How Developers Stop Learning: Rise of the Expert Beginner

By **Erik Dietrich**, www.daedtech.com
September 30th, 2012

Beyond the Dead Sea: When Good Software Groups Go Bad

I recently posted what turned out to be a pretty popular post called "How to Keep Your Best Programmers," in which I described what most skilled programmers tend to want in a job and why they leave if they don't get it. Today, I'd like to make a post that works toward a focus on the software group at an organization rather than on the individual journeys of developers as they move within or among organizations. This post became long enough as I was writing it that I felt I had to break it in at least two pieces. This is part one.

In the previous post I mentioned, I linked to Bruce Webster's "Dead Sea Effect" post, which describes a trend whereby the most talented developers tend to be the most marketable and thus the ones most likely to leave for greener pastures when things go a little sour. On the other hand, the least talented developers are more likely to stay put since they'll have a hard time convincing other companies to hire them. This serves as important perspective for understanding why it's common to find people with titles like "super-duper-senior-principal-fellow-architect-awesome-dude," who make a lot of money and perhaps even wield a lot of authority but aren't very good at what they do. But that perspective still focuses on the individual. It explains the group only if one assumes that a bad group is the result of a number of these individuals happening to work in the same place (or possibly that conditions are so bad that they drive everyone except these people away).

I believe that there is a unique group dynamic that forms and causes the rot of software groups in a way that can't be explained by bad external decisions causing the talented developers to evaporate. Make no mistake–I believe that Bruce's Dead Sea Effect is both the catalyst for and the logical outcome of this dynamic, but I believe that some magic has to happen within the group to transmute external stupidities into internal and pervasive software group incompetence. In the next post in this series, I'm going to describe the mechanism by which some software groups trend toward dysfunction and professional toxicity. In this post, I'm going to set the stage by describing how individuals opt into permanent mediocrity and reap rewards for doing so.

Learning to Bowl

Before I get to any of that, I'd like to treat you to the history of my bowling game. Yes, I'm serious.

I am a fairly athletic person. Growing up, I was always picked at least in the top 1/3rd or so of any people, for any sport or game that was being played, no matter what it was. I was a jack of all trades and master of none. This inspired in me a sort of mildly inappropriate feeling of entitlement to skill without a lot of effort, and so it went when I became a bowler. Most people who bowl put a thumb and two fingers in the ball and carefully cultivate tossing the bowling ball in a pattern that causes the ball to start wide and hook into the middle. With no patience for learning that, I discovered I could do a pretty good job faking it by putting no fingers and thumbs in the ball and kind of twisting my elbow and chucking the ball down the lane. It wasn't pretty, but it worked.

It actually worked pretty well the more I bowled, and, when I started to play in an after work league for fun, my average really started to shoot up. I wasn't the best in the league by any stretch-there were several bowlers, including a former manager of mine, who averaged between 170 and 200, but I rocketed up past 130, 140, and all the way into the 160 range within a few months of playing in the league. Not too shabby.

But then a strange thing happened. I stopped improving. Right at about 160, I topped out. I asked my old manager what I could do to get back on track with improvement, and he said something very interesting to me. Paraphrased, he said something like this:

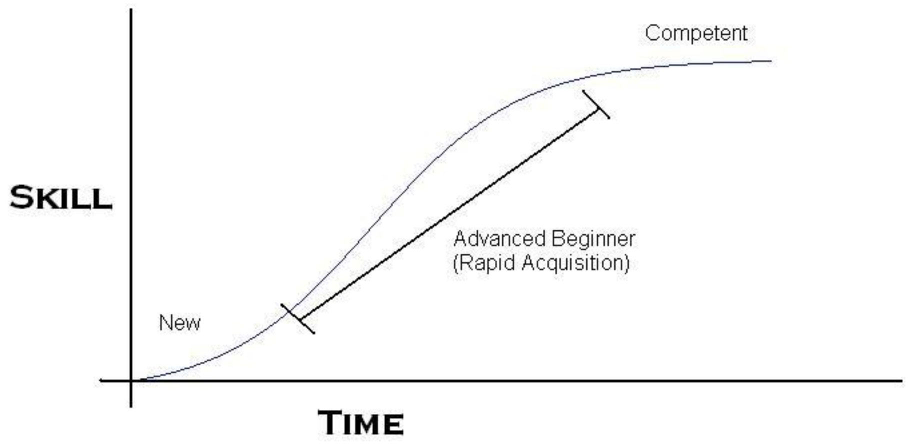
> There's nothing you can do to improve as long as you keep bowling like that. You've maxed out. If you want to get better, you're going to have to learn to bowl properly. You need a different ball, a different style of throwing it, and you need to put your fingers in it like a big boy. And the worst part is that you're going to get way worse before you get better, and it will be a good bit of time before you get back to and surpass your current average.

I resisted this for a while but got bored with my lack of improvement and stagnation (a personal trait of mine-I absolutely need to be working toward mastery or I go insane) and resigned myself to the harder course. I bought a bowling ball, had it custom drilled, and started bowling properly. Ironically, I left that job almost immediately after doing that and have bowled probably eight times in the years since, but c'est la vie, I suppose. When I do go, I never have to rent bowling shoes or sift through the alley balls for ones that fit my fingers.

Dreyfus, Rapid Returns and Arrested Development

In 1980, a couple of brothers with the last name Dreyfus proposed a model of skill acquisition that has gone on to have a fair bit of influence on discussions about learning, process, and practice. Later they would go on to publish a book based on this paper and, in that book, they would refine the model a bit to its current form, as shown on wikipedia. The model lists five phases of skill acquisition: Novice, Advanced Beginner, Competent, Proficient and Expert. There's obviously a lot to it, since it takes an entire book to describe it, but the gist of it is that skill acquirers move from "dogmatic following of rules and lack of big picture" to "intuitive transcending of rules and complete understanding of big picture."

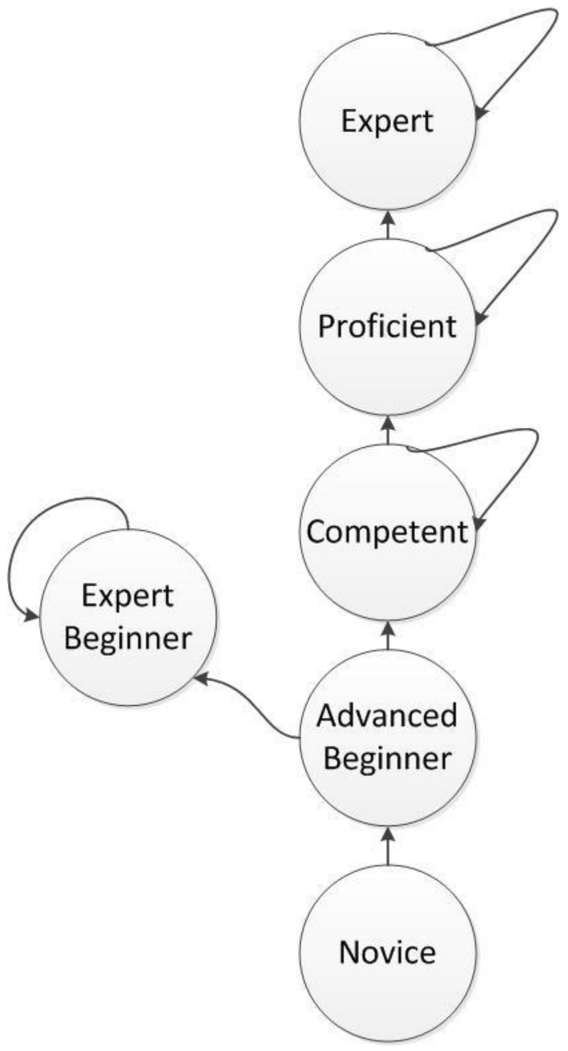
All things being equal, one might assume that there is some sort of natural, linear advancement through these phases, like earning belts in karate or money in the corporate world. But in reality, it doesn't shake out that way, due to both perception and attitude. At the moment one starts acquiring a skill, one is completely incompetent, which triggers an initial period of frustration and being stymied while waiting for someone, like an instructor, to spoon-feed process steps to the acquirer (or else, as Dreyfus and Dreyfus put it, they "like a baby, pick it up by imitation and floundering"). After a relatively short phase of being a complete initiate, however, one reaches a point where the skill acquisition becomes possible as a solo activity via practice, and the renewed and invigorated acquirer begins to improve quite rapidly as he or she picks "low hanging fruit." Once all that fruit is picked, however, the unsustainably rapid pace of improvement levels off somewhat, and further proficiency becomes relatively difficult from there forward. I've created a graph depicting this (which actually took me an embarrassingly long time because I messed around with plotting a variant of the logistic 1/(1 + e^-x) function instead of drawing a line in Paint like a normal human being).



This is actually the exact path that my bowling game followed in my path from bowling incompetence to some degree of bowling competence. I rapidly improved to the point of competence and then completely leveled off. In my case, improvement hit a local maximum and then stopped altogether, as I was too busy to continue on my path as-is or to follow through with my retooling. This is an example of what, for the purposes of this post, I will call “arrested development.” (I understand the overlap with a loaded psychology term, but forget that definition for our purposes here, please.) In the sense of skills acquisition, one generally realizes arrested development and remains at a static skill level due to one of two reasons: maxing out on aptitude or some kind willingness to cease meaningful improvement.

For the remainder of this post and this series, let’s discard the first possibility (since most professional programmers wouldn’t max out at or before bare minimum competence) and consider an interesting, specific instance of the second: voluntarily ceasing to improve *because of a belief that expert status has been reached and thus further improvement is not possible.*. This opting into indefinite mediocrity is the entry into an oblique phase in skills acquisition that I will call “Expert Beginner.”

The Expert Beginner



When you consider the Dreyfus model, you'll notice that there is a trend over time from being heavily rules-oriented and having no understanding of the big picture to being extremely intuitive and fully grasping the big picture. The Advanced Beginner stage is the last one in which the skill acquirer has no understanding of the big picture. As such, it's the last phase in which the acquirer might confuse himself with an Expert. A Competent has too much of a handle on the big picture to confuse himself with an Expert: he knows what he doesn't know. This isn't true during the Advanced Beginner phase, since Advanced Beginners are on the "unskilled" end of the Dunning Kruger Effect and tend to epitomize the notion that, "if I don't understand it, it must be easy."

As such, Advanced Beginners can break one of two ways: they can move to Competent and start to grasp the big picture and their place in it, or they can 'graduate' to Expert Beginner by assuming that they've graduated to Expert. This actually isn't as immediately ridiculous as it sounds. Let's go back to my erstwhile bowling career and consider what might have happened had I been the only or best bowler in the alley. I would have started out doing poorly and then quickly picked the low hanging fruit of skill acquisition to rapidly advance. Dunning-Kruger notwithstanding, I might have rationally concluded that I had a pretty good aptitude for bowling as my skill level grew quickly. And I might also have concluded somewhat rationally (if rather arrogantly) that me leveling off indicated that I had reached the pinnacle of bowling skill. After all, I don't see anyone around me that's better than me, and there must be some point of mastery, so I guess I'm there.

The real shame of this is that a couple of inferences that aren't entirely irrational lead me to a false feeling of achievement and then spur me on to opt out of further improvement. I go from my optimistic self-assessment to a logical fallacy as my bowling career continues: "I know that I'm doing it right because, as an expert, I'm pretty much doing everything right by definition." (For you logical fallacy buffs, this is circular reasoning/begging the question). Looking at the graphic above, you'll notice that it depicts a state machine of the Dreyfus model as you would expect it. At each stage, one might either progress to the next one or remain in the current one (with the exception of Novice or Advanced Beginner who I feel can't really remain at that phase without abandoning the activity). The difference is that I've added the Expert Beginner to the chart as well.

The Expert Beginner has nowhere to go because progression requires an understanding that he has a lot of work to do, and that is not a readily available conclusion. You'll notice that the Expert Beginner is positioned slightly above Advanced Beginner but not on the level of Competence. This is because he is not competent enough to grasp the big picture and recognize the irony of his situation, but he is slightly more competent than the Advanced Beginner due mainly to, well, extensive practice being a Beginner. If you've ever heard the aphorism about "ten years of experience or the same year of experience ten times," the Expert Beginner is the epitome of the latter. The Expert Beginner has perfected the craft of bowling a 160 out of 300 possible points by doing exactly the same thing week in and week out with no significant deviations from routine or desire to experiment. This is because he believes that 160 is the best possible score by virtue of the fact that he scored it.

Expert Beginners in Software

Software is, unsurprisingly, not like bowling. In bowling, feedback cycles are on the order of minutes, whereas in software, feedback cycles tend to be on the order of months, if not years. And what I'm talking about with software is not the feedback cycle of compile or run or unit tests, which *is* minutes or seconds, but rather the project. It's during the full lifetime of a project that a developer gains experience writing code, source controlling it, modifying it, testing it, and living with previous design and architecture decisions during maintenance phases. With everything I've just described, a developer is lucky to have a first try of less than six months, which means that, after five years in the industry, maybe they have ten cracks at application development. (This is on average–some will be stuck on a single one this whole time while others will have dozens.)

What this means is that the rapid acquisition phase of a software developer–Advanced Beginnerism–will last for years rather than weeks. And during these years, the software developers are job-hopping and earning promotions, especially these days. As they breeze through rapid acquisition, so too do they breeze through titles like Software Engineer I and II and then maybe "Associate" and "Senior," and perhaps eventually on up to "Lead" and "Architect" and "Principal." So while in the throes of Dunning-Kruger and Advanced Beginnerism, they're being given expert-sounding titles and told that they're "rock stars" and "ninjas" and whatever by recruiters–especially in today's economy. The only thing stopping them from taking the natural step into the Expert Beginner stage is a combination of peer review and interaction with the development community at large.

But what happens when the Advanced Beginner doesn't care enough to interact with the broader community and for whatever reason doesn't have much interaction with peers? The Daily WTF is filled with such examples. They fail even while convinced that the failure is everyone else's fault, and the nature of the game is such that blaming others is easy and handy to relieve any cognitive dissonance. They come to the conclusion that they've quickly reached Expert status and there's nowhere left to go. They've officially become Expert Beginners, and they're ready to entrench themselves into some niche in an organization and collect a huge paycheck because no one around them, including them, realizes that they can do a lot better.

Until Next Time

And so we have chronicled the rise of the Expert Beginner: where they come from and why they stop progressing. In the next post in this series, I will explore the mechanics by which one or more Expert Beginners create a degenerative situation in which they actively cause festering and rot in the dynamics of groups that have talented members or could otherwise be healthy.

How Software Groups Rot: Legacy of the Expert Beginner

Edit: The E-Book is now available. Here is the publisher website which contains links to the different media for which the book is available.

Sign up for my mailing list, and get about 10 posts' worth of content, excerpted from my latest book as a PDF. (Don't worry about signing up again -- the list is smart enough to keep each email only once.)

Items added