The Culture of Code

Have you noticed how sometimes throwaway comments can have an impact far beyond anything they intended, or even warranted? They can make you realise you have a blind spot (or deficient mental model), and once realised it's impossible to go back to the old way of thinking – indeed sometimes you wonder how you could ever have thought that. You might call it a paradigm shift, I suppose. I remember the time a female friend of mine said "Why do men always think that the bath mat is for drying their feet on after a shower?" Once I'd got over my irritation at the gender stereotype, I thought to myself "Hang on – isn't that exactly what a bath mat is for?" Clearly not! A change in my practice resulted, and I am now sure to dry my feet with a towel!!

Something similar happened to me more recently, when a fellow developer said something like "the organisation's culture might be really good, but if you're talking about the culture of the code that's a different story". The idea that every organisation has a culture is pretty familiar. It may be a positive culture, with high levels of transparency, trust, encouragement, co-operation, and that sort of thing. Or the opposite; having passive aggression, sabotage, negativity, secrecy, politicking, and so on. The reality is most organisations have a mix of these things. But code? How can lines of text (which get compiled into a set of 0s and 1s) possibly have a culture? Isn't that just unhelpful anthropomorphism?

He went on explain. The code, he said, can be passive aggressive, in that it actively resists any attempt to understand it, or engage with it. It's opaque and confusing, with poor naming, unfathomable flow of control. It sabotages attempts to improve it or refactor it, or even test it in some cases. There may be politics embedded, as different developers have wanted to do things in their own special way, ignoring what has gone before. The code is waiting to catch you out, knock you flat on your face, and laugh at you. There may be snarky or disrespectful comments.

This notion was a revelation to me. But he was absolutely right – source code does have a culture. The way we write code embodies a culture within it. Source code can be helpful and co-operative; easy to understand and maintain, so you can have confidence that changing one bit here isn't going to break something over there. Or it can be obstructive and difficult. A spaghetti tangle, where it's almost impossible to understand the logic, and you can be pretty sure that changing one bit is going to have a completely unexpected side effect in an apparently unrelated part of the system.

Obviously it is nothing new to suggest that code can be "good" or "bad". Code reviews are an excellent way of trying to maximise the "good" code, and minimise the "bad". I recently came across Maslow's Hierarchy of Needs applied to code reviews [Dein]. The idea is that the lower levels of the pyramid (such as correctness) need to be satisfied before the upper levels should even be considered (elegance and inspiration). There is much debate about what the bottom levels should be, i.e. which is the most important element of "good" code.

But I think the notion that code embodies a culture lifts it above being merely “good” or “bad”. Rather, we’re dealing with something that has a material impact on other people. It’s not just the quality of code itself that matters, however we choose to measure that, but also the effect it has on others. It can be easy to work with, or difficult and obstructive, and this is a material contribution to the culture, certainly of the code, and potentially of the organisation itself.

A second offhand comment came along a few months later, when another colleague pointed out that the code I write is not “my” code. That is to say, I don’t “own” it – it is the company’s Intellectual Property, and it will probably have a lifespan beyond my employment there. It may not even be me who is the next person to work with it. As proud as I might be of a piece of code, or however much I see it as my baby, it isn’t actually mine. It follows that if code isn’t “mine”, but “ours”, then I am actually interacting with my colleagues (and indeed our customers) when I am writing code.

To return to the bathroom analogy, it’s a bit like using the bath mat to dry your feet. It might do the job for you, but the next person will find a soggy (and quite probably smelly) sodden lump on the floor. This is writing code with a poor culture – it causes pain and discomfort for my colleagues now, as well as those to come, and they have to spend time and effort tidying up my mess before they can get on with their business. On the other hand, if I write code with a good culture, it’s like leaving the bathroom clean and dry. The next person finds it is a pleasure to go into and even a source of blessing. Same with code - they can pick up what I have written; easily understand what it is doing, and why I wrote it that way, and make whatever modifications they may need to. Perhaps we could even take a leaf out of Uncle Bob’s book, and leave the existing code a bit cleaner than when we found it [BoyScoutRule].

The upshot is, the way we write code as software developers embodies a culture, and is quite possibly one of our principal contributions to the culture of the organisation we are a part of (be that a business, a special interest group, or whatever). It’s very easy to see that the way I behave in person (or on Zoom!) towards my colleagues is influencing the culture, but so too does the way I write code. If I write code with a poor culture, it brings my fellow developers down. It slows down the progress, and increases costs and risk. At the risk of overstating, you could describe it as “cursed” – not hexed, but rather a curse to work with, bringing pain and torment. I have certainly seen my share of code like that! But perhaps our code could be blessed instead? Not consecrated, but a means of blessing, by which I mean bringing pleasure to others; beautiful, well crafted, and elegant code that is a blessing to work with.

Clearly this concept doesn’t stop at the IDE, but extends to the resultant applications. The way we design a User Interface, or handle exceptions also embodies a culture, implicitly capturing what we think of our end users or customers. We might ask whether the software we write is a joy to use – co-operative, encouraging, and trustworthy? Or is it aggressive, annoying, punishing, or difficult to understand? And what about the wider ethical considerations, whether intrinsic (such as an inefficient

algorithm, which wastes power doing unnecessary calculations) or in its purpose or use (AI for seek and destroy on UAVs, say). For further exploration of these sorts of considerations, I commend Michaela Greiler's work on Code Reviews, and in particular her code review checklist [Greiler].

The truth is that all source code has a culture, and as software developers we have the choice over the culture we embody in our code. We can approach our art as a means of blessing our colleagues, customers, and even the world, by creating code with a good culture. But good culture doesn't happen by accident, and if we do not attend to it we run the risk of instead bringing about curse through our code. And whatever else, don't forget to dry your feet with a towel after taking a shower!

References

[Dein] Charles-Axel Dein – "Maslow's pyramid of code review": http://www.dein.fr/2015-02-18-maslows-pyramid-of-code-review.html

[Greiler] Dr. Michaela Greiler – "A Code Review Checklist – Focus on the Important Issues": https://www.michaelagreiler.com/code-review-checklist-2/

[BoyScoutRule] Robert C. Martin (Uncle Bob) – "The Boy Scout Rule": https://www.oreilly.com/library/view/97-things-every/9780596809515/ch08.html

Author Biography

The Rev'd Dr James Handley is a simultaneously a commercial software developer and priest in the Church of England. He has worked in computing all his life, and for the past 15 years or so he has focussed mainly on GIS and mapping software. Of late he has become particularly interested in how software development can influence faith and ministry, and vice versa. He can be contacted via revjameshandley@gmail.com.