

Developer on the Move

By Dr. Neil Roodyn

The summer in the southern hemisphere was approaching and I knew it was time to head back to my favourite haunt in the Asia Pacific, Sydney, Australia. Time to catch up with old friends, spend some time on the beach, and increase the value of some Australian software businesses.

Measurable Success

One of the first stops for me was the Sydney .NET user group (www.ssw.com.au/NetUG). This is run by Adam Cogan one of only two Microsoft Regional Directors in Australia. Adam is the Chief Architect for SSW, a software development company in Sydney. He has been working with software for over 10 years and has an unhealthy fascination with mechanisms to measure the success and improvements of his co-workers and his company. The topic of discussion in this particular user group session was this measurability and KPIs (Key Performance Indicators) for software developers. He covered some 30 different things he measures – from number of test cases written (for developers) and the number of Google mentions (for products) to the number of to the number of emails sent (for sales staff and developers). The session has got me thinking about a host of issues that arise when you try to measure performance and success.

Become part of the business

I guess the first question you might be asking is “why?” Why do we need to measure these things? The blunt answer is that unless you code purely for fun and don’t expect to get paid for what you do then you are involved in some sort of business enterprise. Every responsible business should have some way of knowing if it is doing well or poorly. Of course the bottom line is often profitability and accountant types like to keep an eye on things like cash flow, profit to earnings ratios and other such bean counting figures. These are of course important and without someone doing these jobs then we (software developers) won’t get paid. So how do software developers become more responsible within the business and measure if they are doing well or not? Most of us tend to shirk this responsibility because it is too much like the activities of those bean counting accountant types, so we leave it to the project managers and other people in management roles such as human resources.

Don’t leave it to the project manager because this is where it all falls down, because we know these guys don’t have a clue about software development, right? They don’t even understand bitwise operators so how can they measure our success? So what is the answer? Should we just ignore them and get on with our jobs? Let them write reports on our success and failure over the years? Of course not, in fact many teams I worked with in the last few years are not allowing this to happen and are taking the issue into their own hands providing feedback to the organisation of how well they are doing, areas they can see improving and issues they are currently struggling with.

Stop! You mean they are not just writing code? No. They are part of the business and understand that as part of a business they need to take out a small amount of time, from coding, to see how they can improve the business from their side of the game.

So what can we software developers measure? How can we show that we are improving and that things are going well? In the past developers were measured by the LOC (lines of code) they wrote and hours they worked. These are often counter productive measurements, because in order to succeed you can write huge long functions, duplicate code and work long hours writing more bugs the more tired you become. It is better to measure factors that will improve the quality of your product if the factors are improved. Then the developers will know what to aim for and how to impress their success upon the process. Some ideas might be to measure the number of test failures, the number of reported bugs/issues, and the number of function points/use cases/customer stories completed. Be careful though because this can turn into a game of statistics and it is easy to manipulate the stats to prove you are doing well when in fact you are far from hitting the target.

Measure Success as well as Failure

A company I worked with a few years back had an automated overnight build and test machine. This machine would build all the source code that was in their source control system then run a set of automated tests against the build output. Any build failures or failing tests would then be reported via email to the development team. When I first started working with this company they told me that they knew they were doing well because they overnight build and test hadn't reported any failures for several weeks. I was suspicious, it seemed unlikely. I reviewed their process and quickly discovered that when the build and test succeeded it reported nothing, no email not a peep. So because the team had not had any email they assumed all was good, however this was far from the case, the build machine wasn't running the build and test cycle each night, in fact it hadn't been running for sometime and so the team received no error reports and carried on regardless.

Their problem was they were reporting only errors not success, this highlights the importance of reporting when things are going well. I would say it is better to emphasis the positive than dwell on the negative. It is often said "you get what you focus on", so focus on the good things that are happening. I would certainly recommend some automation of reporting but beware of the "no news is good news" issue. Make sure you make very public the fact that your team is doing well, for software developers this is often hard. We like to focus our minds on solving problems and so we often want to know what the problem is, so we can fix it. I am sure that within your organisation you can find some good things to measure and report on. I am not suggesting that you ignore the problem areas, just find some way that you can turn them around and provide the team with positive things they action on to help remove the problem areas or at least alleviate the issue to some degree.

Start Measuring

Some activities and within every organisation are hard to measure and yet they still add to the value of the group, an example is a mission statement or a set of company values. These are good things to have and help focus people on what they are doing and yet they are often hard to directly measure on a day by day, week by week or month by month basis. Adam Cogan has a list of 'rules' like this on his web site which might help you as a starting point <http://www.ssw.com.au/SSW/Standards/Rules/RulesToBetterTeams.aspx>

Don't stop doing these things, but as a professional you need to have some measures of improvement. Here are some ideas to get you started:

1. Use cases/Function points/Customer stories completed over time.
2. Failing unit tests, run all the tests at least once per day and record the number of tests that fail
3. New unit tests written, use a code auditing tool to report all the new tests in the system each day
4. Exception/issue reports from customers
5. Customer recommendations, how many jobs does your team get through recommendations from existing customers
6. User testimonials, how many positive reports do you get from end users each month, raise the profile of these as they make people feel good too!
7. Releases made, how many times a year does your team release products.
8. Department costs vs. income (as a ratio if you are not allowed to show the real numbers!)
9. Rate of fixing outstanding issues
10. Courses attended/books read or other learning experiences that can be measured.
11. Response time for customer requests
12. Accuracy of estimates, record how long a task is expected to take vs. the actual time it took
13. Overtime worked (this is a bad thing, not a good thing in case you are wondering!)
14. Functionality vs. lines of code; you want more functionality for less lines of code, so measure the lines of code and compare it to the user stories (use cases or function points) completed.

English born, Dr. Neil travels the world working with software companies. He loves Australia, where he spends the summer enjoying the Sydney lifestyle and helping software development teams get more productive. Dr. Neil spends his other summer each year flying between northern Europe and the USA working with software teams and writing about his experiences. Neil brings his business and technical skills to the companies he works with to ensure he has happy customers.

You can find out more from his website <http://www.Roodyn.com> or you can email Dr. Neil at Neil@Roodyn.com