



Aalto University
School of Science

Using automatic and semi-automatic feedback to support students's learning process

Lauri Malmi

Aalto University

Department of Computer Science

26.1.2016

Contents

- Some background
 - Some technical solutions and results / experiences of the following:
 - Automated feedback
 - Visualizations
 - Interactive tasks
 - Support for better human feedback
 - References
-

Challenges

- The roots of Learning + Technology research group (LeTech) originate in the intensive work for improving education in very large programming, data structures and algorithms courses at Helsinki University of Technology, started in early 1990's.
 - How to **engage and activate** students in large courses (200 - 1000+) students, where human resources for guidance and assessment are limited?
 - How to provide them **personal feedback** on their work?
 - Tools for automatic assessment and visual interaction
 - Not all students are learning oriented.
 - Main context: programming education, but many results are context-free.
-

Principles

- Students must be active during the whole course.
 - Weekly / biweekly **compulsory** assignments to encourage learning and gradual building of competences (*activation*)
 - Providing model solutions is not good enough. Personal **feedback on their own** solutions is needed (*supporting reflection*)
 - Personal **feedback must be received soon**, not 1-2 weeks later (*supporting reflection*).
 - Assignments must have a **significant effect** on the final grade (*motivation to do the work*).
-

Some solutions

- Automated feedback
- Visual simulation tasks
- Gamification & progress feedback
- Support for better human feedback

Automated feedback

- Note: We are not talking about MCQs
 - Possible when analyzing submissions which are presented using a formal representation, e.g.,
 - Programming assignments (Goblin, Web-CAT)
 - Algorithm simulation tasks (Malmi et al. 2004, Karavirta & Shaffer, 2013, OpenDSA)
 - Program simulation tasks (Uuhistle, Sorva 2012)
 - Database queries
 - Other: Mathematical formalisms, physics calculations (STACK)
-

Automatic feedback on programs

- Could be given on
 - Program correctness (automatic tests)
 - Quality of students' own testing
 - Program structure
 - Use of required / forbidden language features
 - Program runtime effectiveness
 - Required algorithms (prototype only)
-

Visual simulation tasks

- Computing involves many abstract and intangible concepts and processes.
 - Visualizations can help to concretize these concepts and processes.
 - Viewing visualizations is not enough but students have to be engaged with them (Hundhausen et al. 2002)
 - Students often do not concentrate when viewing animations (Sirkiä, Sorva 2015).
 - Interactive simulation tasks with instant feedback allow students to reflect on and tune their mental model of the concept or process.
 - Demo
 - JSAV algorithm simulation tasks and A+ environment
-

Some results and experiences

- Immediate feedback supports students' reflection on the task while engaged with it.
 - Resubmissions should have a smallish limit, unless the basic task can be parameterized and provided as new variation every time (Malmi et al. 2004, 2005)
 - Some basic tasks can be fully automated and therefore human tutoring efforts can be directed to guiding more advanced assignments. (Malmi et al. 2005)
 - Resubmissions may promote trial-and-error working for some students. (Karavirta et al. 2006; Auvinen, 2015)
 - Many students aim at full points, when not needed for the best grade (Malmi et al. 2005)
 - Some students do not test their programs themselves but use automatic testing only (Edwards, 2003)
-

Program construction exercises

- Parsons problems present a simplified program construction exercise: program puzzle.
- js-Parsons implement a 2D Python puzzle.
- Demo

Trace analysis

- Analysing the trace allows to investigate the programming process, not only the final submission. (Helminen et al. 2012)

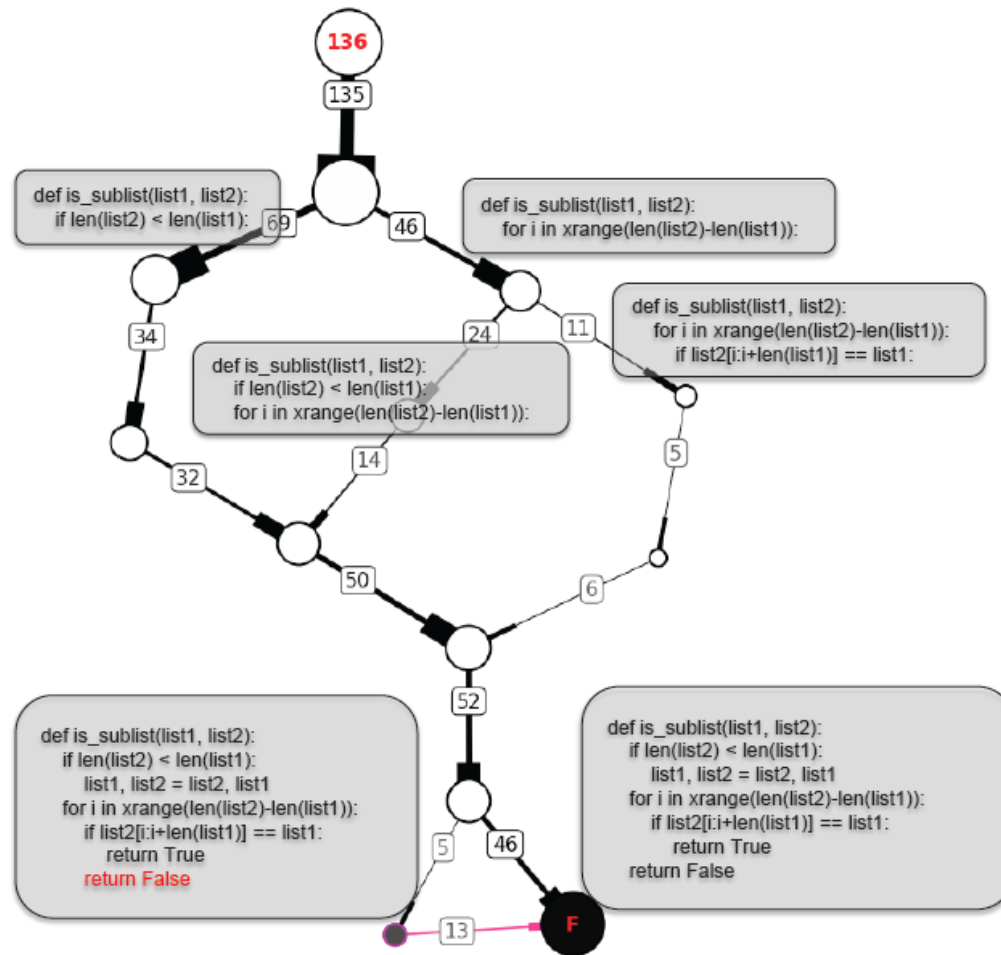
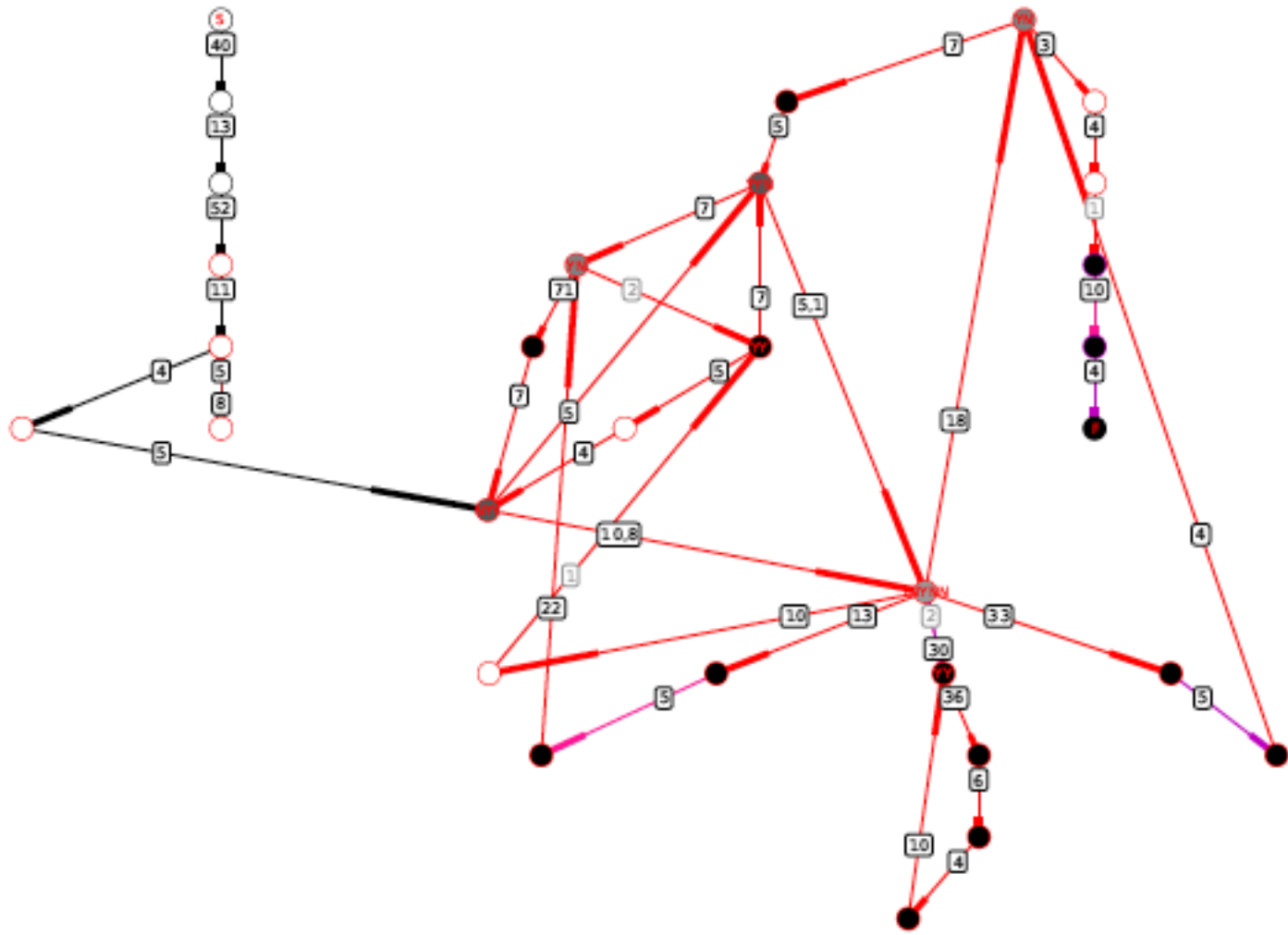





Figure 2: Solution strategies of P5 filtered so that transitions performed by less than five students not visualized. The resulted graph covers 29 % of transitions of all students and 27% of all solution paths.



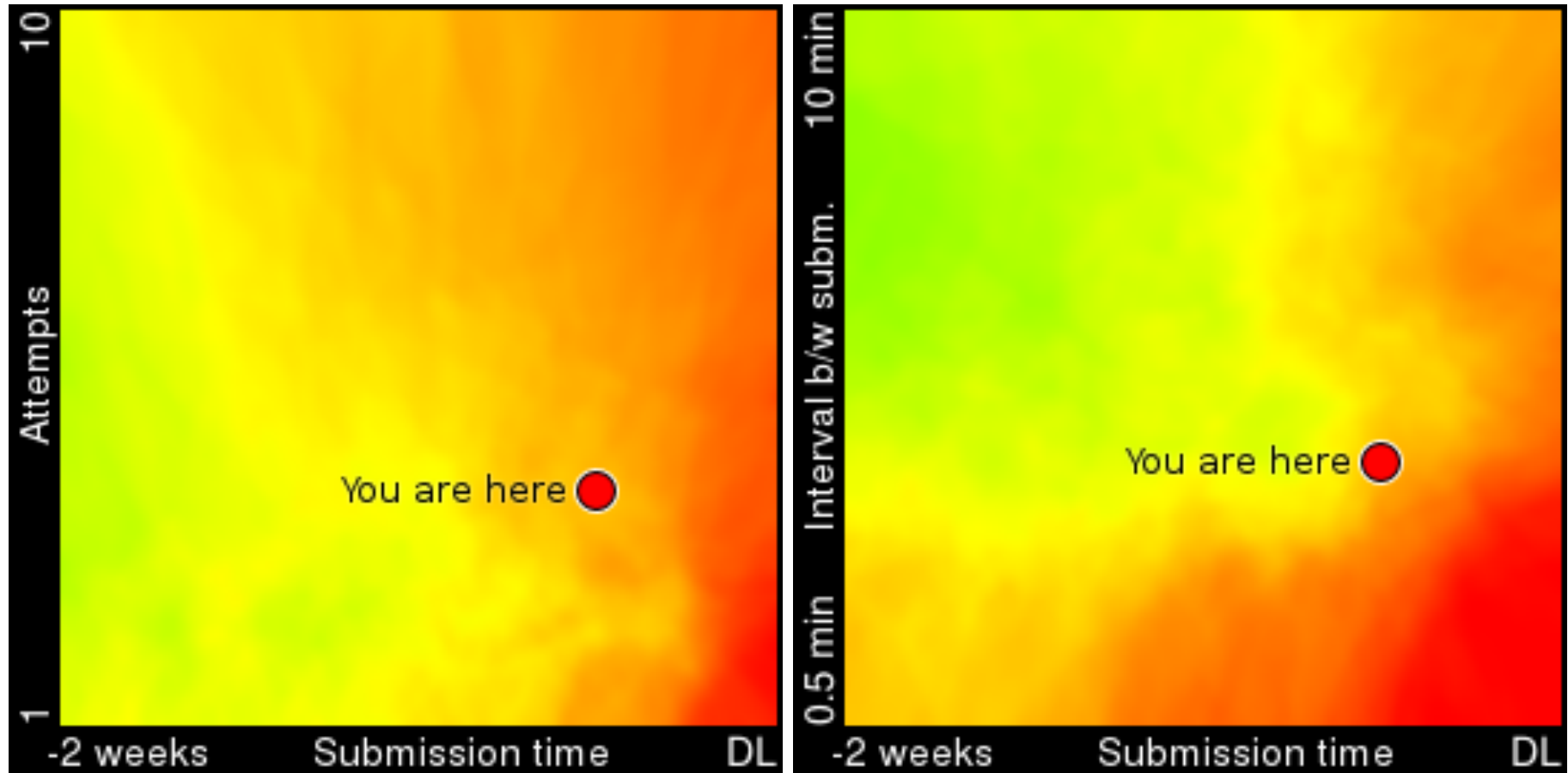
Feedback on study process

- Some students have bad or inefficient study habits, like procrastination and trial-and-error. (Auvinen, 2015)
- Monitoring study process and targeted feedback could help here.

Achievement badges

| Category | Criteria | Value of X (B, S, G) | Example |
|-----------------|---|----------------------|---|
| Time management | 50% of points of the round earned at least X day(s) before the deadline | 1, 3, 7 |  |
| Learning | Round completed with X% score | 50, 75, 100 |  |
| Carefulness | X assignments with perfect score using only one attempt | 5, 10, 15 |  |

Heatmaps



Results

- Intervention studies with gamified / non-gamified feedback
 - Achievement badges
 - Heatmaps
 - Students with *performance orientation* respond positively to achievement badges while students with *performance-avoidance orientation* take advantage of neutral visualizations
 - Effects were, unfortunately, visible only with high-performing students. Low-performing students still struggle.
 - (Auvinen, Hakulinen, Malmi 2015)
-

Tailored human feedback

- Automatic assessment of project reports, essays and exams is not yet available.
- Manual grading is slow and feedback often poor due
- Multiple assistants grading the same tasks may result in inconsistencies in grading and level of feedback.

Rubyruc

- Rubrics-based grading (Auvinen, 2011)
 - Ready-made but editable feedback phrases
 - Fully tailorable (rubrics, points, phrases)
 - Exam grading by annotating scanned pdfs of exam papers.
 - Demo (Rubyruc)
-

2) a) Abstrakti tietotyyppi on tietorakenne, jolle on määritelty tietyt ominaisuudet ja sille voidaan suorittaa tiettyjä määriteltyjä operaatioita. Sen tarkempaan toteutukseen ei oteta kantaa, vaan se voidaan toteuttaa useamman FDT:n (fundamental data type) avulla. Esim. prioriteettijono.

b) Prioriteettijono on abstrakti tietotyyppi, jolle on määritelty seuraavat operaatiot; siitä on helppo löytää ja palauttaa prioriteetti-arvoita suurin/pienin arvoin
 Insert: lisätään uusi arvoin
 Find min/max: palautetaan ja poistetaan suurin/pienin prioriteettiarvo oleva arvoin

lin / max = sama kuin edellinen, mutta poistamatta.

1 Määritelmä OK
 2. Terminologiaa / a. Abstrakti tietotyyppi ✕

1 Esimerkki OK
 • Esimerkki abstraktista tietotyypistä voisi olla hakurakenne, jolle on määritelty operaatiot lisää(avain, arvo), hae(avain), poista(avain), jne. Hakurakenteen voi sitten toteuttaa eri tavoin, esim. hajautustaululla tai hakupuulla.
 2. Terminologiaa / a. Abstrakti tietotyyppi ✕

1 Määritelmä OK
 2. Terminologiaa / b. Prioriteettijono ✕

1 Esimerkki OK
 2. Terminologiaa / b. Prioriteettijono ✕

a. Abstrakti tietotyyppi 1

2
 1 Määritelmä OK
 1 Esimerkki OK
 Abstraktille tietotyypille on määritelty semanttikka, eli mitä tieto tarkoittaa, ja operaatiot, eli mitä tiedolle voi tehdä.
 Esimerkki abstraktista tietotyypistä voisi olla hakurakenne, jolle on määritelty operaatiot lisää(avain, arvo), hae(avain), poista(avain), jne. Hakurakenteen voi sitten toteuttaa eri tavoin, esim. hajautustaululla tai hakupuulla.

b. Prioriteettijono 2

2
 1 Määritelmä OK
 1 Esimerkki OK
 Prioriteettijono on abstrakti tietotyyppi, jolle on määritelty operaatiot 1) lisää ja 2) prioriteettiaan suurimman (tai pienimmän) alkion poisto.
 Esimerkki voisi olla minimikeko.

c. Binaärikeko

2
 1 Määritelmä OK
 1 Esimerkki OK
 Binaärikeko on binaäripuun avulla toteutettu keko. Puussa kunkin solmun prioriteetti-arvo on suurempi (tai minimikeossa pienempi) kuin sen lasten prioriteetti-arvot.
 Hakupuun on eri asia kuin keko. Siinä solmun vasen lapsi on pienempi ja oikea lapsi suurempi kuin isäsolmu.
 Esimerkiksi käy pilirros binaärikeosta.

d. Kekoehto

2
 1 Määritelmä OK
 1 Esimerkki OK
 Kekoehto määrittelee keon alkioiden järjestyksen.
 Esimerkki: Isä suurempi kuin lapsensa.

Points:

Next page →

References

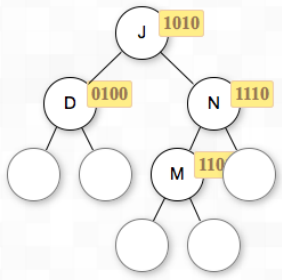

1. Auvinen, Tapio. "Rubyric." Proceedings of the 11th Koli Calling International Conference on Computing Education Research. ACM, 2011.
2. Tapio Auvinen. Harmful Study Habits in Online Learning Environments with Automatic Assessment. Accepted for publication in *Proceedings of the 2015 International Conference on Teaching and Learning in Computing and Engineering (LaTICE)*, Taipei, Taiwan, April 2015.
3. Tapio Auvinen, Lasse Hakulinen, and Lauri Malmi. Increasing Students' Awareness of their Behavior in Online Learning Environments with Visualizations and Achievement Badges, *IEEE Transactions on Learning Technologies*, vol 8 no 3, pp- 261-273.
4. Edwards, Stephen H. "Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance." Proceedings of the International Conference on Education and Information Systems: Technologies and Applications EISTA. Vol. 3. 2003.
5. Lasse Hakulinen and Tapio Auvinen. The Effect of Gamification on Students with Different Achievement Goal Orientations. In Proceedings of the 2014 International Conference on Teaching and Learning in Computing and Engineering, Kuching, Malaysia, pages 9–16, April 2014.
6. Juha Helminen, Petri Ihanola, Ville Karavirta, and Lauri Malmi. 2012. How do students solve parsons programming problems?: an analysis of interaction traces. In *Proceedings of the ninth annual international conference on International computing education research (ICER '12)*. ACM, New York, NY, USA, 119-126.
7. C. Hundhausen, S. Douglas, and J. T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290, 2002.
8. Ville Karavirta and Clifford A. Shaffer. 2013. JSAV: the JavaScript algorithm visualization library. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education (ITICSE '13)*. ACM, New York, NY, USA, 159-164.
9. V. Karavirta, A. Korhonen, L. Malmi, On the use of resubmissions in Automatic Assessment systems. *Computer Science Education*, vol 16 no 3, 2006, pp. 229-240.
10. L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, O. Seppälä, P. Silvasti: Visual Algorithm Simulation Exercise System with Automatic Assessment: TRAKLA2. *Informatics in Education*, vol 3 no 2, 2004, pp. 267-288.
11. L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, Experiences on Automatically Assessed Algorithm Simulation Exercises with Different Resubmission Policies. *ACM Journal of Educational Resources in Computing*, vol 5 no 3, Article 7, 2005.
12. Sirkiä, T., & Sorva, J. (2015, July). How Do Students Use Program Visualizations within an Interactive Ebook?. In Proceedings of the eleventh annual International Conference on International Computing Education Research (pp. 179-188). ACM.
13. J. Sorva: Visual program simulation in introductory programming education, Doctoral Dissertation, Aalto University 2012.

Systems

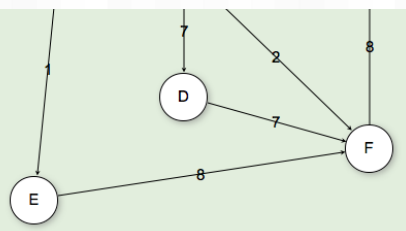
1. OpenDSA project home page: [Algoviz.org/
OpenDSA](http://Algoviz.org/OpenDSA)
 2. Rubyric, <https://rubyric.cs.hut.fi/> or [https://
rubyric.com/](https://rubyric.com/)
 3. STACK: System for Teaching and Assessment using a Computer algebra Kernel: <http://stack.bham.ac.uk/>
 4. Web-CAT home page: <http://web-cat.org/>
-

Thank you

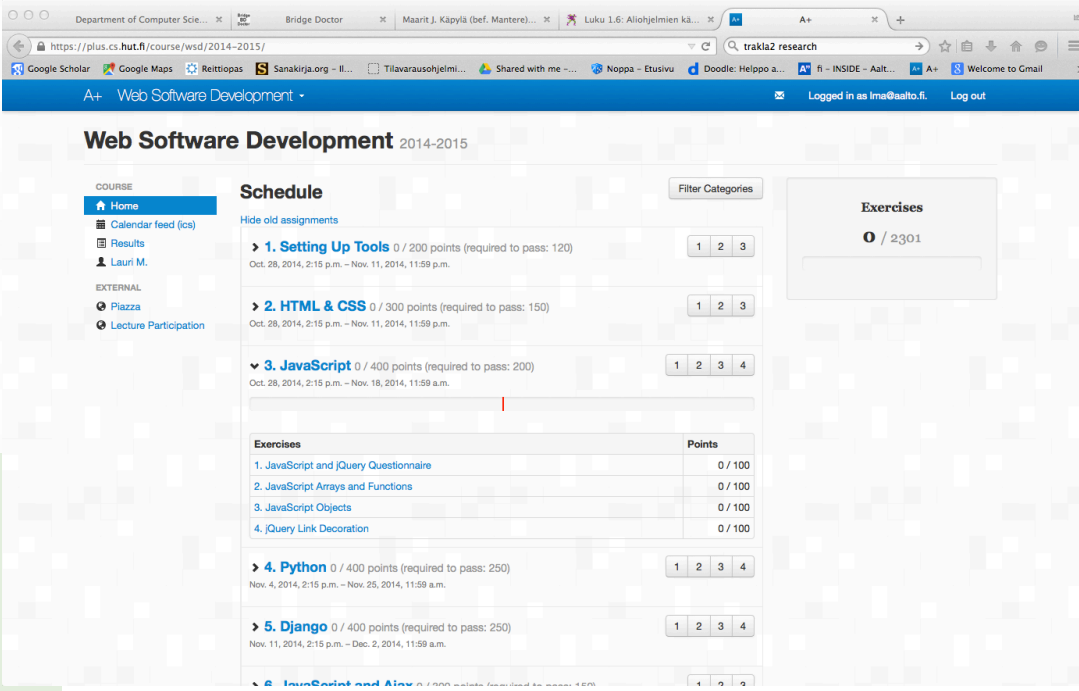
Kumoa Uudelleen Mallivastaus Arvostelee



Instruction
Reprodu
Graph N



| | |
|---|----------|
| B | Infinity |
| C | Infinity |
| D | Infinity |
| E | Infinity |
| F | Infinity |



Learning + Technology group

http://cse.aalto.fi/en/research/learning_technology/