About Allan Kelly
Allan Kelly has held just about every job in IT, these days he provides training and consulting in development management, processes & products, especially around Agile. He specializes in working with software product companies, aligning company strategy with products and processes. More about Allan at http://www.softwarestrategy.co.uk/allankelly

Programmers without TDD will be unemployable by 2022 (a prediction)

by Allan Kelly on January 16th, 2014 | **Filed in:** Software Development

New year is traditionally the time of predictions, and several of the blogs I read have been engaging in predictions (e.g. Ian Sommerville "Software Engineerng looking forward 20 years."). This is not a tradition I usually engage in myself but for once I'd like to make one. (I'll get back to software economics next time, I need to make some conclusions.)

Actually, this is not a new prediction, it is a prediction I've been making verbally for a couple of years but I've never put it on the record so here goes:

> ***By 2022 it will be not be possible to get a professional programming job if you do not practice TDD routinely.***

I started making this prediction a couple of years ago when I said: "In ten years time", sometimes when I've repeated the prediction I've stuck to 10-years, other times I've compensated and said 9-years or 8-years. I might be out slightly – if anything I think it will happen sooner rather than later, 2022 might be conservative.

By TDD I mean Test Driven Development – also called Test First (or Design Driven) Development. This might be Classic/Chicago-TDD, London-School-TDD or Dan North style Behaviour Driven Development. Broadly speaking the same skills and similar tools are involved although there are significant differences, i.e. if you don't have the ability to do TDD you can't do BDD, but there is more to BDD than to TDD.

The characteristics I am concerned with are:

- Developer written automated unit test, e.g. if you write Java code you write unit tests in Java… or Ruby, or some other computer language
- The automated unit tests are executed routinely, at least every day

This probably means refactoring, although as I've heard Jason Gorman point out: interest in refactoring training is far less than that in TDD training.

I'd like to think that TDD as standard – especially London School – also implies more delayed design decisions but I'm not sure this will follow through. In part that is because there is a cadre of "designers" (senior developers, older developers, often with the title "architect") who are happy to talk, and possibly do, "design" but would not denigrate themselves to write code. Until we fix our career model big up front design is here to stay. (Another blog entry I must write one day…)

I'm not making any predictions about the quality of the TDD undertaken. Like programming in general I expect the best will be truly excellent, while the bulk will be at best mediocre.

What I am claiming is:

- It will not be acceptable to question TDD in an interview. It will be so accepted that anyone doesn't know what TDD is, who can't use TDD in an exercise or who claims "I don't do TDD because its a waste of time" or "TDD is unproven" will not get the job. (I already know companies where this is the case, I expect it to be universal by 2022.)
- Programmers will once again be expected to write unit tests for their work. (Before the home computer revolution I believe most professional programmers actually did this. My generation didn't.)
- Unit testing will be overwhelmingly automated. Manual testing is a sin. Manual unit testing doubly so.

And I believe, in general, software will be better (fewer bugs, more maintainable) as a result of these changes, and as a result programmer productivity will be generally higher (even if they write less code they will have fewer bugs to fix.)

Why do I feel confident in making this prediction?

Exactly because of those last points: with any form of TDD in place the number of code bugs is reduced, maintainability is enhanced and productivity is increased. These are benefits both programmers and businesses want.

The timescale I suggest is purely intuition, this might happen before 2022 or it might happen after. I'm one of the worst people to ask because of my work I overwhelmingly see companies that don't do this but would benefit from doing it – and if they listen to the advice they are paying me for they start doing it.

However I believe we are rapidly approaching "the tipping point". Once TDD as standard reaches a certain critical mass it will become the norm, even those companies that don't actively choose to do it will find that their programmers start doing it as simple professionalism.

A more interesting question to ask is: **What does this mean? What are the implications?**

Right now I think the industry is undergoing a major skills overhaul as all the programmers out there who don't know how to do TDD learn how to do it. As TDD is a testable skill it is very easy to tell who has done it/can do it, and who just decided to "sex up" their CV/Resume. (This is unlike Agile in general where it is very difficult to tell who actually understand it and who has just read a book or two.)

In the next few years I think there will be plenty of work for those offering TDD training and coaching – I regularly get enquiries about C++ TDD, less so about other languages but TDD and TDD training is more widespread there. The work won't dry up but it will change from being "Introduction to TDD" to "Improving TDD" and "Advanced TDD" style courses.

A bigger hit is going to be on Universities and other Colleges which claim to teach programming. Almost all the recent graduates I meet have not been taught TDD at all. If TDD has even been mentioned then they are ahead of the game. I do meet a few who have been taught to programme this way but they are few and far between.

Simply: if Colleges don't teach TDD as part of programming courses their graduates aren't going to employable, that will make the colleges less attractive to good students.

Unfortunately I also predict that it won't be until colleges see their students can't get jobs that colleges sit up and take notice.

If you are a potential student looking to study Computer Science/Software Engineering at College I recommend you ignore any college that does not teach programming with TDD. If you are a college looking to produce employable programmers from your IT course I recommend you embrace TDD as fast as possible – it will give you an advantage in recruiting students now, and give your students an advantage finding work.

(If you are a University or College that claims to run an "Agile" module then make sure teach TDD – yes, I'm thinking of one in particular, its kind of embarrassing, Ric.)

And if you are a University which still believes that your Computer Science students don't really need to programme – because they are scientists, logisticians, mathematicians and shouldn't be programming at all then make sure you write this in big red letters on your prospectus.

In business simply doing TDD, especially done well, will over time fix a lot of the day-to-day issues software companies and corporate IT have, the supply side will be improved. However unless companies address the supply side they won't actually see much of this benefit, if anything things will get worse (read my software demand curve analysis or wait for the next posts on software economics.)

Finally, debuggers are going to be less important, good use of TDD removes most of the need for a debugger (thats where the time comes from), which means IDEs will be less important, which means the developers tool market is going to change.

(Postscript: sorry about the formatting problems with the first upload.)

Reference: Programmers without TDD will be unemployable by 2022 (a prediction) from our JCG partner Allan Kelly at the Agile, Lean, Patterns blog.