Create an Apprenticeship Program

Bridge Your Talent Gap

David H. Hoover[1], dave@apprentice.us, June 18, 2012

You are faced with a problem. You need people who understand how to create and grow the software that will grow your business. You need them now, but they're out of reach.

The bad news is that you're probably not going to find them right now. The good news is that with the rise of "Hacker Academies" like Code Academy, Dev Bootcamp, and Hacker School, it is becoming easier to find world-class beginners. The great news is that with the right approach and ingredients, you can grow these high-potential beginners into effective software developers.

Universities are the typical place that established businesses expect to find these high-potential beginners. While many software developers finish college with a good education, they're often burned out, deep in debt, and understandably eager to cash in on their hard work. Apprentices, on the other hand, inject enthusiasm, hard work, and a thirst for knowledge into your teams. They will consistently launch from your apprenticeship program with context, momentum, and loyalty to your organization.

A Young Apprentice's Journey

Jacob "JR" Richardson was a wrestler for Waukee High School in Waukee, Iowa. While he had always been interested in computers, the thought of working with them all the time wasn't appealing. Then, a teacher urged him to get involved with an after-school technology club where he rediscovered his passion for programming. JR graduated in 2010, and instead of going to college, he chose to continue his local programming internship, and soon after found a full-time job programming at IMT Insurance. About a year later, JR heard about Groupon's apprenticeship program in Chicago. He applied, was accepted, and moved into the Groupon-provided apartment in October 2011. Six months later, he graduated from the program, accepted a full-time job at Groupon, and chose to relocate to Boulder to work with Groupon's Denver team. Through his apprenticeship, he developed expertise in abstract syntax trees, lazy collections, and static analysis. He is currently building static analysis tools to help Groupon's developers and release engineers maintain a stable production environment. Unfortunately, most of JR's fellow class of 2010 are struggling to find similar success.[2]

---

1 Editorial contributions by Dale J. Stephens, Michael Staton, Cory Flanigan, and Kevin Taylor

2 http://www.nytimes.com/2012/06/06/business/more-young-americans-out-of-high-school-are-also-out-of-work.html

This document provides a brief overview of a the **7 ingredients** you can use to grow game-changing software developers. I use the label "apprenticeship" to describe this approach, which comes from the literature and methods of the software craftsmanship movement. Over my decade of experience with software apprenticeship programs at both public and private companies, I have observed that it's possible to *consistently* grow viable software developers despite highly varied academic and professional backgrounds.

1. Mentor > Team > Business Owner

To create an apprenticeship program, you need to have at least one viable mentor on staff. Finding viable mentors tends to be the limiting factor when people are looking to start or expand their apprenticeship programs. That said, a mentor is just one of the foundations of an apprenticeship:

1) a mentor who is excited to work with a beginner
2) a team that is willing to incorporate a beginner
3) business owners who are willing to allow beginners to participate

The mentor doesn't necessarily need to be a master of software development. In fact, there are advantages to having a mentor who is closer in proximity to the apprentice's level of experience. That said, if the mentor is relatively inexperienced, it's important that there are other more senior practitioners nearby.

2. Sustainable Ratio

Having more apprentices than experienced practitioners is dangerous. Assuming every practitioner in your organization is ready to take on an apprentice, exceeding a 1:1 ratio on an ongoing basis is ill advised. While some experienced mentors will be able to take on multiple apprentices at once, this should not be expected. Furthermore, all mentors will need to periodically take some time off from mentoring in order to focus on other aspects of their practice and career. Due to these dynamics, as your organization approaches a 1:1 ratio, you will inevitably have apprentices working alone or with each other, and this is where good software goes bad, bad software gets worse, and apprenticeship programs falter. Be patient. Even a 2:1 non-apprentice:apprentice ratio will grow your company by 50% relatively quickly.

3. Culture over Curriculum

It's tempting to approach apprenticeship with the same mindset that we approach the more familiar classroom-based education. That is, in preparation for an apprenticeship program, many people work long and hard to create awesome and extensive beginner-oriented curriculums. Don't do this. If you have the ability to create an awesome curriculum ahead of time, you should also be able to guide apprentices toward a more personally customized set of topics and resources when

the time is right for them. Instead of focusing on curricula, work to ensure that your organization's culture is welcoming and healthy for beginners.

If it feels like your culture has a long way to go to incorporate apprentices, consider seeding it with "experienced" apprentices coming from established cultures like the programs at 8th Light and apprentice.io, along with the aforementioned "Hacker Academies". These people, if given sufficient mentoring and encouragement, will help to invigorate your organization with their passion and expectations for continuous growth and learning.

It's critical that apprentices continue to learn at a rapid pace in the months and years after they've completed an apprenticeship program. Healthy cultures facilitate this ongoing learning, while curriculums do not. The simplest way to get started in creating a learning-oriented culture is with lunchtime study groups. As a leader in those groups, the surest way to make beginners feel safe is to expose your ignorance by asking the "stupid" or obvious questions yourself. A beginner-friendly, learning-oriented culture takes a lot of the pressure off of mentors, and is the most sustainable way to grow apprentices.

4. In The Trenches

Similar to the mentor/apprentice relationship, hands-on, real-world learning is fundamental to apprenticeship. Apprentices need to learn on real projects, with real problems, where project leaders are regularly balancing team size, deadlines, and scope. When a beginner starts an apprenticeship program, the trenches will feel overwhelming. To them, it may feel more like they're *next* to the trenches rather than *in* them. During the first weeks, and possibly months of a program, it can be beneficial to give the the apprentice a few hours a day to study and practice *next* to the trenches. Over the course of the program, though, a successful apprentice will begin feeling more comfortable, and by the end, the successful apprentice should be spending the full workday in the trenches, outperforming some of their entry-level peers on certain tasks. Their progress toward this point should be the primary measure of success at each of the milestones.

5. Pet Project

While learning "in the trenches" is key, it's important for apprentices to have a project of their own to work on independently. This is a "breakable toy" in that they get to choose their own (often bleeding edge) technologies and are free to take risks with new approaches. Apprentices often struggle to come up with a pet project to work on, and this is where an experienced mentor, or other members of the team can help with some initial ideation and planning. The critical attribute of an apprentice's pet project is that they're genuinely interested in the problem that the project is solving. They will be spending many, solitary late nights on the project. The more excited they are about it, the more likely their success.

6. Milestones

People grow attached to their apprentices, so it's important to have a predefined schedule that periodically forces a decision to hire, continue with, or fail the apprentice. Furthermore, it is beneficial to have an oversight group of 4-8 people who weigh in on the hire/continue/fail decision so that it's not entirely in the hands of the mentor.

Use these milestones as opportunities for the apprentice to demonstrate their progress. This is a natural opportunity for the apprentice to put her best foot forward in anticipation of the big decision. The oversight group should expect to "have their socks knocked off" by the apprentice's progress.

The sequence of a typical milestone is:

1. Apprentice gives a brief experience report about her work in the trenches.
2. Apprentice provides a demonstration of the pet project.
3. Facilitator leads a code review of the pet project.
4. Apprentice presents for 10 minutes on a technology she has recently learned.
5. Facilitator leads a post-mortem.
6. Oversight group makes hire/continue/fail decision.
7. Mentor communicates decision to apprentice.

Plan for 6-month apprenticeships, with milestones every 2 months. You can obviously customize the duration of the apprenticeship and the frequency of the milestones, but it unadvisable to customize per apprentice. By sticking with a standard structure, it will make your hiring and milestone decisions simpler as you develop a feel for the characteristics of successful candidates and apprentices.

The final milestone hire/fail decision should be based on the entry-level hiring criteria for your organization: Has the apprentice surpassed the expectations you have for your entry-level hires? Having some existing entry-level developers in the organization to use as a baseline makes this a lot easier. It's critical to stand firm on the timing of the final milestone for the hire/fail decision, even if you deviate from the recommended 6-month schedule.

7. Feedback Loops

Without feedback, both apprenticeships and apprenticeship programs become stagnant. Therefore, it's helpful to have frequent opportunities for feedback, as well as feedback at multiple levels.

Start by attaching a post-mortem to each milestone. The oversight group, along with the mentor and apprentice, participate in the session, which is an opportunity to hear feedback from all parties about a) what worked well, b) what didn't work well, c) what questions need exploring, and d) action items. This is where a lot of program improvements and adaptations happen. This is also an opportunity to give

an apprentice a clear warning that things need to improve before the next milestone. If things do not improve, the decision to eventually fail the apprentice is much more straightforward.

It's important to have smaller feedback loops in between the milestones, since they are 2 months apart. Therefore, the mentor and the apprentice should meet briefly every week to focus solely on communicating about progress, issues, and upcoming plans. This information should flow in both directions, and often involves more input from the apprentice than from the mentor.

The tightest feedback loops happen in the trenches. Apprentices work side-by-side with more experienced practitioners, and should be given significant time to pair program and have their code reviewed by their teammates. These feedback loops happen on a daily basis, providing the underlying momentum for the apprentice's progress.

Apprenticeship programs vs. Internships

There is an understandable confusion regarding apprenticeship and internships. I'll state the two extremes of each to help clarify the distinction.

Apprenticeship programs require high-commitment from everyone involved, and always involve a strong dose of mentoring. They exist to transition someone from pre-employability to full-time employment over the course of 6 months. A successful apprenticeship ends with a job offer and acceptance.

Internships are typically geared for university students who are on summer break. These students will go back to school in the Fall and therefore have no expectation that they will work their way into a full-time role. An internship typically lasts just a couple months and is meant as a low-commitment, exploratory exercise.

In reality, the line between the two concepts can be blurry in the cases where internships have strong mentoring and lead directly to a job. The label is less important than creating a sustainable learning structure.

Conclusion

In the United States, a proliferation of apprenticeship programs would be a boon for the 1 in 5 people between the ages 18-24 who are unemployed[3], as well as the astounding 50% of US employers who are unable to fill critical jobs[4]. Countries like Germany, who have integrated apprenticeships into their high schools and universities, are not experiencing the same talent gap.[5] There are thousands of high potential people who will jump at the opportunity to get their foot in the door of

3 http://blogs.reuters.com/felix-salmon/2011/12/22/the-global-youth-unemployment-crisis/

4 http://www.chicagobusiness.com/article/20120529/NEWS08/120529814/a-good-tech-guy-or-gal-is-hard-to-find

your software development organization. I've seen what they're capable of, and the value that they can quickly add to your business with the right structure.

If you're interested in finding out more about apprenticeship, have your own ideas about how to make it happen, or need help starting a program, send an email to support@apprentice.us.

If you aspire to be a great software developer, I suggest reading the book I co-authored, "Apprenticeship Patterns: Guidance for the Aspiring Software Craftsman".

Please visit apprentice.us and thank your mentors!

5 http://www.nationaljournal.com/magazine/workers-without-right-education-skills-floundering-in-weak-economy-20110728