Apprenticeships: This time it's personal

Jason Gorman and Will Price, SPA2014, Monday 30th June, 2014 1:30pm – 2:30pm

Will currently studies Computer Science at Bristol. He'll play the role of software apprentice for this presentation. Jason will describe a particular experience, which may or may not be typical.

Beginnings

Jason began at age 11 in 1982 with a ZX-81. He and his brother played with Basic. As a hobbyist, he explored various languages on various different machines. In 1989, Jason decided to study Computer Graphics at university. There was no such course, so did a degree in Physics with a large computational element using "grown-up" computers such as DEC and Sun.

In Jason's industrial year, Jason worked for an aunt, doing business system development in C++. For most of the time, he was the only developer working for them. He found that he was a better programmer than physicist. He was feeling quite smug in 1994.

After graduation, a series of fairly sh*tty jobs followed. In 1995 he joined Intergraph as a trainee software engineer and was re-educated – a very humbling experience. In 1997 he went freelance as a result of being made redundant. He headed towards London.

Initially he assumed that all freelance contractors must be brilliant, because they got paid so much. Finding out the dreadful truth, he became ambitious to improve. A DIY apprenticeship began in 1997 and hasn't really ended. He read lots of books, joined lots of groups and tried a lot of stuff.

Towards the end of the 1990s, a couple of books appeared: "After the Gold Rush" by Steve McConnell and "Software Craftsmanship" by Pete McBreen. Jason began to realise that many of the gurus were really saying the same things, but with different words. He determined to extract what was common. For example, the cost of defect correction is generally agreed to be lower the sooner the defects are found.

Jason's own "handle" on software development is summed up in his own book "Back to Basics".

Software Apprenticeship

Around 2001, Jason had a feeling that he knew enough to call himself a software developer – about requirements management, architecture, languages, tools, testing etc. How could someone else be brought up to this level?

The first place to look was academia. The head of a computer science department has to be a brilliant politician. They'll be very keen to entertain you to lunch and bring current industrial practice into the class. Unfortunately you are then handed down to someone less busy with a different agenda and blinkers. On balance, Jason concluded that universities simply cannot teach this stuff.

Jason conducted a survey of computer science students to ask why they were studying computing. Over two thirds of them wanted to become software developers. Smaller fractions wanted to research CS, some wanted to teach it, and a substantial fraction didn't know.

Disappointingly, less than 500 hours of the three-year course typically is practical software development. Jason believes that the proportion should be much higher.

Industry was equally disappointing. About two-thirds of companies offer no training whatsoever. A much smaller proportion offer training / apprenticeship of a few months up to 18 months. Even this is not nearly enough to become a competent software engineer.

Industry and Universities don't see it as their own responsibility to provide vocational training. Industry sees the university qualification as worthless – "graduates have to be trained from scratch". Yet most companies still demand a computer science or equivalent degree.

Lastly, Microsoft and Oracle certifications exist – but what value are they? They only train you to develop software using one specific toolset.

What are their strengths then?

- University: some theory, some strategies for researching solutions
- Employers: experience (school of hard knocks)
- Nobody: practice

The Community: another brick wall. People are generally unwilling to give up their time for free.

Alternative Models

EpiGenesys

Run by University of Sheffield, this is specifically structured to help students learn software development in a genuine production environment. Jason would like to see this type of initiative in every university.

Renishaw

The only example Jason has found in the UK of a commercial company offering long-term software apprenticeships. Per week, apprentices perform 4 days of productive work and 1 day of academic learning. This has been running since 2008. The value is continuity of practice over half a dozen years.

Guided DIY Apprenticeship

Jason offers to mentor young people through the early part of their career. Will Price is one example. There is a written agreement (covering 1 year at a time) requiring Jason to provide 2 hours of one-on-one mentoring every 2 weeks. In return, Will undertakes to attend a number of relevant conferences in the year, provide talks etc.

Apart from one or two face-to-face meetings per year, they use Skype and TeamShare to collaborate. Will keeps an online Apprenticeship Diary. Jointly they are developing a game platform (codemanship-app.herokuapp.com). The code is on github.com – http://github.com/willprice.

Learnings:

- Jason didn't find Will – he approached Jason. Some kind of brokerage is probably needed
- Paired informally for some months
- Had lunch together, drank some cocktails to get to know each other
- Jason discovered (again) that he was not a "master software craftsman"

Demo

Will explained that Jason and he had been working together on a tic-tac-toe rule engine. The tests run successfully. A refactoring session was run with Jason able to see Will's screen.

Questions

Eighth Light is a company offering apprenticeships – have you heard of them? Yes, they are one of those offering one-year apprenticeships.

Shouldn't the apprenticeship begin with some basic information about how computers work? That's what the theory taught by Universities can be for.

Who do you believe when University tells you UML is excellent and Jason says it is useless? Mainly Jason!

What can you do to avoid the master's views being taken as gospel by the apprentice? That's why Will is here today – to expose him to different influences that might take issue with the received wisdom.

How can you demonstrate a return on investment in training? It requires a long-term view. Privately held companies like Zuhlke probably find it easier, as does Renishaw because it does projects over a period of years. For Jason, the question is no longer "what's in it for me?", but "what would have been in it for me 20 years earlier?".