seminar, the participants provided feedback on the first draft which then was made available online; see <https://faculty.washington.edu/ajko/cer>).

The seminar was seen as a step towards formalizing research agendas in a way that they can be communicated to a broader audience. In addition to the focus of the current seminar, K–12 computer science education research should be considered in a similar setting.

Finally, several groups of attendees agreed about further collaborative work to continue discussion and work on the research questions that arose during this week.

6 Poster abstracts

6.1 Programming Education for Novices

Michael E. Caspersen (Aarhus University, DK)

License  Creative Commons BY 3.0 Unported license
© Michael E. Caspersen

I am interested in programming education for novices. Previously, my interest concentrated on higher education, but with the increased focus on computing and programming in secondary (and primary) education, it becomes relevant to rethink purpose, learning goals and didactics of programming education with these specific target groups in mind. In higher computing education, the perspective is education of to-be experts whereas in secondary (and even more so in primary) education, the purpose is computing, computational thinking and programming as general education and 'building'. In particular, the need for (pre- and in-service) training of teachers for primary and secondary school requires development of an associated computing and programming didactics.

6.2 KETTI – Competence Development of Student Teaching Assistants in Computer Science

Holger Danielsiek (Universität Münster, DE)

License © Creative Commons BY 3.0 Unported license

© Holger Danielsiek

Joint work of Hubwieser, Peter; Krugel, Johannes; Magenheimer, Johannes; Ohrndorf, Laura; Ossenschmidt, Daniel; Schaper, Niclas; Vahrenhold, Jan

The KETTI-project is a research cluster focused on developing a competence model for undergraduate teaching assistants in computer science along with corresponding instruments. As part of the deliverables of the project, training modules aligned with the desired competencies will be made available for academic use. As discussed in the introduction the focus of KETTI are first-year courses in computer science. KETTI is funded by the BMBF and brings together researchers from computer science education, computer science, and psychology. The core of the project consists of four research groups at three large public German research universities (Technical University of Munich, Westfälische Wilhelms-Universität Münster, and University of Paderborn). These universities cover a wide range of student audiences. All departments have offered formal UTA training courses in the past.

To broaden the scope even further, KETTI includes six associated partners at five other German computer science departments and two associated partners at two German mathematics departments. The associated partners at the computer science departments have not yet established formal UTA training courses. These partners are consulted on a regular basis to ensure that assumptions about student and UTA populations are not biased towards the departments with a tradition of UTA training. The partners at the mathematics departments are consulted to enable transfer of knowledge between disciplines and to better contrast general requirements and requirements specific to computer science.

6.3 A Method to Analyse Computer Science Students' Teamwork in Online Collaborative Learning Environments

Katrina Falkner (University of Adelaide, AU)

License © Creative Commons BY 3.0 Unported license

© Katrina Falkner


Joint work of Vivian, Rebecca; Falkner, Nickolas; Tarmazdi, Hamid

Although teamwork has been identified as an essential skill for Computer Science (CS) graduates, these skills are identified as lacking by industry employers, which suggests a need for more proactive measures to teach and assess teamwork. In one CS course, students worked in teams to create a wiki solution to problem-based questions. Through a case-study approach, we test a developed teamwork framework, using manual content analysis and sentiment analysis, to determine if the framework can provide insight into students' teamwork behavior and to determine if the wiki task encouraged students to collaborate, share knowledge, and self-adopt teamwork roles. Analysis revealed the identification of both active and cohesive teams, disengaged students, and particular roles and behaviors that were lacking. Furthermore, sentiment analysis revealed that teams moved through positive and negative emotions over the course of developing their solution, toward satisfaction. The findings demonstrate the value of the detailed analysis of online teamwork. However, we propose the need for automated measures that provide real-time feedback to assist educators in the fair

and efficient assessment of teamwork. We present a prototype system and recommendations, based on our analysis, for automated teamwork analysis tools.

6.4 Plan Composition

Kathi Fisler (Worcester Polytechnic Institute, US)

License  Creative Commons BY 3.0 Unported license
© Kathi Fisler

Several decades ago, researchers identified “plan composition” as a programming-related task that students struggle to perform well. We have conducted a series of studies that revisit planning studies, most recently attempting to contrast how students using different programming languages (some functional, some procedural) approach similar problems. We discuss our efforts to modernize planning studies to contexts framed by lightweight scripting for data analysis. Our results show the role that libraries and built-ins play in guiding how students decompose problems, and identify potential student misconceptions about the costs associated with using built-in operations. Overall, our goal is to help reframe research questions about planning for today’s programming contexts.

6.5 Critiquing CS Assessment from a CS for All Lens

Mark Guzdial (Georgia Institute of Technology – Atlanta, US)

License  Creative Commons BY 3.0 Unported license
© Mark Guzdial

Not everyone who learns CS is going to want to be a software engineer. Then why teach them CS? Perhaps there are reasons to learn CS that are not about software development. As computing pervades many professions, there are reasons to learn to program that are relevant to those professions, those communities of practice, that may not be relevant to software engineers. For example, computational scientists and engineers may not care about developing code to professional standards, since their programs may not be meant to be used beyond a single time. If we have different learning outcomes for different communities of practice, assessment has to change, too. I argue that we have to consider what the learner wants to do and wants to be (i.e., their desired Community of Practice) when assessing learning. Different CoP, different outcomes, different assessments.

6.6 Communicating through Sketches and Diagrams

Geoffrey L. Herman (University of Illinois – Urbana Champaign, US)

License  Creative Commons BY 3.0 Unported license
© Geoffrey L. Herman

Communicating through sketches and diagrams is a foundational skill in students’ problem solving, yet little is known about how or why students use sketches and or how they learn about them. We are exploring students’ representational fluency by exploring the similarities and distinctions between experts and novices in their use and production of sketches during

problem solving. We are conducting clinical interviews in two domains: digital logic and trusses. While digital logic sketches are visually distinct from real-world implementations, trusses rely on fascimiles of real-world objects. We will share preliminary findings about students' use of sketching in digital logic contexts.

6.7 Neo-Piagetian Theory and the Novice programmer

Raymond Lister (University of Technology – Sydney, AU)

License © Creative Commons BY 3.0 Unported license
© Raymond Lister

Piagetian-based cognitive development theories (Flavell, 1977; Morra, Gobbo, Marini, & Sheese, 2007; Piaget, 1952) provide a framework for describing the domain-specific development of cognition. In the case of programming, students exhibit characteristics at each of the neo-Piagetian sensorimotor, preoperational, and concrete operational stages. At the least mature stage of cognitive development, the sensorimotor stage, a novice programmer has difficulty tracing (i.e. manually executing) code. These students tend to see code as an ad hoc collection of tricks. At the next more mature stage, a preoperational novice has a more systematic grasp of programming, and can trace code with some accuracy. However, a preoperational programmer tends not to abstract from the code itself. Instead, the preoperational programmer favours inductive inference. That is, the preoperational programmer tends to infer what a piece of code does from input/output pairs, which are found by tracing the code. It is only at the third stage, the concrete operational stage, that the novice programmer begins to reason about abstractions of code, simply by reading the code. A defining characteristic of the concrete operational stage is the ability to reason about the concepts of conservation, reversibility and transitive inference. Neo-Piagetian theory emphasises that developmentally appropriate exposure to the domain of knowledge is paramount to the progression between developmental stages of reasoning. Unfortunately, contemporary teaching methods for programming assume that students begin at the concrete operational level.

6.8 Educational Data Mining

Andreas Mühlring (TU München, DE)

License © Creative Commons BY 3.0 Unported license
© Andreas Mühlring

Educational Data Mining allows to search for pattern in large amounts of data that are not directly observable. As such, the methods are often exploratory in nature and therefore well suited for the context of CSEd where an accepted body of fundamental results concerning the teaching and learning of the subject is still not available. A method of aggregating concept map data has been used to investigate the effects of CS education on the structural configuration of knowledge in learners. The results show that artifacts of a curriculum can be traced to the knowledge structures of learners. Also, the large scale Bebras contest data has been used investigate the underlying psychometric structure of the items. Here, the results indicate that success in the contest is determined to a large extent by visuo-spatial thinking.

6.9 Teaching and Learning a First Programming Language

Anthony Robins (University of Otago, NZ)


License  Creative Commons BY 3.0 Unported license
© Anthony Robins

I am interested in novices learning a first programming language, in particular why they fail and why they succeed in a first programming course (CS1). I believe that the well explored distinction between novice and expert programmers is less important than the distinction between ineffective and effective novices. I argue that the early stages of learning in CS1 are crucial. Initial success or failure builds rapidly on itself to create momentum towards a final successful or unsuccessful outcome (the “learning-edge momentum” effect described in Robins, 2010).

It is particularly important to understand the process of novice learning as programming and related computational thinking topics are increasingly being introduced into the school curriculum, even elementary/primary school, in several countries around the world.

6.10 Novice Programmers’ Difficulties with Program Dynamics and Misconceptions of the so-called Notional Machine

Juha Sorva (Aalto University, FI)

License  Creative Commons BY 3.0 Unported license
© Juha Sorva

In this poster, I present an overview of some of the main themes of my research over the past five years. These themes include: novice programmers’ difficulties with program dynamics and misconceptions of the so-called notional machine; interactive program visualization for education; the research-based instructional design of interactive electronic textbooks for computing education, and the benchmarking of students conceptual knowledge and programming skill across institutions.

6.11 Qualitative Feedback of Program Code and Programs


Martijn Stegeman (University of Amsterdam, NL)

License  Creative Commons BY 3.0 Unported license
© Martijn Stegeman

Martijn Stegeman is finalising plans for a research project on the use of feedback in programming courses. During these courses, it is common to provide qualitative feedback on the program code as well as on the resulting programs, while students (hopefully) use this feedback to better understand quality norms and improve on new programs they write. Interestingly, the norms used by teachers are usually very personal and subjective, although there appears to be a lot of commonality. This project aims to produce understanding of the norms in use, and the way feedback is integrated into programming courses.

6.12 Potential Factors Indicating Success in an Algorithms and Data Structures Course

Jan Vahrenhold (Universität Münster, DE)

License  Creative Commons BY 3.0 Unported license
© Jan Vahrenhold

Joint work of Danielsiek, Holger

We report on first steps towards identifying factors indicating students' performance in an Algorithms and Data Structure course. We discuss a study undertaken to investigate the predictive and explanation power as well as the limits of weekly test items based on concept inventory questions, homework grades, and performance in a preceding CS1 course. We relate our findings for two subgroups to results on academic success in general and performance in a CS1 course in particular.

6.13 (Mis)conceptions Surrounding Exponential Growth in the Foundations of Computing Concept Inventory

Steven A. Wolfman (University of British Columbia – Vancouver, CA)

License  Creative Commons BY 3.0 Unported license
© Steven A. Wolfman

Exponential growth commonly appears as a topic in foundations of computing textbooks and exams. In our interviews with foundations of computing course instructors, about half mentioned exponential growth as an important but tricky topic for students. In our think-aloud interviews with students working problems not directly related to exponential growth, students nonetheless often expressed confusion around or simply misused the concept. As a result, we gathered and analyzed data on students' definitions of “exponential growth” from an open-ended question on a low-stakes assessment. Then, we designed two closed-ended (multiple choice) concept inventory questions on exponential growth based on this data, instructors' and students' comments in interviews, and our own experience. We performed think-aloud interviews with students working these closed-ended questions and administered the closed-ended questions to several cohorts of students (and several hundred students) in our three required foundations of computing courses, adapting the questions in response to strengths and weaknesses in the distractors exposed by both think-aloud and simple multiple-choice responses. The data show that students have persistent and distressing misconceptions around the concept of exponential growth, with the three most resilient distractors showing a graph curving “up and to the right”, a function described as growing “more and more quickly”, and the simple quadratic function n^2 . Further research will be needed to connect these misconceptions to any impact on students' ability to judge and select appropriate approaches to computational problems.

Participants

- Michael E. Caspersen
Aarhus University, DK
- Holger Danielsiek
Universität Münster, DE
- Brian Dorn
University of Nebraska, US
- Katrina Falkner
University of Adelaide, AU
- Sally Fincher
University of Kent, GB
- Kathi Fisler
Worcester Polytechnic Inst., US
- Mark Guzdial
Georgia Institute of Technology –
Atlanta, US
- Geoffrey L. Herman
University of Illinois – Urbana
Champaign, US
- Lisa C. Kaczmarczyk
San Diego, US
- Andrew J. Ko
University of Washington –
Seattle, US
- Michael Kölling
University of Kent, GB
- Shriram Krishnamurthi
Brown Univ. – Providence, US
- Raymond Lister
University of Technology –
Sydney, AU
- Briana Morrison
Georgia Institute of Technology –
Atlanta, US
- Jan Erik Moström
University of Umeå, SE
- Andreas Mühling
TU München, DE
- Anthony Robins
University of Otago, NZ
- Rolf Schulmeister
Universität Hamburg, DE
- Carsten Schulte
FU Berlin, DE
- R. Benjamin Shapiro
Univ. of Colorado – Boulder, US
- Beth Simon
University of California – San
Diego, US
- Juha Sorva
Aalto University, FI
- Martijn Stegeman
University of Amsterdam, NL
- Heike Theyssen
Universität Duisburg-Essen, DE
- Jan Vahrenhold
Universität Münster, DE
- Mirko Westermeier
Universität Münster, DE
- Steven A. Wolfman
University of British Columbia –
Vancouver, CA

