

Forum:[Hack Forums](#)

Thread: [The Stages of Domain Knowledge: Using the Dreyfus Model of Skill Acquisition](#)

Haxalot - May 24, 2014, 12:45 pm

The Stages of Domain Knowledge:
Using the Dreyfus Model of Skill Acquisition

"Ignorance more frequently begets confidence than does knowledge." -Charles Darwin

Throughout our careers, we'll acquire a number of different skills. For us programmers (to whom this article is tailored towards), we'll most probably learn multiple programming languages and be forced to keep up with the latest developments in those languages (unless you intend on teaching, ahem). There are no formal methods of measuring our knowledge of a particular domain - that is to say, no event theories in place that can actually 'measure' how knowledgeable we are in a particular field. Instead, there's only construct theories that can be used to help us rate our abilities.

One construct theory is that of the Dreyfus model, and that's what we're going to be looking at throughout the rest of this article. We will firstly look at a bit of history with the Dreyfus model, by seeing where it has been successfully applied to in the past. We will then go through each of the five stages outlined by the Dreyfus model: novice, advanced beginner, competent, proficient, and finally expert. Lastly, we'll see how we can move up through the stages of the model to the final and most sought-after stage (expert). Throughout this article, we will also see what prevents us from moving up to the higher stages, and how we can overcome these hurdles.

Definitions:
Event TheoryA theory that can be measured - that is to say, it can be proven and judged by its accuracy.

Construct TheoryA theory that makes for an abstraction. They cannot be measured, but rather are evaluated in terms of their usefulness. These types of theories are just ideas and concepts, and so it makes no sense to try to 'prove' them.

Historical Values

As defined above, construct theories are evaluated in terms of their usefulness. It therefore makes sense for us to firstly look at where the Dreyfus model has been successfully applied to in the past and what effect its application has had.

In the late 1970s, the US medical profession was experiencing a number of problems. Many of these problems derived from the widely taught and ubiquitously adopted practices. One of these dogmatic practices was ensuring that nurses only carried out orders given by doctors. This was problematic because doctors weren't always (or more accurately, were rarely) free to perform continuous check-ups upon all of their patients, and so they missed many of the subtle changes in their health (and therefore changes in their medical needs). Even though the nurses, who regularly tended the patients, noticed these subtle changes, they diligently went against their own intuition to follow the orders given by the doctors (due to common practice). This lack of responsibility given to nurses inadvertently made patient care worse, and as a result, the true goal of getting patients better was lost for practice over patient welfare. Things needed to be changed.

In the early 1980s, the Dreyfus model was applied to the nursing profession. Nurses were given the ability to use their own intuition when caring for their patients as their medical conditions (and therefore needs) changed. This meant that they no longer carried out only orders given by doctors, but rather used their knowledge and experience to suggest different medical requirements - thus

patient care improved. The encouragement in leveraging nursing experience instead of giving them context-free rules to follow ultimately placed patient welfare over common practice, which was after all the intended goal of patient care. Doctors still gave their (valuable) input based upon their intuition, but they were no longer definitive orders that couldn't be changed.

With the above scenario in mind, we will now look at each of the stages proclaimed by the Dreyfus, along with the changes that occur when passing through each of these stages.

The Five Stages

The Dreyfus model states that there is a five-stage scale when it comes to human learning of a particular skill. It shows that when moving up through each of these stages, you don't just gain more knowledge or experience, but rather that you also undergo a number of other changes too. Such changes include how you perceive problems and how you go about approaching them, as well as the mental models you form and use during problem solving. Each stage in the model attempts to describe how and why our abilities, attitudes, capabilities, and perspectives change, as we're about to see.

Stage 1 - Novice The first stage of the Dreyfus model is that of the novice. Whenever you venture into a completely new field, you'll start on this stage.

A novice will have little or no experience in a particular domain. This means that everything they attempt to do will have a novel feeling to them. They are usually concerned with their ability to succeed in the new, alien-like domain, and whilst they don't particularly want to learn much, they seek to accomplish a set goal. Because of their lack of experience, when things go awry they are often left confused and not sure what to do.

As programmers, we can relate to this. When beginning programming, we all had idealistic views of projects we were going to undertake to make ourselves wealthy, successful, or well-known (only to have reality knock us to our senses). We were very goal-directed, and whilst we were motivated to achieve our goal, we weren't overly interested in learning the ins-and-outs of the language at hand to get there. This naivety in thinking is all-too common for the novice.

Whilst novices don't really have any measure of capability, they can be given some capability through following context-free rules (think PC World support).

Stage 2 - Advanced Beginner Next up, we have the advanced beginner. At this stage, we begin to try some new things out, but we still have difficulty in getting past problems encountered. We still don't want to learn the mundane theory yet, but we also don't want to be retaught the absolute basics again. Holistic understanding is still non-existent, and so the context of problems still isn't considered too important when problem solving since we simply don't have the past experience yet.

Again, as programmers we should know this feeling. We've overcome the novice stage of not really knowing what our code is doing, to having some idea of what's going on but without knowing (or caring about) the bigger picture of things - there's a lack of having a holistic understanding of the language. We understand the syntax of our language, can implement its basic constructs, and can create basic applications. But we still cannot, however, quickly problem solve with our language because of our lack of experience, and cannot leverage more intricate features of the language and its libraries.

Whilst we still like to follow simple, context-free rules, we can begin to deviate from these a little.

It's also worth noting that this category caters for the largest group of people on the stages

spectrum. Many people will only ever reach this far, usually because of second-order incompetence (where you're unskilled and unaware of it, and so you have an inflated opinion of your ability).

Stage 3 - Competent The third stage is having competence in a skill. At this level, we have the ability to more swiftly solve novel problems (partly because of past experience) by ourselves and can apply advice given by others somewhat effectively. We can develop conceptual models of the problem at hand, and can work with those mental models effectively - we're becoming better problem solvers. At this stage we seem to be a lot more independent. Unfortunately, the ability of self-correction is still lacking, and we still can't really reflect upon past mistakes.

For programmers at this level, they can look into a variety of the libraries and technologies available for their language. They can problem solve with these new technologies, and can leverage some of the more complex features of a language. But because they still lack much of a holistic understanding, they aren't able to integrate the usage of multiple technologies together for a better-designed and more robust application.

Having a holistic understanding with the ability to perform self-correction and be able to reflect upon past mistakes is still lacking or non-existent. Also, metacognitive abilities (the ability of being self-aware) are still very much underdeveloped. This often means that people of this level (and even more so of the lower levels) generally have an inflated opinion of their own skills and so will usually end up overestimating themselves. The only way to overcome this problem (often referred to as second-order incompetence) is to move further up the stages in the Dreyfus model.

Stage 4 - Proficient At the proficient stage, we have a need for a holistic understanding. We will seek out and want to understand the bigger picture of our skill and will become frustrated when presented with oversimplified information. We have the ability to correct previous poor performance on tasks through being able to reflect upon our approaches to them and revise them for a better performance next time. We can also learn effectively from the experience of others, such as when hearing about another's past project problems and being able to aptly apply the solutions to their issues within our own projects. Furthermore, we can reflect upon feedback given to us and take full advantage of it to better ourselves.

For programmers, this means that we can leverage more advanced software development techniques. This can include being able to appropriately apply software design patterns, use programming methodologies and adhere to their maxims within specific contexts, leverage arcane features of your language, solve novel, intricate problems efficiently, and so on.

This is perhaps the largest step up in the Dreyfus model between each of the stages.

Stage 5 - Expert The final and most sought after stage is that of the expert. Experts are the primary sources of information in their field and always try to seek better ways to carry out tasks in the most efficient manner. They work intuitively and rely on knowing the full context of the problem to aptly apply the right solutions. Their vast experience enables them to perform targeted, focussed pattern matching, where they can simply tell what pieces of information are and are not pertinent to a given problem. It's almost as though they just instinctively know the right answer. Many will write books, speak at conferences, and innovate new technologies.

Experts are still, however, very cautious - especially when things go awry for unusual reasons. They will show far more self-doubt than someone lower down on the Dreyfus model because of their far more developed metacognitive abilities (also hence the quote at the top of this article).

Progressing Through the Five Stages

I know most of you who have bothered to read this far are probably wondering how you can move up through these stages as fast as possible, right? Well unfortunately, there are no short-cuts to get you to the final stages in a matter of weeks (or months, or even years). But with that said, you can help yourself to move up through the stages more easily by knowing what barriers lie between you

and the next stage and how to overcome these. I've spoken about many of these barriers in each of the Dreyfus model's five stages, so let's cover some new points about how to overcome these barriers.

Avoid overestimating your own skills because that will lead to complacency. Complacency comes from ignorance (of not knowing what you don't know) - which is a condition mentioned a few times above called second-order incompetence. It's far better to underestimate your abilities and to actively seek out gaps in your knowledge than to coerce yourself into thinking you know it all already. The classic example of this that many will have fallen victim to is overestimating your knowledge in HTML. Many proclaim to be expert in this skill, and even laugh at HTML's simplicity; but are you really a primary source of knowledge in the field of HTML, even enough to write a whole book about it? (See [The Definitive Guide to HTML5](#) - a 1080 page-long book written on HTML5 alone.) For example, do you know the various markup languages and how they can be embedded together, or even what the various versions of HTML bring? What about the plethora of [Web Standards](#) involved when writing in these markup languages? Or what about some of the more intricate features of them such as web sockets, web storage, the geolocation API, etc? Even with all of that knowledge behind you, you may not be considered expert in HTML.

Don't rush your learning process. It's a trap I've fallen into many times (and I still do, unfortunately). When trying to master knowledge, without gaining experience in the process, you'll simply forget that knowledge (unless you're blessed with a perfect memory). It's just not effective and you'll find yourself spending a lot of time learning, just to forget what you've learnt a month or two (if that) down the line. You have to reinforce your knowledge somehow, and you must do this continually or your risk forgetting it. Everyone has their own method of this (and if you don't yet, then try to find at least one ASAP). For myself, I enjoy writing articles about what I've learnt. I also practice a lot by writing code to solve particular problems. Doing this each day (even if it's just 30 minutes or so) will greatly improve your problem-solving skills (one of the most important skills for a programmer to have), whilst helping you to reinforce your knowledge and give you some experience points too.

Continually seek to evaluate your own work and look for improvements within your work flow. Self-assessment is a critical part of self-improvement and is very important when trying to progress through the stages of the Dreyfus model. Look for things that are time consuming in your work and try to automate them. Look over code you recently released and try rewriting it another way. Try out different tools that aren't in your de facto toolbox to see if they improve your efficiency when programming. There's a number of things you can do, so just give them a try and see what happens! At worst, you'll have wasted some time learning a (perhaps redundant) new skill.

The Cessation

Throughout this article, we've looked at the Dreyfus model and its five discrete stages, along with how we can better progress up through these stages when acquiring new skills. We've spoken about the curse of second-order incompetence and how to overcome this by developing better metacognitive abilities and actively seeking out what you don't know. We've also briefly looked at better retaining what you've learnt to prevent yourself from forgetting things soon after learning them. (A classic example everyone can relate to is revision-cramming, where you intensively revise something shortly before an exam, just to forget all of that knowledge in just a couple of weeks time.)

I hope this has given the people of HackForums a better idea of how you can not only rate each of your individual abilities (albeit subjectively, still), but also increase your proficiency in each of your skills with the aforementioned scale. I also hope this has shown why proclaiming to 'know' something is incredibly vague and subjective, therefore making it a rather useless statement because of the broadness of the spectrum involved.

If you have enjoyed reading my article, and would also like to read more about the process of learning and how to become a more efficient learner, then grab a copy of this book: [Pragmatic Thinking and Learning: Refactor Your Wetware](#). I have found it to be a very enjoyable and helpful read, and I'm sure you will too!

?????? - May 25, 2014, 8:21 pm

So far this is a great read and puts a lot of things into perspective. However, I have one question. What exactly do you mean by "holistic understanding"?

I'll write a proper reply once I finish - just wanted to ask that question.

We need more of these types of articles around here.

~Dec - May 25, 2014, 10:11 pm

(May 25, 2014 08:21 PM)?????? Wrote: [\[CITE\]](#) So far this is a great read and puts a lot of things into perspective. However, I have one question. What exactly do you mean by "holistic understanding"?

I'll write a proper reply once I finish - just wanted to ask that question.

We need more of these types of articles around here.

When you talk of a holistic understanding, you essentially mean that you do not only understand the concept as a whole, or what parts of the whole concept are necessary, but how each and every part/step/process of the whole concept fits together.

?????? - May 26, 2014, 4:26 am

(May 25, 2014 10:11 PM)~Dec Wrote: [\[CITE\]](#) When you talk of a holistic understanding, you essentially mean that you do not only understand the concept as a whole, or what parts of the whole concept are necessary, but how each and every part/step/process of the whole concept fits together.

I see, thank you.

The article itself was a great read and brings one back to his senses, in a way. I was mostly thinking of myself and Python while reading - it appears that I'm somewhere in between competent and proficient. I don't want to overestimate myself, but... The first two paragraphs of competent apply to me, even though I'm a bit higher than that. But I'll call it competent so as not to overestimate. However, I feel that I'm not completely lacking in a holistic understanding of the language and in the ability to reflect and self-correct. In that sense, I would say I'm proficient.

However, skill-wise, I'm probably still only competent.

I see lots of people boasting about how they are experts at something or at least proficient at something. I usually laugh it off because I know of only a few people on the forum who are indeed approaching real proficiency. Hard to find such people because it seems to be in the human nature to overestimate oneself.

Anyways, thanks for the read. Great article. Really helped put things into perspective and allowed for a self-evaluation.

Haxalot - May 26, 2014, 8:09 am

(May 26, 2014 04:26 AM)?????? Wrote: [\[CITE\]](#) I see, thank you.

The article itself was a great read and brings one back to his senses, in a way. I was mostly thinking

of myself and Python while reading - it appears that I'm somewhere in between competent and proficient. I don't want to overestimate myself, but... The first two paragraphs of competent apply to me, even though I'm a bit higher than that. But I'll call it competent so as not to overestimate. However, I feel that I'm not completely lacking in a holistic understanding of the language and in the ability to reflect and self-correct. In that sense, I would say I'm proficient.

However, skill-wise, I'm probably still only competent.

I see lots of people boasting about how they are experts at something or at least proficient at something. I usually laugh it off because I know of only a few people on the forum who are indeed approaching real proficiency. Hard to find such people because it seems to be in the human nature to overestimate oneself.

Anyways, thanks for the read. Great article. Really helped put things into perspective and allowed for a self-evaluation.

Thanks for the comment. When learning about the Dreyfus model, I too felt that whilst I could reflect upon and correct my previous performance, I didn't completely fit into the category of proficiency for my particular skill. I think we cannot be cleanly bound to a particular category because we develop parts of the same skill at different levels, and therefore we find ourselves overlapping to the next level before having finished the current level we are at.

This is where the value of underestimating yourself comes in. It prevents you from thinking that you are better than you really are, and so you avoid complacency when building up a new skill.

?????? - May 26, 2014, 1:31 pm

(May 26, 2014 08:09 AM)Haxalot Wrote: [\[CITE\]](#) Thanks for the comment. When learning about the Dreyfus model, I too felt that whilst I could reflect upon and correct my previous performance, I didn't completely fit into the category of proficiency for my particular skill. I think we cannot be cleanly bound to a particular category because we develop parts of the same skill at different levels, and therefore we find ourselves overlapping to the next level before having finished the current level we are at.

This is where the value of underestimating yourself comes in. It prevents you from thinking that you are better than you really are, and so you avoid complacency when building up a new skill.

I was going to mention that but I forgot. Indeed, it's hard to place someone into strictly one category. Actual skill and the understanding of your skill are both interrelated, but they may not always be on the same point in the Dreyfus scale.

Skyla Blue - May 26, 2014, 6:23 pm

Very nice article you've got here. I do have one question though. Where do the seemingly smart, yet extremely douchey people fit in that model? Are they proficient? Are they experts? Are they simply novices with big egos? I've encountered several people who are like that.

EDIT: Also, what are your thoughts on child prodigies? Are they legit? I don't remember his name off hand, but some kid makes quite a bit of money selling iPad/iPhone apps, and he is 15 (at the time of the Youtube video, so he is probably 16 or 17 now). I looked at some of his apps, and they were extremely basic, a calculator, some organiser thing, etc. Maybe he's selling some super crazy stuff on the underground, but from what I saw he's no prodigy programmer.

lalalox4 - May 27, 2014, 3:08 pm

Very interesting article, I greatly admire your skill in the learning field and your will to share knowledge about it.

Haxalot - May 27, 2014, 3:38 pm

(May 26, 2014 06:23 PM)Skyla Blue Wrote: [\[CITE\]](#) Very nice article you've got here. I do have one question though. Where do the seemingly smart, yet extremely douchey people fit in that model? Are they proficient? Are they experts? Are they simply novices with big egos? I've encountered several people who are like that.

It's difficult to say based upon someone's personality (and I'm not quite sure how you're exactly defining 'douchey'). You have answered your own question though - they could be experts, proficient, or more simply egotistical people on the lower end of the Dreyfus model (though if they're seemingly smart, then they're unlikely to be novices).

(May 26, 2014 06:23 PM)Skyla Blue Wrote: [\[CITE\]](#) EDIT: Also, what are your thoughts on child prodigies? Are they legit? I don't remember his name off hand, but some kid makes quite a bit of money selling iPad/iPhone apps, and he is 15 (at the time of the Youtube video, so he is probably 16 or 17 now). I looked at some of his apps, and they were extremely basic, a calculator, some organiser thing, etc. Maybe he's selling some super crazy stuff on the underground, but from what I saw he's no prodigy programmer.

Some people do just naturally take to particular things (especially if they're passionate about it). Whilst people of that age wouldn't eligible for the expert stage (because of their obvious lack of experience given their age), they could easily be at the proficiency stage.

--

(May 27, 2014 03:08 PM)lalalox4 Wrote: [\[CITE\]](#) Very interesting article, I greatly admire your skill in the learning field and your will to share knowledge about it.

Thanks for the comment. I'm glad you enjoyed the article.