

**developer.\*****A Web Magazine for Software Developers**

## Specialties and Strategies (Part 1)

By Daniel Read

*Note: this is Part 1 of a two part essay.*

Career-wise, software developers really have an amazing variety of specialties to choose from. The various specialties can be differentiated in several ways:

- by programming language
- by hardware platform
- by operating system
- by technique and/or process
- by persistence type (that is, relational databases vs. object databases vs. flat file databases etc.)
- by vertical market
- by market sector (government vs. business vs. non-profit)
- internal use vs. shrink-wrap
- client/server vs. distributed n-tier
- fat client vs. thin client
- Open Source vs. proprietary

*This list could go on and on. It seems as if there are as many specialties (or rather, combinations of specialties) as there are individual developers. Most developers cultivate not just one specialty, but rather a combination of specialties that cut across all of these differentiating factors. Independent consultants, especially, must diversify their portfolio of specialties in order to protect against the whims of the marketplace—or perhaps just to prevent boredom. The difference between one developer’s combination of specialties and another’s is often simply a matter of emphasis of one factor over the others.*

How does one come about one’s specialty or specialties? Once we have chosen a specialty (or after our specialty has chosen us), how do we maintain that specialty, and, conversely, how do we decide whether it might be a good time to abandon our specialty for another one? If we desire to maintain a variety of specialties, which one should be emphasized, if any? Which specialty is more marketable than others? What is the “right” niche for me?

It pays to give attention to these kinds of questions. Conscious thought and strategizing are required in order to acquire, maintain, and build on a software development specialty (or combination thereof). The only one who can manage your career is you. My intention here is not to tell you the answers to these questions, since I could not if I wanted to try. I don’t even have all the answers for my own career. As I navigate my career and the landmine-filled marketplace, I struggle constantly with finding the right questions to ask

and providing myself with the best answers. Hopefully, though, in the pages that follow I can shine some light on the dynamics at work in this crazy software development universe which we are lucky enough to call home.

## Choosing the Right Specialty

Which is the “right” specialty? The answer to that question will, of course, be different for each person. Furthermore, there are most likely several “right” specialties. Any given person might thrive and be happy in any one of them—or even *several* of them over the course of a career. With so many potentially lucrative and enjoyable specialties, and potentially several decades in a career, there is no reason a person must limit herself to just one specialty. That said, I think the “right” specialty is one that

- makes you happy most of the time,
- offers you the amount and kind of security you desire,
- offers you the level and type of challenges you desire,
- and has the potential for the amount of money you require or desire (assuming money is important to you in your software development career).

These are minimal and general criteria, but there are other more specific concerns as well:

- Does the specialty have longevity? Where in its life cycle is the specialty? (see below)
- Does the specialty appeal to your sensibilities? For example, your sensibilities might lean towards using a low level language in a command line environment, while someone else might strongly prefer a higher level language in a graphical environment. One person might want to work on relatively low risk business applications, while another might thrive on the pressure of developing life-critical systems.
- Is the specialty in line with your abilities? That is, do you have a) the right kind of training and knowledge, and b) the required intelligence and/or temperament?, and/or c) the drive and dedication to acquire the right training and knowledge, whether or not you possess the required intelligence and temperament.
- Given that you have the intelligence and temperament, and that you have, or can obtain, the required training and knowledge, is the specialty otherwise within your reach? How high is the barrier to entry? That is, who do you have to know to get a job working in a particular specialty? How many of those jobs are available? What parts of the world are those jobs located in? What kind of background is required? What kind of credentials or certifications are required? What kind of government security clearances are required?
- Is the specialty too narrow for you, too wide, or just right? (see below)
- Is the specialty in line with your life principles and ethics? Particularly, are you willing to work for the companies or government agencies that will pay for the specialty?

- Is the specialty undergoing rapid evolution, or are techniques, body of knowledge, and technologies relatively stable? If the specialty is undergoing rapid evolution, are you willing to put in the extra time and energy to keep up?
- Is the specialty going to suit your ego? That is, how is the specialty regarded in the industry at large, and do you care what other people might think about the specialty? Do you care that your specialty might be ignored by technology journalists, book publishers, academics, and/or gurus? Or do you require that your specialty be a popular one?

All of these seemingly objective criteria imply that a person has the opportunity to sit down, perform an evaluation based on these criteria, and then just choose a specialty. It may be this simple for people in particular circumstances. For instance, a student who is near the beginning of her academic career can ask these questions and make some decisions about which direction to take. A person already in the non-technical working world who wants to try to break into the software development field also has the opportunity to pick what kinds of technologies, tools, and domain knowledge to acquire. Similarly, a working software developer seeking to switch to a new specialty can exercise some amount of discretion in choosing his next specialty.

However, for many people, including myself, our specialties largely choose us. At the time of this writing, my specialty is developing business software, centered around databases and data exchange, using languages and tools that operate at a fairly high level of abstraction. At the risk of dating this essay for future readers, I specialize in distributed, client/server, and component-based software for the Microsoft Windows platform. I work for the business sector, developing in-house, shrink wrap, turnkey, and/or web-based software. I am skilled in relational database design, object oriented design, and user interface design.

What's interesting is that, while I have cultivated and refined these specialties over several years, they are 100% rooted in the very first job I was lucky enough to get in the software field. That job was with a company which developed database-centric accounting software, using high level languages on the Microsoft DOS and Windows platforms. The evolution from there to where I am now has been fairly logical. If my first job, which I landed at a time when I had close to zero experience, had been with a company that developed device drivers for the UNIX platform, my career might have been much different. So, while we can put a lot of thought into choosing a specialty, and we can strategize to navigate our careers in one direction or another, often the directions we go are determined by the opportunities that come our way.

## **Assessing the Risks of Having Too Narrow a Specialty**

The judgment of whether one's specialty is "too narrow" will differ from person to person, specialty to specialty. Your career strategy, temperament, economic goals, and ego will all play a part in making this judgment. That said, I think there are several categories of factors to consider. Examine these trade-offs closely when considering a very narrow specialty:

**Economic and market factors** For people who wish to make their living as software developers, market forces are probably the most important factors to consider in the wide vs. narrow question. Interestingly, a narrow specialization can work for and/or against you in different situations. For example, if you are an expert in a certain programming language that is used by a relatively small number of companies, then there are less customers out there to whom you can sell your services. So that works against you because it makes it harder for you to find work, and also because it increases the likelihood you will have to travel or move around to get work. However, scarcity can also work in your favor, because just as there are few companies who use the language, there are also correspondingly few developers who are experts in it, meaning that you can potentially charge a high fee for your services.

**Availability of information and instruction** If your specialty is narrow, then there will be fewer magazines, books, and web sites dedicated to your specialty. General technical periodicals and web sites will tend to give less, if any, coverage to your specialty. There will be fewer mailing lists, discussion forums, and user groups. There will be fewer training classes and conferences. These kinds of resources are invaluable to a developer, especially when one is just starting out, or in an intense learning phase.

**Speed of innovation** With fewer people and fewer companies working with in given technology or area, innovation will tend to be slower. This happens because fewer people means fewer new ideas thrown into the mix, and fewer companies means market competition is less of a factor in motivating companies to create new products. Speed of innovation is also hampered by the fact that information is flowing less quickly and less widely because of the aforementioned dearth of publications, conferences, etc.

**Availability and variety of tools** If the market is small for a given specialty, then fewer companies will exist who are developing tools to make it easier to work in that specialty. Similarly, fewer motivated developers might exist to develop shareware, freeware, and Open Source tools and solutions.

**Risk of obsolescence** If a market is small, then it might tend to grow smaller, (see the section titled “Monitoring the Trends” below). If the market grows smaller, then tools vendors might go out of business. Tools vendors might also get bought up by larger companies, who might not maintain the same level of stewardship that the original company did. Also, if a specialty’s survival depends on a particular technology, such as an operating system or hardware platform, then the obsolescence of that technology can lead to obsolescence of the specialty. This kind of dependency is definitely an important risk to consider.

**Ego factors** Many of us might not like to admit it, but our egos and personalities play a big part in our career decisions. If a particular specialty is especially popular, widespread, and hyped up, then that fact might appeal to one person, but repel another person. Some very narrow specialties might be looked down upon by many developers and publications, if they notice the specialty at all—this situation might really bother some people, who would

want to work in a specialty that is well thought of and respected, even if it's not popular. Also, what world do you look to for validation and guidance? Do you gravitate towards academia or the commercial sector? These groups might tend to favor certain specialties and not others. If the acceptance and approval of people in these camps is important to you, then that's something to consider.

Here is an example from my own career: as mentioned, I specialize right now in developing for the Microsoft Windows platforms. While they are of course not perfect, I personally very much like the tools and technologies that Microsoft produces. I especially find it a pleasure to use their development tools. Also, Microsoft's platforms and tools are very well suited for the kind of high level, business-oriented, database-oriented development that I do. If I were doing a different kind of development, I might find that their software was not well suited for what I was doing.

At the time of this writing, the vast majority of the people who are writing what I consider to be the important works on design, methodology, architecture, construction techniques, etc. look down on Microsoft tools and technologies with total disdain. (In my opinion, this disdain is rooted in hatred for Microsoft's business practices and current monopoly status rather than in objective evaluation of the technologies themselves, but that's a whole different subject.) My point is, right now, the Java programming language and Open Source software are hot, hot, hot, and it is not exactly considered cool in the world of software magazines, web sites, and books to be a Microsoft-oriented developer (even though there are millions of developers writing to the Microsoft platforms). Many people feel so strongly about this, they probably stopped reading this a couple pages ago simply because I revealed my Microsoft-oriented specialization.

This is where ego comes in. Does this unpopular status affect how I feel about my work? Does it affect my future career plans and strategies? Simply by switching technology platforms, I could gain the acceptance of what I perceive to be the current mainstream in software engineering thought. While I will admit to giving thought to the issue and being slightly bothered by this prevailing bias, I have decided to ignore it and proceed on my chosen path. This path makes me happy, I make a good living at it, and it has a viable commercial future—that's enough for me.

(I look forward to re-reading this a couple decades from now and laughing about how silly and dated the above three paragraphs will sound.)

**Ethics and principles** With certain specialties, you might want to ask yourself some tough questions about your own ethics, morals, and principles. For example, a certain programming language might be used primarily by parts of government or industry that you might not want to work for. If serving non-profit, religious, or charitable organizations is your goal, you might want to specialize in technologies that these organizations can afford, and which will be relatively stable for a relatively long time. You might not want to use a particular company's technologies because you disagree with that company's business practices (see above). You might not want to use any proprietary or commercial

technology at all because you have strong feelings about the effect of capitalism on technology. No one can decide these kinds of questions but you.

Another wrinkle in this is to be honest with yourself about how much you are willing to set aside certain principles in the interest of making money.

## Aiming Consciously at the Right Level For You

Let's face it: not everyone is as smart as the smartest among us. Likewise, even among people who are equally "smart," people have different talents. For example, some people just have an amazing natural ability with numbers, complex calculations, and "higher math." Other people might have less natural ability with math, but are gifted in the use of language and writing. Other people might have a gift for the graphic arts. It is not my intention here to open the nature vs. nurture debate regarding how people get the way they are, but rather to acknowledge that people have differences in these kinds of areas, and that this is normal.

Furthermore, my point here is not to make anyone feel bad about talents they may feel they lack, but rather to say that it would be wise to aim for a specialty that complements your talents. For example, there is no doubt in my mind that the world of software development is full of people a lot smarter than me, and that many of these people can work miracles with numbers and algorithms and really hardcore stuff that I have no idea about. When I look at any of Donald Knuth's *The Art of Computer Programming* books, my head starts to spin. Simply put, I am not a "math person."

Given that I am not a math person, I did not choose a specialty that requires me to be one. I leave those specialties to all those people who seem to have been born for them. I chose a specialty that allows me to work at a fairly high level of abstraction, and to use talents that I do have, such as analytical thinking, logic, abstract thinking, writing, organization, and people skills. In addition to thinking about what you *like* to do, think also about what you have a natural talent for, and look for ways to apply that.

So that I don't leave you with the wrong impression, I want to make two things clear: First, if you feel you don't have any natural talents, you are mistaken. You just are not aware of them yet. Second, if you are passionate about an area that you feel you might not have a natural talent for, then absolutely do not let any self-limiting ideas about what talents you may or may not have stop you. Let me quote Theodore Sturgeon:

Anybody can do anything he wants to if he wants to do it badly enough. Now I know that's a vast oversimplification, but in principle, it's true. For example, I think that you or I or anyone could be a wire-walker for Barnum and Bailey, and all it would take would be an absolute determination and practice, practice, practice. You have to study your field and you have to find out how other people do it, and you have to keep working and learning and practicing and ultimately, you would be able to do it.

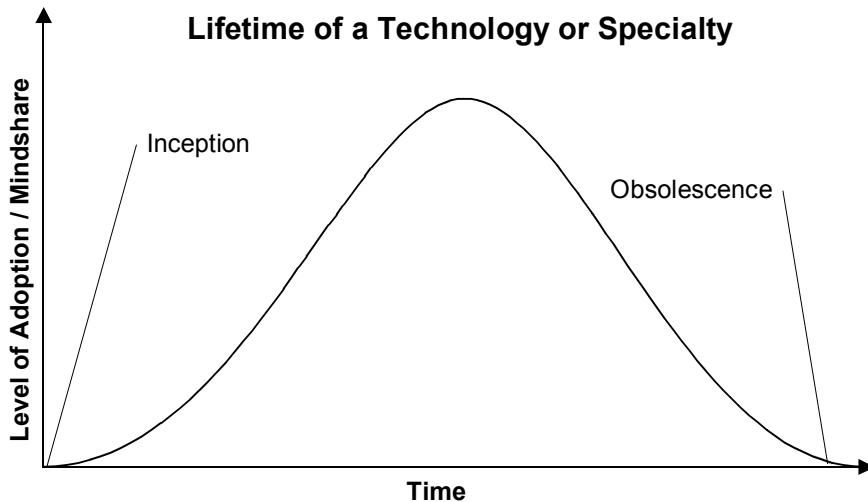
However, there are people who are able to do it in the first three tries. They're born with such coordination and talent, a construct of the semi-circular canals in the inner ear, that they're able to do a thing like that with great ease. Now skill is what you develop by that kind of practice and work and study. Skills can be developed and refined and brought to a very, very high pitch indeed.

Talents you are born with. There is simply no question about it. People with a high talent unfortunately don't have to continually practice. They don't have to, so they don't do it. It is a great loss because a lot of highly talented people never work at their talents at all. In the end, their product is lesser than somebody who has worked his buns off to get somewhere and refine the talent that he has. (1)

I have used this quote in an essay before, but it is powerful enough that I decided it was worth repeating.

### Monitoring the Trends

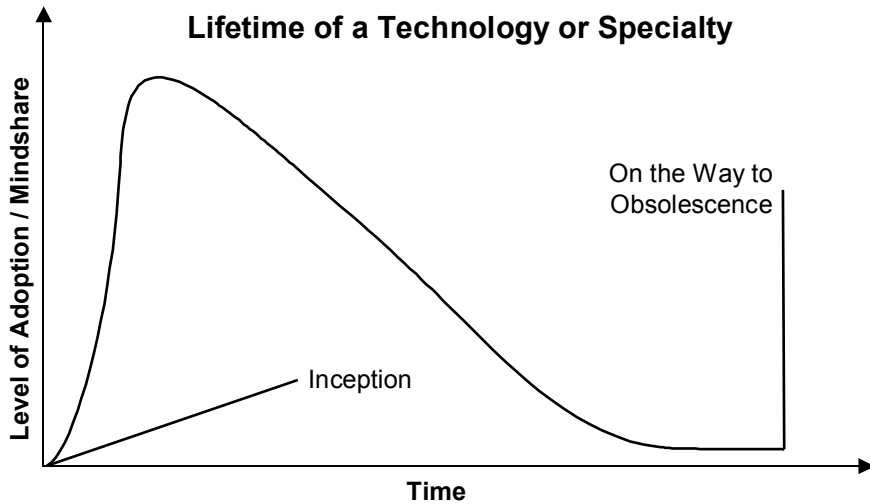
All technologies and specialties are subject to trends. Any one technology or specialty is likely to follow a bell curve from obscurity to popularity and back to obscurity again. Here's an illustration of what I mean:



**Figure 1: A gradual rise and fall in popularity.**

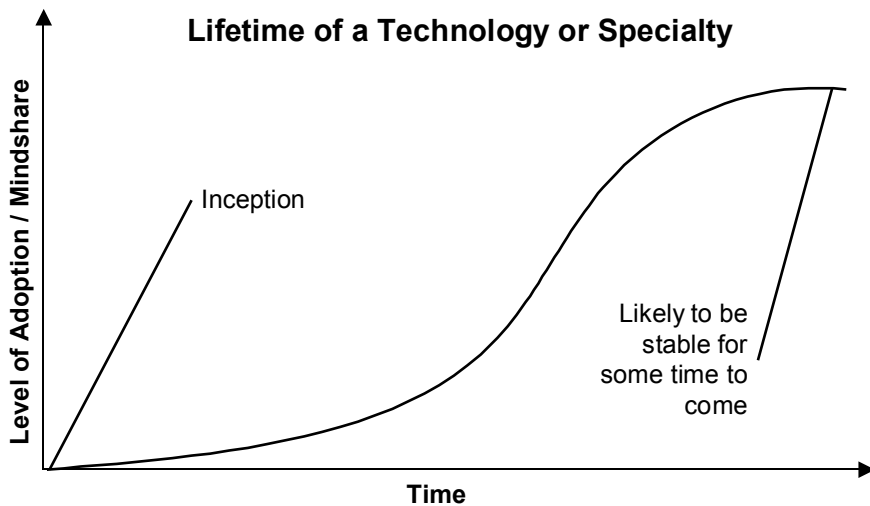
Here in Figure 1 you can see a technology over time gradually growing in popularity, peaking, and then dwindling towards obsolescence. (Please note that none of the diagrams I will be using in this discussion were scientifically or statistically generated. I conceived them for illustrative purposes only.) The curve in Figure 1 shows a relatively equal and gradual amount of time spent rising and falling in adoption and mindshare.

In some cases, you might have a more meteoric rise and fall, as in this curve:



**Figure 2: A fast rise in popularity, with almost immediate falling off.**

In Figure 2 we see a technology that became very popular very fast, but did not stay popular for very long. This might happen because the technology was a great idea that caught a lot of people’s attention, but early implementations were problematic. The fact that the decline is more gradual than the rise suggests that perhaps patches were released to fix some of the problems, but not soon enough to keep some of the initial people who ran into problems from abandoning the technology.



**Figure 3: A gradual rise in popularity, with a sustained period of high adoption.**

In Figure 3, we see yet another example: a more gradual rise in adoption and mindshare, and once the peak is reached, it turns into a plateau that is sustained for some time.

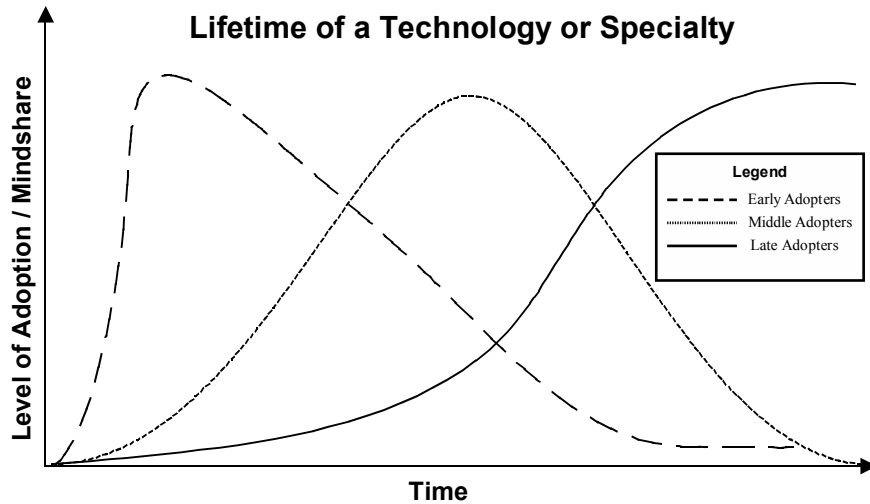
So what's the point of all these graphs? The point is that, from an industry-wide point of view, your specialty, no matter what it is, is subject to these kinds of trends. Will your favorite technology look exactly like one of these curves? Probably not, but you want to stop and consider a few things about your favorite technology or specialty: What might its curve look like? How long will the curve stretch out? Months? Years? Decades? Where are we on the curve now? Is my specialty on the rise, or on the decline? If my specialty is on the rise, is it a healthy, gradual rise, as in Figure 1 or 3? Or is it a meteoric rise fueled by hype? If the rise is fueled by hype, what's going to happen when technology journalists get bored with it and the hype dies off? Can the technology come close to living up to the hype? Asking and trying to answer some of these questions can play an important part in both choosing a specialty, and evaluating when it might be time for you to start moving away from your specialty and finding another one.

What's interesting about these graphs is not the graphs themselves, but the underlying reasons for the rising and falling. When we examine these reasons, and examine the interrelationships between the graphs of multiple technologies and specialties, we can see how oversimplified these graphs in Figures 1, 2, and 3 are. A contemporary example: remember the period from 1994 to 1997 or so, when web pages were really hot, and everyone had to have one. The problem was that in order to make web pages, especially really slick ones, you had to write them by hand using a combination of markup language and scripting that was arcane to most people. During this period, programmers were making a lot of money as HTML developers.

So there was a meteoric rise in demand for this skill, and a corresponding meteoric rise in what people were willing to pay for it. However, over time tool vendors started coming out with better and better programs for making web pages that required less and less programming skill. Not to mention, the ability to create a web page was added to word processing and spreadsheet programs, database products, report generators, etc. Soon anyone could create a decent looking web page with no programming whatsoever. Eventually, the demand for HTML developers fell off—even though the technology itself (web pages) continued to become more and more popular, and continues to be very popular five years later. So the technology keeps rising, but the specialty (HTML coding) falls off dramatically.

If you were making a living during this period as an HTML developer, you would have hopefully been paying close attention to the market forces at work so that by the time people were no longer willing to pay for your skills as an HTML coder, you had already adopted a new specialty that people were willing to pay for. Even better, perhaps you were clever enough not to make HTML coding your only specialty. Perhaps you would have had multiple specialties: a foundation of specialties that are in periods of sustained and stable high demand, supplemented with more fleeting specialties like HTML coding. While people are paying a lot of money for the fleeting specialties, you can cash in, but when those trains leave the station, you'll still have your bread-and-butter sustainable specialties. These are the same principles people use when making financial investments: diversify to protect yourself against the up-and-down cycles of the market.

You would also do well to consider how different kinds of companies have different tolerances for all these rises and falls in adoption of technologies and techniques. The graphs in Figures 1, 2, and 3 give you and industry wide view. Let's look at a more detailed view that is still from an industry-wide perspective, but which is divided up to show how different kinds of companies have their own adoption curves:



**Figure 4: Different kinds of companies and developers have different tolerances and timelines for adoption of the same technology.**

In Figure 4, we see three different categories of companies. Early adopters frequently jump right on new technologies and techniques, but don't tend to stick with them as long, because there are always other new technologies and techniques to adopt. Middle adopters take a little longer to adopt a new technology, but they stick with it a little longer than the early adopters. Late adopters only pick up on a technology or technique when it is already very well established, but then stick with it for much longer because the cost of changing technologies too frequently is too high. When late adopters make a technology choice, they have to make sure they make the right one, because they don't want to have to change it too soon. (Note: this is very much a simplified discussion of these kinds of market forces. For a much richer view on these subjects, check out Geoffrey Moore's popular books *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers* and *Inside the Tornado: Marketing Strategies from Silicon Valley's Cutting Edge*.)

What does this mean to you? If you are an individual developer who tends to work for one employer at a time, you need to ask yourself what kinds of companies you want to work for. If you want to always be working with the latest technologies and techniques, you want to work for early adopters—but the price you will pay is that you will have to put in a lot of extra time and energy learning new technologies and making sure you stay ahead of the trends. If you really just want to become good at one stable specialty, and not have to be chasing rabbits down holes all the time, then that's fine too: just pick a specialty with a long, sustained high adoption rate and go to work for a late adopter company that is

committed to that specialty. If you are a consultant or consulting company with multiple clients, then you might do well to diversify, and keep a mixture of early, middle, and late adopter companies in your client list. Finally, notice the link here between you the individual and the companies who might pay for your services: just as companies are early, middle, and late adopters, so too are people. Consider your own temperament and goals, and line those up with companies who have similar temperaments and goals.

This discussion on the variations of trends and trend-based career strategies could go on for many more pages, but I will stop here. The bottom line is this: if you plan to have a multi-year career as a software developer/technologist, then you need to pay some attention to the trends going on around you. You need to develop a personal strategy that will ensure that you can use these trends to your advantage. Even if you have no desire to chase trends and cutting edge technologies, you don't want to be caught by surprise someday to find that your only bread-and-butter specialty is on the verge of being obsolete.

## Career Strategies

This is the end of Part 1 of this installment. Part 2 will pick up with a discussion of strategies to stay ahead of the game—no matter what your specialty.

## Additional Resources

Anyone interested in the market and mass psychological forces at work in the computer technology industry, might want to check out Geoffrey Moore's books *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers* and *Inside the Tornado: Marketing Strategies from Silicon Valley's Cutting Edge*.

## References

(1) From an interview with Theodore Sturgeon by David D. Duncan; copyright unknown, but published with the author's permission at <http://glinda.lrsm.upenn.edu/~weeks/misc/duncan.html>.

###

Daniel Read is editor and publisher of the **developer.\*** web magazine. He lives in Atlanta, GA, where he works as a software architect. He is currently at work on a book about software development crafted for a business audience.