Texas
TECHNOLOGY
MARCH 2001 DALLAS/FORT WORTH • AUSTIN • HOUSTON • SAN ANTONIO WWW.TTECHNOLOGY.COM

WEB DOC > DO's AND DON'Ts WHEN BRANDING YOUR SITE [P.28]

PURSUIT
WHEN COMPANIES FALL OFF THE NASDAQ [P.25]

CAREERS
LIFE AFTER LAYOFFS FOR TECH PROS [P.47]

name game

The Difficult Task of Establishing BRAND IDENTITY ON THE INTERNET

HOUSTON'S NO. 1 TECHNOLOGY MAGAZINE

Teaching the Techie

training

THE SEARCH FOR INTELLIGENT INSTRUCTION IN A CRAMPED UNIVERSE **BY EMILY NEDELL**

Deciding what career path to take is a daunting task for someone vaguely familiar with the various certifications and commercial applications that define the curriculum of continuing education courses in information technology.

People considering a career change may know little more than which certifications pay well, and which ones don't do much to a paycheck. Perhaps it's enticing to the IT professional that those who "know Oracle" command high salaries. But taking the plunge into learning a new language is a big commitment, and IT professionals many times don't know if the new knowledge yields enjoyment on the job until he's in the position to apply what he's learned.

Last year I attended Oracle University's "Intro to Oracle: PL/SQL," a 3-day class for experienced SQL users. It was more affordable ($1,250) than the 5-day for beginners, and I couldn't take five days off from work, even though the company I worked for was in the process of implementing an Oracle database. The fact that my contract was up at this company and Oracle was being installed was an added incentive for me to sign up for training.

Everyone else in the class was a working professional whose company was paying for their training. At first I felt embarrassed by my lack of corporate sponsorship – a peon who had merely bought her way in rather than being chosen by my company. I didn't even have a business card to send sliding across the tables. But when the leader had everyone make introductions, I discovered that many people spent the day just running SQL queries, generating reports for other departments. Others were database administrators (DBAs) trained on other relational database systems, and their companies were transitioning to Oracle. I tried to imagine what they did exactly in "administering" a large database.

Needing no review of SQL and relational database theory (they were there just for the PL part), the "experienced SQL users" slept through the morning session while I, the wannabe, diligently took notes. Though I certainly didn't object to the Sequel review, I was disappointed that we weren't going to be taught even the most basic tasks, such as backing up the database—"Oh, you don't get to that until DBA training."—approximately $5,000 and several classes away. Although this was the first class in the sequence of Oracle certification classes, this wasn't quite the "Intro to Oracle" I'd expected.

At the end of the day I cornered the trainer in her office and began asking questions, including some having to do with applications that run on top of Oracle and how they interfaced. (This knowledge had particular relevance to me at the time.) "Oh, you'll learn the answer to that question in this other course," she said. I couldn't tell if she didn't know the answers or was sworn by Oracle University not to disclose that information. I got home from day one, neck sore from having to sit at an angle for 8 hours

(we were two to a computer), thinking that it was crazy for me to be taking this course. My company was outsourcing its Oracle needs to experienced Oracle developers and DBAs — and most companies that can afford Oracle can pay for highly qualified people. Even if I managed to get that entry-level Oracle job — not that I have ever seen ads for such a thing — would I be happy working in a large company running SQL queries all day long?

To their credit — and my relief — Oracle University refunded my tuition when I explained to them that the course was not what I had in mind.

Easy Doesn't Do It

Nonprogrammers have no idea what is involved in learning how to program; I could see myself easily making the same mistake nine months ago, before I took my first C++ and Visual Basic class from Houston Community College. The proliferation of books such as *Sams Teach Yourself C++ in 24 Hours* has led to the widespread perception that programming can be learned quickly, and from a book. But even the author of that book, Jesse Liberty, explains elsewhere that what he means by his title is "300 hours, 24 hours of lessons and 276 of practice." And then: "After that, six months of full-time work is required to achieve proficiency." (Liberty, *The Complete Idiot's Guide to a Career in Computer Programming*) He is also skeptical that people can learn programming "on their own" anymore, as they could in the old days. Programs and programming languages have become too complex.

Liberal arts majors are smart enough to learn anything, and many of them have successfully transitioned into IT. Yet it is easy to underestimate the difficulty of bootcamp IT training – such as the MCSD – for those who have no prior programming or database experience, and no exposure outside of class to the applications they will be learning. Beware: Private training outfits who used to depend exclusively on large corporate clients to fill their classes are now aggressively pitching certification training to individuals looking for an entrée into IT. Where before some knowledgeable and experienced person made the decision to purchase training for an employee or a department — how much training and at what level — now people with little background are risking thousands on classes in the hope of improving their net worth. Sales reps promise people with no experience that they'll pass certification exams or land a $60K job upon completion of training. Isn't this something like selling Bar exam prep courses to people who haven't attended law school?

Looking for Mr. Right

A few weeks after my Oracle experience, I began researching my options, and I couldn't find one that suited me. It wasn't just my anxiety about writing a check for several thousand dollars to mom-and-pop trainers in barely furnished office suites, or the free-wheeling pricing strategies of some of the more upscale providers (the prices in the brochures are never the real prices). Or the fact that certification classes would run for one week during the day and break for three or four weeks, making it impossible to work a regular job while pursuing certification.

The truth is, none of these certifications resonated with me. I fell between Macromedia and the MCSD. I wanted to be a programmer, but didn't want to be limited to programming forms and reports. I wanted to learn how to develop relational databases, but ones that could be used in expert systems or multimedia applications. I wanted to learn how to create educational software, develop natural language interfaces, and develop real-world simulations. I exasperated salespeople with my naivete and ambition, but no one ever suggested that IT training was not the answer. One suggested I learn Active Server Pages, which allows for interactivity; another accused me of not sufficiently appreciating the difference between front end and back end.

They just didn't know what to do with me.

I enrolled in the community college where I could take whatever I wanted from any department. Tuition is relatively cheap, and it turned out to be a good decision. But, like with the commercial certifications, I didn't seem to fit anywhere. From the beginning I was troubled by the partitioning of technical communications (interactive multimedia, graphic design, web technologies, 3D animation; Macromedia, and Adobe) from computer science (C++, VB, database management, Microsoft technologies, Oracle, and Unix). My multimedia instructors didn't know any VB or C++, and with their out-of-the-box applications and graphic design orientation, they couldn't create interactivity beyond mouse clicks. The graphic art of interactive multimedia doesn't include modeling real world conditions or behaviors, or creating interactivity using language or textual input. For these things, you need programming. My programming instructors, on the other hand, as excellent as they have been, don't know much about programming interactivity, multimedia, or graphics.

Much of the focus of computer science over the last 20 years has been on developing the infrastructure for business processing, financial reporting, networking, personnel and inventory management systems, customer management, etc. Now that most companies are using the same proprietary systems to accomplish these tasks, the majority of IT positions have to do with implementing, managing, and updating these applications. The consolidation of the industry around pre-existing commercial technologies — Microsoft, Sun, Oracle, Cisco – has resulted in the proliferation of certifications and training programs, and with it, the substitution of "IT" for "CS" when referring to technologies that are practical, market-driven, and commercial. Though undergraduate computer science majors learn Oracle and MS SQL Server too, computer science at that level has become preoccupied with practical business applications and solutions.

The disciplinary boundaries separating computer science and multimedia evolved not only for historical and intellectual reasons (prejudice by software engineers that graphical interfaces are for children), but for commercial ones also. Microsoft didn't do well in the multimedia market, which continues to be dominated by software applications developed originally for the Macintosh.

Commercial certifications are also business apps or graphics – or Microsoft or Macromedia – leaving designers in the dark about how to put intelligence into that GUI, and forcing programmers to figure out how to get Macromedia Director to talk to an ODBC database.

The IT training industry is pragmatic. It trains workers in the fields that are most in demand in a particular part of the country, imparting what is needed for you as an employee (or potential employee) to accomplish the most common tasks, using the most popular software packages, under the most ideal business conditions. With this training and luck, you may find yourself using the latest tools, commanding a good salary, and having a lot of responsibility within your organization. But despite the hype, IT positions are not on the cutting edge of technology. In fact, they are the opposite, serving to maintain the status quo within an industry that's highly consolidated. IT professionals are specialists in commodity solutions.

The Future

Edward Youron, author of *The Rise and Resurrection of the American Programmer*, asked, "Are we [IT professionals] doomed to spend our careers working as Microsoft-style, shrink-wrapped product developers or building customer-based, service oriented applications for big companies?" He suggests no, that the most well paid, exciting, and fun jobs over the next few years will be in the areas of virtual reality, interactive multimedia, AI, expert systems, and robotics. But he fails to mention in that competencies required to become a commercial developer are far short of what is required to develop expert systems – at least the way the curriculum is currently structured. You aren't going to get this knowledge from a commercial certification training program or from IT courses at the community college — at least not yet. Why? These technologies haven't developed enough of a consumer-base. There are no products associated with them.

Without additional foundation in computer science, IT professionals are trapped within the matrix of Microsoft, Sun, Cisco, and Oracle — caught up learning the particular applications, obtaining (and renewing) certifications, and buying magazines, which serve to promote commercial products or upgrades. Even the titles carried in the "Computing" (not "Computer Science") sections of Borders and Barnes & Noble feature a monotonous selection of how-to books, *Dummies* books, and prep guides for certification exams. I don't believe, as Dave Faries implies (*Texas Technology*, "The Average American," Jan. 2001), that techies are anti-intellectual, or any less intellectual than any other specialist, but that our field has been excessively commercialized to the point where we are all reduced to end users, the passive recipients of technology.

One upshot of this commercialization is a leveling of the playing field; someone attending a community college is not much worse off than an undergraduate in computer science at a major university, at least for the first two years. In fact, for those considering certification training, I'd recommend taking an introductory programming, networking, and a database course before investing several thousand dollars in a training program. It's not only good preparation for this accelerated training, but it might help you to decide whether IT is where you want to be. ■

Emily Nedell holds an MA in Library and Information Science from UW-Madison. She has worked on library automation projects, from school districts to museum collections and large digital libraries. She is currently enjoying classes in computer science at Houston Community College and contemplating her next career move.