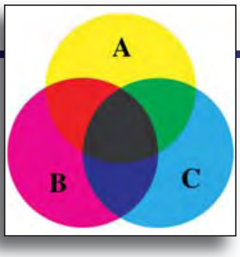
Staying Disciplined During an Interdisciplinary Degree



By Daniel W. Goldberg

Every year, at commencement ceremonies around the world, university presidents and college deans tout the benefits of an interdisciplinary approach to pursuing a degree. Awards are typically bestowed on exceptional undergraduate students who have simultaneously double- or even triple-majored in disciplines such as fine arts, business, and computer science (CS), or exemplary graduate students who have earned a PhD under the tutelage of advisors from equally disparate programs.

Exemplary, perhaps not, but I am one of these people. During the last four years at the University of Southern California, I've been a CS PhD student in the Viterbi School of Engineering. All throughout, I've been working as a graduate research assistant in the Department of Geography in the College of Letters, Arts, and Sciences, funded by grants obtained in the Department of Preventive Medicine in the Keck School of Medicine, with advisors from each.

At times, I've looked at other students in my cohort and felt sure I was going down the wrong path because their CVs had peer-reviewed conference papers while mine only had a series of non-peer-reviewed "technical reports" and two peer-reviewed articles in non-CS journals. On more than one occasion I've had the less-than-pleasurable experience of rationalizing to myself and my advisors that what I am doing is actually CS research. Perhaps most difficult of all, I've had to mediate the scientific approaches inherent to the disciplines of my advisors while carving out a research topic and schedule amenable to all that would fly for a CS degree.

Despite these challenges, I can thankfully and confidently state that my experience has been, overall, quite a positive one. My experience has imparted first-hand practical knowledge on how to identify underlying computational problems that others don't recognize, which my CS education and training has prepared me to solve or optimize. Although it was sometimes difficult to appreciate during day-long debugging sessions, the encouragement and requirement to step out of the proof-of-concept-only mold allowed me to make contributions that are actually being used by thousands of people.

This exposed me to big-picture thinking, and has made me consider the scalability, reliability, and generalizability of my solutions at every turn, concepts that I hope will prepare me well for a career in either industry or academia.

Equally beneficial, I now know a great deal more about potential funding sources and collaborators than I would have otherwise, having written grant proposals to institutes interested in things other than CS, forcing me to figure out how to situate my specific research contributions in clear and concise terms that non-experts can understand.

8 Tips for CS-Hybrid Students

1. Know your roles and responsibilities
2. Stay focused
3. Be prepared to say no
4. Keep track of tasks and time
5. Be aware of the expectations
6. Get Involved
7. Be a sponge and a chameleon
8. Stay disciplined

Most importantly, I now have a proven track record of being able to successfully work as part of an interdisciplinary team, demonstrating that I am capable of applying specialized training toward solving larger scientific goals in many disciplines. This quality, I've been told, ranks high in the eyes of many people in hiring committees in both industry and academia.

What follows are several key lessons drawn from my experience. These are insights that would have been useful to know before starting to ensure that I focused my efforts in the appropriate places. While geared toward those working in an interdisciplinary team, many are applicable to the success of any PhD student.

Know Your Roles and Responsibilities

As the "computer person" in the eyes of many on my interdisciplinary team, I've had to play hand-holder, "programming wizard," and naysayer to people with widely varying understandings of and exposures to CS. When my training allowed me to see potential logical and computational pitfalls in all aspects of study, design, and implementation, it was my responsibility to let people know. It then also became my responsibility to propose creative solutions using what I learned in my coursework, industry, and research experience.

Being a member of a team means that you will have to do your fair share of introducing your colleagues to relevant CS techniques and principles. Although it's not your responsibility to provide them an undergraduate education in the subject, you must tread carefully, thoughtfully, patiently, and respectfully. Draw clear lines as to what you can and cannot be expected to do or explain.

"I now know a great deal more about potential funding sources and collaborators than I would have otherwise, having written grant proposals to institutes interested in things other than CS…."

Stay Focused

I implore you—at all times remember what your number one priority is while in school: completing your coursework and thesis, and getting out of school. This means stop programming right now. I mean it.

Closing down the IDE and opening up the word processing program was one of the hardest parts for me. As engineers, we often pride ourselves on the beauty of the perfect implementation. However, my colleagues from other departments typically had no idea what was under the hood, nor did they care as long as it worked. It took real effort to wean myself away from immediately fixing every minor bug that anyone ever reported or adding that last bell and whistle. In order to achieve goal number one, quick and dirty are the two words to live by.

Be Prepared to Say No

Don't be afraid to decline a request to get involved in a project. Your advisors and colleagues have succeeded in their careers by effectively managing their schedules. They will completely understand if you explain that you simply don't have the time to devote to a new project right now. Make sure that your priorities are focused in the right place.

For me, it took a while to get comfortable turning people down. Early on, I would enthusiastically agree to "help out" on every "small job" which "shouldn't take me that long." Feature creep nearly always occurred, and I came to realize that non-CS people often do not appreciate the amount of time and effort it takes to design, implement, test, and debug something. When I told them, "It won't take that long," they fully expected that it wouldn't take that long, which caused me some trouble.

Once a funding agency, colleague, or potential collaborator sees even the smallest glimpse of a semi-working prototype, the scope and importance of a project will often explode. Several times I found myself devoting far more time and effort under far more pressure than I had originally anticipated. So, be sure that you have enough time to commit to your existing set of tasks (most importantly graduating!) before you agree to take on anything new.

Keep Track of Tasks and Time

Like most research assistants, I spent most of my time working my advisors' projects. To make these interactions as smooth as possible, I outlined and documented exactly what tasks I was responsible for, deadlines for those tasks, and specific deliverables prior to getting involved in the project. Because my non-CS colleagues were often blissfully unaware of the underlying implementation complexity of a project, they often underestimated the work required. Spelling out the specific technical tasks ahead of time allowed me to extend the finish line or cut all but the essential tasks from the work scope as well as have a record of what I was responsible for doing.

In addition, I found it very useful to keep a log of how much time I was spending on each task and sub-task such that I could truthfully report my progress and explain why I was ahead of, or behind, schedule. When I did find myself behind schedule, I learned that it was better to tell my advisors immediately, as they were able to re-prioritize and eliminate tasks. Most importantly, I found that waiting until near the deadline to have these conversations never worked out well.

Be Aware of the Expectations

I've found people in other disciplines often have very different expectations as to the scale and scope of the output at the end of a project. In CS, we often research and write papers showing a proof of concept of some new approach or technique that incrementally advances the state of the art. Many times, this gave me the luxury of working on small datasets without worrying about the exceptional cases. My colleagues from other disciplines, however, were oftentimes expecting production-scale systems to work equally well on tested and untested data at scales I was not anticipating.

Fixing one or two exceptional cases or scaling a system can be challenging. I found that speaking up ahead of time to set acceptable levels of performance prevented uncomfortable situations later on when the timeframe for the project was coming to an end and certain cases or workloads were not being handled perfectly. If and when you are asked if something is possible or how long it will take, don't promise the world, and don't hesitate to let your advisors know that something is just not possible given the time available.

Get Involved

My experience has given me the opportunity to learn about many new journal outlets, funding sources, conferences, and academic and professional organizations. Many of these were above my head and not applicable to my status and goals at the time, but I did encounter several that I could participate in, submit to, present at, or simply keep in the back of my head for future reference.

Sure, I was somewhat insecure about not knowing enough, but I would recommend not letting that stop you from getting involved with organizations in which you are interested or think you may have something to contribute. In my case, the other members always appreciated the effort and were more than willing to help and mentor once they saw I was serious and committed.

When a journal from another field looked like the most opportune outlet for portions of my work, I exploited the discipline-specific knowledge of my advisors about style, tone, and format to make my work applicable. It was painful to trim out my pseudo-code or waste space putting in well known formulas, such as those for precision and recall, but I've learned that you must write to your audience if you want your work to be well-received. Reading previous articles from the journal shed light on how to explain my complex techniques to the primarily non-CS audience.

I would also recommend proactively looking for sources of funding from organizations and agencies outside your field. These funding sources are always looking for CS approaches to solve problems that you might be familiar with or have a creative and novel solution for. I kept my eyes and ears open and informed my advisors when I saw potential funding streams. This creates contacts for later funding and builds experience in the process.

Be a Sponge and a Chameleon

My fly-on-the-wall status in many meetings taught me the problems people have in other disciplines and the vocabulary they use to talk about them. I soaked up everything I could understand and did a little research into what I couldn't to obtain rudimentary levels of knowledge. These insights into other disciplines continue to provide me with valuable context I leverage when I need motivating examples in papers or answers to the inevitable and implicit "why should I care about your work" line of questioning. Interesting CS problems can be dry by themselves, but when you have a great example of an application that can benefit, people's ears typically perk up.

More often than not, at conferences in other disciplines people thought I was a student in that discipline. There is more than one peer-reviewed publication on my CV that lists me as being in the Department of Geography when I know I put CS when we submitted the manuscript.

My advice in these instances is just go with it. If you continue to work in the same research area, these same editors will most likely see your papers or be reviewing your promotion or tenure package one day, so in my mind it's great that they know my name at all.

Stay Disciplined

Most importantly, don't forget that you are pursuing a degree in CS.

Regardless of what I have worked on, in the end, I need to convince CS faculty members that my work makes sufficient and significant contributions to the field of CS in order to obtain my PhD. It takes discipline to prevent your non-CS interests and activities from drawing you to the point where it becomes fuzzy as to which discipline you are actually studying. I know this for a fact; I have gone too far and needed to find my way back.

While it's perfectly acceptable to present at non-CS conferences and publish in non-CS journals, remember that you will be required to show—specifically—how you have advanced the state of the art by creating something novel, solving a particularly difficult problem, or done something faster, cheaper, or more accurately. Be prepared to justify time, and again why and how your work advances CS.

"Being a member of an interdisciplinary team means that you will have to do your fair share of introducing your colleagues to relevant CS techniques and principles"

Previously published CS works and special interest groups have been very useful for me. In my case, the ACM SIGSPATIAL proposal specifically lists my topic, but there are many places to look. Is there a workshop at an IEEE conference related to your approach? Is there a special issue of *AI Magazine* that discusses a similar problem? Are there CS professors at other universities whose work you cite?

Practice your explanation and have it in your back pocket at all times. Not only will you need it around the CS department, but it makes for excellent elevator talk should you find yourself next to a potential funder, recruiter, or faculty search committee member while sitting in a conference, working on the train, or enjoying a cocktail party.

Reflections Near the End

All in all, I'd say I picked the right path for my PhD. The most unique part of this path has definitely been the scholarly opportunities that have popped up along the way. My experience with non-CS researchers has been particularly rewarding because I have worked on large-scale multi-institution projects that may actually have a real impact on reducing the burden of diseases such as cancer.

It has changed my perspective as a student and a scholar because I now grasp how the unique talents drawn from many fields can come together to tackle many sides of the same problem. At the same time, because CS principles underlie much of the computational tools used in geography and health, my experience has allowed me to delve far deeper into the practical uses of the algorithms we CS students sometimes only get to know in theory.

Biography

Daniel W. Goldberg is a PhD student in computer science at the University of Southern California. His work in the USC GIS Research Laboratory focuses on the uncertainty of geospatial information and processes. He is the recipient of numerous Geospatial Intelligence scholarships and is the author of the North American Association of Central Cancer Registries' Geocoding Best Practices Guide.