Is it possible manage developers with high turnover if you can't lower the turnover rate? [closed]

Asked 15 years, 10 months ago Modified 13 years ago

Viewed 859 times

2

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 5 years ago.

Improve this question

I lead a small group of programmers in a university setting, having just moved into this position last year. While the majority of our team are full time employees, we have a couple of people who are traditionally graduate assistants.

The competition for these assistantships is fairly intense, as they get free graduate school tuition on top of their salary while they have the job. We require that they sign up for at least a year, though we consider ourselves lucky

if they stay for two. After that, they get their master's degree and move on to bigger and better things.

As you can imagine, hiring and re-training these positions is time- and resource-intensive. To make matters worse, up to now they have typically been the sole developer working on their respective projects, with me acting in an advisory and supervisory role, so wrangling the projects themselves to fight the entropy as we switch from developer to developer is a task unto itself.

I'm tempted to bring up to the administrators the possibility of hiring a full- (and long-haul) developer to replace these two positions, but for a school in a budget crisis, paying for two half-time graduate assistants is far cheaper (in terms of salary and benefits) than paying for one full-time developer. Also, since I'm new to this position, I'd like to avoid seeming as though I'm not able to deal with what I signed up for. For the forseeable future, I don't think the practice of hiring short-term graduate assistants is going to change.

My question: **What can I do to create an effective training program considering that the employees may be gone after as little as a year on the job?**

- How much time should I invest in training them, and how much would simply be a waste of time?
- How much time should they take simply getting acclamated to our process and the project?

- Are there any specific training practices or techniques that can help with this kind of situation?
- Has anyone dealt with a similar situation before?
- Do I worry too much, or not enough?

By the way, and for the record, we do the vast majority of our development in Perl. It's hard to find grad students who know Perl, while on the other hand everybody seems to have at least an academic understanding of Java. Hence this question which I asked a while back.

project-management

Share

Improve this question

Follow

edited May 23, 2017 at 11:47

Community Bot

1 ● 1

asked Feb 12, 2009 at 15:43

Adam Bellaire

110k ● 19 ● 152 ● 165

8 Answers

Sorted by: Highest score (default)

5

Why don't you ask the students what they find difficult and make cheat sheets, lectures, etc. for the parts of the job that they have trouble with? Maybe you need to create some introductory Perl lectures or purchase some dead trees. How about a Safari subscription at O'Reilly?

I'd ask the students how they prefer to learn, though, before embarking on a training project. Everyone has different learning styles.

I'd also spend some time and capital creating a culture of professional software development at work. It'll be tough since academic programmers are often neophytes and used to kludging up solutions (I'm an academic programmer, btw) but the students will thank you in the long run. Maybe you can all go out to lunch once a week to discuss programming and other topics. You might also want to take some time to do code reviews so people can learn from each other.

With high turnover you definitely need to ensure that knowledge transfer occurs. Make sure you are using source code control and that your students understand proper commenting. I'd also make the students create brief documentation for posterity. If they are getting credit, make them turn in a writeup of their progress once a semester. You can put this in a directory in the project's repository for anyone who inherits it. As mentioned in other posts, a group wiki can really help with knowledge transfer. We use Mediawiki in our group and like it a lot.

One last thing I should add is that I find it helps to keep a list of projects for new developers that relatively easy and can be completed in a month or so. They are a great way for new people to get acclimated to your development environment.

Share Improve this answer Follow

edited Feb 12, 2009 at 16:18

answered Feb 12, 2009 at 16:12

Dana Robinson

4,364 ● 8 ● 34 ● 41

3

1. This is a relative question, and should be taken on a case-by-case basis. If the new hire already knows Perl, you don't need to go over this piece of training (yes, you could put Perl as a mandatory prerequisite, but that would significantly limit your applicant pool), and their first bit of training should be something like fixing a bug in an existing application or walking them through an application they will maintain. Though, given that the developers are only there for a year makes me think the development styles are going to vary some (if not a lot).
2. Getting the new person up to speed with your process is very important, as long as your process *works*. In this high turnover environment, you should put a strong emphasis on documentation in your process. A Wiki is a great thing to have for this documentation, since it's centralized and any of the developers can access it. Having them try to figure out how a project works by themselves (with little to no documentation) is a waste of both their time and your time.

Share Improve this answer

Follow

answered Feb 12, 2009 at 16:04

Ryan Abbott

5,397 ● 7 ● 33 ● 34

3

Perhaps I'm reading too much into the question, but if your university teaches java, why are you using Perl? Wouldn't it make more sense to use the tools that your students already know? This alone would cut the learning curve significantly. [once you eliminate the legacy code of course]

other than that, try:

- break the projects up into month-sized bites
- overlap the internships by at least 2 months, if not 6, so the new guy can work with and be trained by the old guy(s)
- document whatever repeatable processes you have (as was suggested by Mark Nold)
- if the grad students are cheaper than full-time pros, quit whinin' ;-) If not, go for the pros.

Share Improve this answer

Follow

answered Feb 13, 2009 at 6:08

Steven A. Lowe

61.1k ● 19 ● 135 ● 204

That's probably a good suggestion for the future, but for now the projects are in Perl and it's not trivial to switch. At the moment, the language barrier is not the main issue (it was included as an afterthought), so I don't think that the benefit

of eliminating it is worth the cost of switching.
– Adam Bellaire Feb 13, 2009 at 13:07

3

Have you considered making a "three ring binder" like Macdonalds and many other high turn over industries have? Have one folder which you can print out and hand to the new hire which shows the new hire some basics of getting up and running with Perl in your environment. This should be a "hello world", plus some basic regex and array manipulation. Lastly your manual should go on to show examples of the 5 things you find yourself doing all the time.

The example code may be authenticating users against an external security system, walking through recordsets or using ghostscript to create PDFs. Whatever they are, they should cover the basics of what you meet 80% of the time. More importantly the examples should show users how you expect the code to be written for clarity (eg: naming and approach), and give them some insight into servers and software in use and other practicalities which a generic book won't show them.

You won't get the binder right first time, but since you have a high staff turn over, you'll have plenty of time to test and improve it.

On top of this i would pick a single Perl programming book and give the new user their own copy of your three ring binder, plus "Programming Perl" to keep on their first

day. At a cost of $50 per hire i'm sure it's a lot cheaper than the alternatives and you'll have them flipping burgers.... i mean cutting code in no time.

Share Improve this answer Follow

edited Feb 13, 2009 at 21:08

answered Feb 13, 2009 at 5:58

Mark Nold

5,698 ● 7 ● 33 ● 34

2

My initial couple of thoughts are that you should:

1. hire for the position, i.e. it's Perl-centric so make that a big part of the pre-requisites. That way you don't need that piece of training as well.
2. invest time in the on-boarding process, maybe use a wiki so that you can easily update it to help bringing them on-board.

Edit: Some extra points:

1. maybe have a chat and see if Perl can be introduced into the curriculum? If not, then make it known six months before the ads go up that applicants need to know Perl. This way you'll get people who have Perl experience and who have actively demonstrated their motivation.
2. can you open up some small projects so that they could be done by potential candidates during this six

months?

3. approach the design of your large-scale projects so that they can be done in a piece meal manner. This is how The ACE Components have been done iirc.
4. allow a specific period for documentation and review of the work done by the departing grad student.
5. allow an overlap period of at least a couple of weeks where the new grad student can work with the departing grad student. They can learn the development environment and they can be guinea pigs for any updates to your wiki.

Still more to come...

HTH

cheers

Share Improve this answer Follow

edited Feb 12, 2009 at 17:34

answered Feb 12, 2009 at 15:52

Rob Wells

37k • 13 • 84 • 147

1

That is a pickle, but it is not as uncommon in the commercial sector as you would think. I heard a statistic once that the average tenure of a programmer industry-wide is about 18-24 months. Normally I would suggest getting more experienced programmers who would

require less ramp-up time and only need to be trained on the problem domain/technology updates and not the basics

I think your best bet is just to ask for about 30-50% more grad students than it will take to actually perform the job to account for the learning and ramp-up time and invest in some additional resources for testing as this environment is a recipe for mistakes since everyone is learning on the job. Also, this is probably difficult given the academic schedule, but try to stagger the start dates as much as possible to maximize the overlap between employees. Pair-programming teams of new-hires/old-hires might also help increase consistency and supplement the training without sacrificing too much productivity.

Share Improve this answer Follow

edited Feb 12, 2009 at 16:09

answered Feb 12, 2009 at 15:57

JohnFx

34.9k ● 18 ● 107 ● 166

1

- How much time should I invest in training them, and how much would simply be a waste of time?
  Answer: It's not the amount of time or the amount of waste, but perhaps the approach. Would it be possible to video train - video yourself training one person and provide it as training for subsequent students/developers. You can add over time, but it

does reduce your time needed to go through the same process.

- How much time should they take simply getting acclamated to our process and the project? **Answer:** This is all dependent on the person...min. of a day max of 2-3 weeks I would guess on avg.
- Are there any specific training practices or techniques that can help with this kind of situation? **Answer:** Video training (home made), having the current student/developer create/update a wiki of needed info, points of interest, etc.
- Has anyone dealt with a similar situation before? **Answer**: Use to be a 12-18 month avg. turnover - I would imagine that it's changed now (longer), but every company has turn-over, but perhaps not forced like yours due to the resource being students.
- Do I worry too much, or not enough? **Answer:** Knowledge lost through transition is a key risk area...

Share Improve this answer Follow

edited Feb 13, 2009 at 17:48

answered Feb 13, 2009 at 17:42

meade

23.3k ● 12 ● 33 ● 36

Is the application something you could consider open-sourcing?

0

Share Improve this answer

Follow

answered Feb 17, 2009 at 13:44

Vlad Gudim

23.5k ● 16 ● 71 ● 92