Eos, Vol. 92, No. 50, 13 December 2011

FORUM

Computer Programing for Geosciences: Teach Your Students How to Make Tools

PAGES 469–470

When I announced my intention to pursue a Ph.D. in geophysics, some people gave me confused looks, because I was working on a master's degree in computer science at the time. My friends, like many incoming geoscience graduate students, have trouble linking these two fields. From my perspective, it is pretty straightforward: Much of geoscience evolves around novel analyses of large data sets that require custom tools—computer programs—to minimize the drudgery of manual data handling; other disciplines share this characteristic.

While most faculty adapted to the need for tool development quite naturally, as they grew up around computer terminal interfaces, incoming graduate students lack intuitive understanding of programing concepts such as generalization and automation. I believe the major cause is the intuitive graphical user interfaces of modern operating systems and applications, which isolate the user from all technical details. Generally, current curricula do not recognize this gap between user and machine. For students to operate effectively, they require specialized courses teaching them the skills they need to make tools that operate on particular data sets and solve their specific problems. Courses in computer science departments are aimed at a different audience and are of limited help.

In 2009, my adviser, Jeff Freymueller, and I began to experiment with a course on programing for geoscience graduate students in our department at the University of Alaska Fairbanks. This emerged from a fortunate mix of people in one room: a graduate student in need; me, already thinking about such a course; and supportive and aware faculty. We now have gone through three iterations of this experiment. Our course goals are ambitious for a one-semester, two-credit course. We learned a lot from our many mistakes, and I want to share some of our experiences and encourage other institutions to follow along. Specific programing languages and tools vary by discipline and department, but the general ideas from our course could be applied widely. The overarching main points we believe such a course should touch on are as follows:

Repetitive work is for machines. Students need to realize that a problem is worth being solved once. Exactly once. Yet there are students manually laboring through identical procedures on a daily basis. We want them to understand that breaking down a complex problem into simple tasks, writing out the respective steps, testing them individually, and finally bundling them into one command is of great value and is time well invested. From this we advance to generalizing specific solutions such that their tool tackles an array of problems. For example, suppose one has a tool that analyzes a day's worth of data for one sensor. We want students to ask how this tool can be used to treat all available sensors on all days. Trying to think of such a configuration is a worthwhile yet challenging exercise. The solution to questions like this is abstract and entirely free of code, but it establishes the fundamental concept of having computers do the work while you are out for an afternoon run.

Understand fundamental principles. No single programing language is the ultimate tool for all problems. Handing your students one tool to solve a specific task will be a great quick fix until a different kind of problem emerges, rendering this tool a poor fit. Exposing students to a small variety of programing languages and the connecting fundamental principles loosens the tension a new syntax brings and hands them abilities they crave. Comprehension of the concepts of variables, functions, and flow control gives students sufficient momentum and the ability to transition to whatever shiny new language comes around in the future. While object-oriented programing certainly deserves consideration because it enables wonderful software design, it seems impossible to teach such advanced concepts well in a few lectures and labs, so we decided against including it in our course.

Organize data consistently. Data-related programing revolves around traversing directories, picking files, reading data, processing data, and writing out results. To have a computer operate effectively and keep coding efforts under control, a consistent naming scheme for files and directories is crucial. Imagine needing all available data for 23 May 2012. It's easy if all files carry the date in their name in a consistent format, say, 20120523. Consistent data archiving allows your program to find files in a minimal number of steps. Admittedly, this is pretty straightforward, but students are so accustomed to the fact that they can easily recognize a multitude of date formats that they do not realize how hard it is for a machine to do so.

Create legible, reproducible figures. In many disciplines the figure is the ultimate conveyor of achievements, summarizing findings (we think) in an accessible way. A lot of effort goes into figure creation. Yet this should not be repeated whenever new data come around. Once created, a figure is a solved problem. Hours wasted on re-creating it indicate the use of the wrong tool. Similarly, illegible axis labels or poor color schemes should prompt everyone at least to wonder about a tool's capabilities and, if necessary, switch to a tool that offers the required level of freedom. Sadly, more often than not, this is not done. Conveying these thoughts is not unique to us; we join the choir of people like Edward Tufte and Jon Claerbout, scientists who are calling for sensible and reproducible visualization of data.

The course has been well received by both students and faculty in our department. Several biology students have taken the class in the last two iterations, which shows that the demand for the class extends beyond geoscience. Apart from classic lecture settings, three core ideas are responsible for this success:

Provide guided practical application. Probably the biggest mistakes we made were to assume too much prior knowledge and to provide too little individualized guidance. We assumed we were instructing experienced students, but in reality they were entering a new field and were beginners on this topic. Although banging your head against a wall is an integral part of computer programing, it is necessary to keep a healthy balance between frustration and gratification; this makes a controlled lab environment indispensable. It is of great help to demonstrate individually how to solve the mostly minor problems encountered when working through problem sets. Most of this knowledge seems so deeply ingrained in the mind of experienced programers that it appears natural. Conveying these techniques and simple concepts is critical and is impossible in a pure lecture setting.

Solve student-specific problems. We assign projects that are ideally related to a student's thesis work so that they include course concepts in their daily routine. Here the key to success is heavy mentoring, which includes time-intensive code review. Given the diversity of student research, this is hard, but it comes with the tremendous gratification of engaging education that sticks with the student.

Demonstrate problem solving. A final point that inspires significant progress is "live coding." I pick a simple problem and think it through but write the actual program with the students in class. Naturally, this brings embarrassment and high entertainment potential. Between bouts of laughter, students break down complex problems into simpler tasks, learn to read error messages, see the value of search engines in debugging, and get a feeling for connecting the dots.

As a result of the course, our students make enormous strides in their programing skills and their confidence to take on problems that require those skills. We see them apply these techniques in their research and in other courses. Other course instructors will be able to assume that students who have taken our class have basic programing knowledge. This allows those instructors to use computational exercises to teach geoscientific concepts rather than programing. Our experience gives us confidence that our students will leave behind a trace of useful tools. Some already advance their community by making their work freely available; some consider publishing papers about their tools. This is surely more desirable than stacks of sticky notes. The hope is that these ideas will be fresh in your mind as you consider coming curriculum changes.

Acknowledgments

I thank Jeff Freymueller for supporting me in the pursuit of this experiment and providing valuable feedback on the manuscript. Jamie Martin and Bernie Coakley also provided many helpful comments on early drafts of the article, for which I am grateful. The Department of Geology and Geophysics at University of Alaska Fairbanks provided financial support to realize the course. I also acknowledge the supportive and constructive comments by editor Christina Cohen and two anonymous reviewers, which improved the manuscript.

—RONNI GRAPENTHIN, Geophysical Institute, University of Alaska Fairbanks; E-mail: ronni@gi.alaska.edu

MEETING

Updating the Science of Atmospheric Electricity

XIV International Conference on Atmospheric Electricity; Rio de Janeiro, Brazil, 7–12 August 2011

PAGE 470

The main goal of the XIV International Conference on Atmospheric Electricity (ICAE 2011) was to provide a comprehensive description of the status of knowledge in the field of atmospheric electricity, as well as to provide an opportunity for extensive interaction among researchers in this field. The history of the ICAE goes back to the first conference held in May 1954 in Portsmouth, N. H. The conference was attended by 51 scientists from 10 countries, and only three topics were addressed: fair weather electricity, thunderstorm electrification, and lightning.

Fifty-seven years later, ICAE 2011 was held for the first time in the Southern Hemisphere, in Brazil. The conference was attended by 191 participants from 24 countries. Eighty oral and 200 poster presentations dealt with the following topics: global circuit, fair weather electricity and atmospheric ions, thunderstorm electrification, lightning physics, lightning and meteorology, lightning and climate change, lightning and atmospheric chemistry, electrical effects of thunderstorms on the middle and upper atmosphere, lightning detection technologies and their application to power systems on the ground, planetary physics, and lightning hazard and mitigation.

In addition, seven overview presentations discussed key developments and new and persistent challenges in the field. One of these talks described new challenges in understanding lighting-related gamma ray effects, the effects of aerosols on cloud microphysics and electrification, and the still unsolved roles of electrified shower clouds and El Niño–Southern Oscillation variations on the global circuit. Another presentation covered the role of high-energy electrons and gamma rays in lightning initiation, stepped leader propagation, and limiting the electric field of thunderclouds. The growing evidence for charge separation by the mechanism of ice-ice collision, and without supercooled water, was described in an overview talk. This mechanism appears to be prevalent in long-lived anvils or trailing stratiform regions where the charging can take place slowly over extended periods of time.

A fourth overview presented new evidence for the influence of global warming on local thunderstorm activity based on proxy data in regions subject to large temperature changes. These observations stand in puzzling contrast to the available global lightning evidence showing no upward trend in recent decades as the mean global temperature increases. The talk also covered the influence of the sea surface temperature on the thunderstorm activity over land. The fifth overview described new research related to breakdown characteristics and the observational evidence for the occurrence of "recoil" breakdown in lightning, showing how it explains many of the features of interferometric and time-of-arrival very high frequency observations of the detailed development of discharges inside storms. Another presentation described recent results about rocket-triggered lightning, in particular, the recent and large body of observations in China. The final overview discussed new attempts at describing all forms of atmospheric discharge at the level of fundamental kinetic processes, including all chemical, molecular, atomic, nuclear, and photon interactions and their interactions with the macrophysical environment.

The conference generated a growing awareness of and interest in the natural framework of the global electrical circuit. Understanding this framework can help to identify manifestations of various phenomena on a local scale. Examples of these local effects include El Niño effects on lightning in Southeast Asia, the radioactivity anomaly from the Fukushima reactor accident in Japan, urban effects and the "weekend effect" on lightning activity, the behavior of ice particle charging in the laboratory and its relevance in a global set of thunderstorms, the thermodynamic and aerosol contributions to lightning activity, and the surface electric field variation in Antarctica. Many discussions centered on detection of global signals representative of the collective response to local phenomena.

The conference Web site (http://www.icae2011.net.br) includes the program, list of papers, and video files of the overviews. A special issue of *Atmospheric Research* is planned for this conference.

—OSMAR PINTO JR., National Institute for Space Research, São Paulo, Brazil; and EARLE R. WILLIAMS, Massachusetts Institute of Technology, Cambridge; E-mail: earlew@ll.mit.edu