New Tools and Practices for Online Collaboration in Teaching, Learning, and Research of Programming Languages

William E. Byrd
University of Utah
Will.Byrd@utah.edu

For the past three years I have been experimenting intensively with teaching programming and programming languages online. In contrast with Massive Open Online Courses (MOOCs), such as those offered by EdX and Coursera, I have been exploring more interactive models of online learning, closer to mentorship and apprenticeship than to traditional classroom models. I have also begun collaborating on programming language research with an ad hoc but very talented, creative, and energetic group of computer scientists and programmers from around the world, including several productive collaborators who wish to remain partially or completely anonymous. And I have begun experimenting with talks and tutorials on programming languages using real-time audience interaction, in which the audience can immediately interact with the code I am writing in real-time, modify and experiment with this code immediately, and even push back changes and interesting or confusing example queries for me to examine during the talk.

Based on these experiments, and from the experience of others running MOOCs or experimenting with streaming their work online, I am convinced that:

- current tools for online collaboration and learning are inadequate for more interactive, ad hoc, and potentially long-term forms of learning, mentorship, and research;
- programming languages researchers and educators should be especially excited about exploring online approaches to deep collaboration, since unlike in biology (for example) there are few if any research and teaching tasks we cannot perform entirely online;
- fully harnessing the power of ad hoc online collaboration and learning is difficult but possible, and will require inventing new models of collaboration and academic practices;
- and finally, programming languages researchers are exactly the people who should be developing improved tools for these collaborations, since such tools will probably incorporate compilers, interpreters, read-eval-print loops (REPLs), immutable data structures, and many other topics closely related to programming languages.

In this talk I will give an overview of my experiments, talk about ongoing efforts to develop better tools for online collaboration and learning, and explain how programming languages researchers can contribute to the development of such tools, and benefit from using those tools.

A Few of the Many Experiments I Have Tried

- I have taught a 40-hour-long informal, highly interactive "uncourse" on logic programming using Google Hangouts on Air/YouTube and GitHub/Gist, with over a dozen participants, who often took turns "driving" by giving demos, livecoding, asking questions, mob programming, etc.
  `https://www.youtube.com/playlist?list=PLO4TbomOdn2cks2n5PvifialL8kQwt0aW`

- I do a large portion of my research on logic programming online. For example, I am working with Rob Zinkov at Indiana University on probabilistic logic programming. We've done extensive pair programming using Google Hangouts with screensharing, GitHub, Etherpad, Jupyter Notebooks, etc.
- I am currently working with several programmers, using Google Hangouts on Air, to develop new software for interactive programming and collaboration. (See `http://webyrd.net/thunk.html` and `https://www.youtube.com/playlist?list=PLO4TbomOdn2ejAyIxWcLOq5c7G4uCeIwp`)
- I have been using Google Hangouts and Skype for online mentoring and collaboration with people interested in learning logic programming, and for collaboratively exploring interesting areas of programming. I met many of these people through Twitter, or from their watching my YouTube videos.

A More Interactive Model of Online Learning

In contrast with a MOOC, I'm interested in a much more decentralized way of learning, in which various learners take turns "driving" the discussion, giving demos, livecoding, posing problems, etc. Even if I am "driving," real-time chat and voice communication, along with IRC and Twitter, allow me to immediately answer questions, facilitates mob programming, etc. Later hangouts of the 40-hour miniKanren uncourse show some of this interactivity: `https://www.youtube.com/playlist?list=PLO4TbomOdn2cks2n5PvifialL8kQwtOaW`

Unfortunately, tool limitations made it difficult for the miniKanren uncourse to be as interactive as I would have liked. For example, I had originally intended for arbitrary participants to seamlessly switch roles (between mob programming collaborator to driver, for example). However, given the frequent crashes of Google Hangouts on Air, problems with microphones and muting, screen resolution problems, etc., switching between participants (or even switching between video and screen sharing for one participant) became a risky and time-consuming proposition. This significantly hampered the interactivity of the course.

Other Inadequacies of Existing Tools

Google Hangouts on Air is also highly susceptible to disruptive trolls (which I found to be a constant problem), only supports 10 participants, has an interface that changes unpredictably and arbitrarily, doesn't record text chat in the video recording pushed to YouTube, often crashes when trying to screen share, doesn't support multiple participants sharing the same screen, and has many other limitations when it comes to online learning and collaboration. Skype and all the other tools I have tried have at least as many problems. This isn't surprising, since these tools were never designed for programmers to collaborate online.

To be clear, I find Hangouts on Air extremely useful, and don't mean to pick on it in particular. All of the other tools I have used (GitHub, Gist, Etherpad, etc.) also have their limitations for online ad hoc collaboration and learning. My point is that much better tools could be (and should be) designed specifically for collaborative research and teaching in computer science and programming languages. Since collaborators want to share, edit, and run code, programming languages researchers and implementors are in a perfect position to contribute to these tools.

New Academic Practices & Further Experiments

Is it possible for ad hoc groups of collaborators to do innovative programming languages research online? I believe so. And I believe that high-quality academic papers could be written by such groups. This brings up interesting questions. Who would be the co-authors on the paper? What if a co-author wished to remain anonymous—could their Twitter handle be used for author attribution? If all of the work were to be done live, and recorded as YouTube videos, would this preclude submission to double-blind conferences?

I strongly suspect we will see such collaborations in the near future, in programming languages, computer science, and other academic disciplines. Indeed, I would be very interested in trying some of these experiments with other OBT attendees, and with other people from around the world.