Software Measurement – What's in it For Me?

Introduction

Recently I attended a conference where I heard a lot of software quality and process professionals lamenting the fact that the software measurement programs they were responsible for were doomed to failure. Further discussion led me to believe that the problem wasn't poorly conceived or executed measurement programs, but poorly accepted ones. The biggest obstacle to implementing a successful software measurement program is getting the software professionals, the ones in the trenches designing, coding, documenting and testing the software, to measure their work. To many software developers, not only is there no obvious incentive, there is actually the disincentive of having their output scrutinized and possibly misunderstood. Who wants his boss to know that the guy in the next cubicle wrote more code than he did last week? If we are to make measurement programs work we need to make sure that the software professionals we are asking to measure understand the personal and professional benefits of having a successful software measurement program. We need to sell the program to them in order for them to make it work. This article discusses the benefits that the folks in the trenches can derive from a well executed measurement program and presents tactics that management can use to motivate their developers to make these programs work.

Why the resistance to software measurement programs?

I have been in the software development business long enough to appreciate the value of a well-implemented software measurement program. This appreciation has developed as I have progressed through my career from programmer to analyst to manager of software development teams. In fact, it would be fair to say that early in my career I viewed measurement programs with skepticism and maybe even a little contempt.

Here are some of the reasons that the typical software developer may resist attempts to institute a measurement program:

- If you are measuring my output, then you may try to use this as an input to my performance evaluation (and thus your measurement will in some way impact my salary). It is not unusual, nor is it unexpected, that software developers resist any measurement of the output of the development process. As software developers, we understand that the measurement of what we do requires a more detailed understanding of software and it's development than is often expressed with the traditional measurements applied to our output.
- The act of measuring my output takes time away from what you hired me to do – develop software. If I am expected to spend all my time measuring how long it took me to develop last week's software, where will I find the time to develop this week's software. Software development is a real time occupation – if I stop to measure I will lose valuable time on the task of the moment
- What does input into the measurement program buy me? You ask me to spend my time and put my reputation (translate ego) on the line to help you develop a measurement program when the only results of this program I can see are more work, more frustration and possibly less pay for me.

If you look at the objections it is clear where the areas of resistance are. While most software measurement programs are counting on the cooperation of the software developers, many fail to address the concerns of these same software developers. If you develop software or work with people that do, you know what I mean when I say software developers are often quite intense in how they approach the work they do. They take tremendous pride in the ability to build something from nothing. They prefer to spend their working day in 'The Zone' – giving their full concentration to the current problem, oblivious to conversations, office politics, telephone calls and incoming e-mails. In general, these developers despise what they see as meaningless paperwork because it detracts from the main mission – creating software. If we, as the managers of software projects and the quality assurance programs associated with them, don't look at

software measurement from the developer's perspective, we will never be successful with our measurement programs.

Do software developers benefit from measurement programs?

We all know the benefits of effective measurement to the software project, the software project manager and the organization that uses or sells the software. We often don't bother to think of the benefits these same programs might offer to the software developer. Rather than ignoring these benefits we should be identifying them and using them to sell the measurement programs to the developers. Measurement programs are intended to improve organizational processes in order to make higher quality software in less time with less effort. Certainly the main benefit to the organization can be seen as a benefit to the developer as well – since the result will (hopefully) put more money in everybody's pockets. But there are secondary benefits for the developer as well. With defined processes in place, they stand to find themselves less subject to the side effects of bad software development plans. They are less likely to end up working on a project that is doomed to fail and more likely to end up working on a project where the goals are clearly defined, the software development process is established at the start and carried through to successful completion. Most importantly, they are more likely to find themselves consistently involved in projects where they feel good about the work they are doing.

Selling measurement to the measurers

What steps can you take to ensure that the software designers, developers, testers, etc. in your organization buy into your measurement program? Start by having clear goals for the measurement program and making sure that everyone on the team understands what these goals are and what the organization expects to gain. Then follow up. Make sure that there are periodic reviews of the measurement program to determine whether or not these goals are being met. Make the results of these reviews available in concise, easy to understand terms. Celebrate successes but don't hide failures. If goals are not being met, show how the problems are being addressed and what changes are planned. No one wants to take time away from the important (and fun) part of their job to do paperwork that feeds some sort of a rubber stamp process. Show how the measurement program is helping to meet specific organizational goals. Relate it to the bottom line wherever possible.

Involve everyone in planning the measurement program. Include the measurers in the decisions about what to measure and how to measure it. This benefits your program in two ways. First, the measurers will feel more ownership in the program if they have had real input from the beginning. Second, they will have valuable input into what measurements are easy to make and which ones will be impossible to measure consistently and accurately. You may find major resistance to some of your measurements but this is likely to be information you would want to know up front. It may be necessary to rethink some of the measurement goals or to consider additional automation investments. You may find that you get suggestions for alternate measurements or measurement processes that make good sense for your program but cost substantially less in time to accomplish. These folks know what they know and know how best to get to it. Make sure you include them in the planning process.

Support the measurement program from the top down. Software developers, like everyone else, are evaluated against specific goals. If none of my goals include measurement tasks but I am asked to include measurement as a deliverable – this will negatively impact the amount of time I'm going to put into the measurement tasks. If, whenever there is a schedule crunch, my management tells me to put off the measurement tasks to meet the current schedule goal – the seriousness with which I approach measurement will be diminished. In order to get acceptance from the measurers, they must see commitment from above. Deliverables at every milestone should include specific measurement data. They should be as important to the milestone as the requirements document, design document or executable being delivered. All the players should understand that measurement is part of their job. Everyone on the team should have specific goals that relate to measurement and that are considered along with other goals when performance is evaluated. In order to get the folks in the trenches to take measurement seriously, management needs to take it seriously. It can't be the first thing that gets pushed aside in a crunch.

This is much easier said than done. We often are developing software against aggressive, inflexible schedule constraints. And let's face it, when the business is riding on timely delivery, how important is it

that we stop and report how much time we spent on each task and how many defects we delivered in the design? The answer depends on how long you expect your business to be in business and how many more times you want to be in this schedule crunch! Of course you don't want to sacrifice your business to make your measurement program succeed. On the other hand, you want to make sure that exceptions to the measurement rules are carefully considered and occur only rarely.

Relate successful measurement to better working conditions. Having spent many years in the trenches writing code, I have occasionally found myself in the office late at night trying to finish a module or track down a problem. Although sometimes I had only myself to blame, there were many instances where I found myself wishing my boss had done a better job of planning the project (or done a better job selling his plans to his management). I suspect I'm not the only software developer who has found him or herself rushing to catch up to a schedule that didn't make a whole lot of sense in the first place. Over time I have come to realize that one of the ways I could have helped my boss plan projects better was to supply him with accurate software measurements against which future plans could be based. Not only has this helped to reduce schedule pressure, it has also resulted in my participation in calmer projects with much more predictable outcomes. It's amazing how much easier it is to estimate the future accurately with data from the past. It's equally amazing how much easier it is to negotiate a software plan with management if you have data (or previous estimating successes) to back it up.

Not to say this will be an easy sell up front. You can't show improvements until you've done a fair amount of measurement and you have to be committed to using the mesaurement in this way. If making more realistic schedule estimates is not one of the goals of you measurement program don't present it as one. Some organizations use schedule pressure as a means of increasing productivity. (I don't think this is a particularly effective long-term strategy and would recommend it only for those projects already recognized as death march projects). If you belong to an organization like this than I wouldn't recommend trying to sell reduced schedule pressure as a benefit of the measurement program. If however, reduced schedule pressure due to more realistic estimates is a benefit of your program, make sure you sell this aspect to your measurers – the ones who are traditionally there working late at night as the end of the project draws near. Take care to manage expectations as well since this is sure to be more of a long term than a short term goal and you don't want to lose enthusiasm just as you are about to start making real progress. As you begin to log successes publicize them well.

Make measurement easy. Invest in automation. Automation is one of the biggest keys to a successful measurement program for two reasons. First, proper automation ensures accurate and consistent measures. This is not to suggest that your software professionals are not honest and accurate, but only that any measurement you ask a person to perform is subject to all of the inconsistencies and errors that we humans are prone to make. Second, by showing the software developers that you are willing to invest in lightening their burdens, you are encouraging them to put the proper effort into executing the non-automated parts of the process.

There are so many areas where automation should be considered. Modifications could be made to a time keeping system so that effort is tracked at the level required for your measurement program. An automated defect tracking system could be employed to keep track of testing, defect and rework statistics. Automatic code counters could maintain size counts on your software. Testing tools that extract test time, defect statistics, and cyclomatic complexities could be added to the test suite. Tools could be developed to work with the integrated development environments to determine complexity or size information about the code as it is being designed and developed. The trick to making automation work is to identify what tools you currently use that could be modified to do measurement or that have exportable information that meets one or more of your measurement objectives. A configuration management system could be modified to collect effort information automatically by time stamping check in and check out. If you could automate the collection of this information and enforce rules concerning how these activities occur – you could keep excellent track of the amount of time spent doing code and rework on your software systems. Evaluate the tools you have with respect to the types of data they collect and how available and exportable that data is. Develop interfaces with other products and stand alone utilities to build on existing capabilities. Involve the developers in this evaluation since they will have wonderful ideas about areas where small investments might reap big dividends.

Combat the BIG perceived negative of measurement programs. Don't ever let your measurement program become a basis for performance evaluation of individuals. It is important to establish realistic goals for each individual and it is entirely appropriate to have measures to determine degree of success toward reaching those goals. Take great care to minimize any overlap between these measurements and those that are essential to your measurement program. It is already a stretch to get developers to take the time to provide accurate data, if they feel the information might be used against them it will be impossible.

There are some fairly effective ways to prevent the impression that measurement programs are being used to evaluate individuals. Limit visibility of data that tracks directly to individuals. Certainly make sure that all public presentations of productivity and defect data show statistics at higher levels of abstraction. It may make sense to do this by software component, software systems, or even by development teams – depending on your organization and the goals of your measurement program, but never let it be against individuals. It is important to be sensitive to the fact that you are measuring output and quality but not measuring all of the things that would explain deviations at the individual level. Work hard to prevent any impression that this might happen.

Conclusion

Software teams like to work on well planned, well executed software projects where they feel they are being given the proper amount of time to build quality products and they feel they have some control over events. They will be more than willing to support programs and processes that will take them closer to this goal. As the person responsible for instituting a measurement program, it is an important part of your job to make the measurers understand what they stand to gain from a successful measurement program. You do this by involving them in the process from the ground up, offering incentives for meeting measurement goals, and making sure that you and your management support the program properly. You also need to make measurement as easy as possible and to ensure that there is a clear vision of how successful measurement will lead to better plans and processes. When you start thinking about developing a measurement program don't make the mistake of dismissing the measurers as some necessary evil to be dealt with when the program is implemented. Take the time to understand what their needs and concerns might be and address these early on. Involve them throughout the process and treat them as partners in your quest toward better software.

Arlene Minkiewicz

Arlene Minkiewicz is the chief scientist at PRICE Systems L.L.C. In this role, she leads the cost research activity for the entire suite of cost estimating products that PRICE develops and maintains. She has over 15 years of experience with PRICE, designing and implementing cost models. Prior to her current assignment as chief scientist, Arlene functioned as the lead of the Product Enhancement Team with responsibility for the maintenance and enhancement of all the PRICE products, including the PRICE Estimating Suite, PRICE Enterprise, and ForeSight. Ms. Minkiewicz speaks frequently on software measurement and estimating and has published articles *in Software Development Magazine* and the *British Software Review*.

PRICE Systems, L.L.C.
Arlene.minkiewicz@PRICESystems.com