JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2006**

Vol. 5, No. 1, January - February 2006

The Unnecessary Tension between Process and Programmer

Some of My Best Friends Use an Agile Software Process

Dave Thomas, Bedarra Research Labs

1 THE PROCESS BORG

Today all organizations are required to achieve process compliance, be it ISO 9000, Six Sigma, CMM, Sarbanes-Oxley, Balanced Score Cards etc. Each of these compliance activities seeks to ensure that the company has the appropriate processes in place to ensure that the company can serve the needs of their customers and shareholders. Most mandate some form of continuous improvement measurement, dare I say metrics, so that the organization can monitor its improvement.

Many of these efforts were originally targeted at improving manufacturing, operations or finance, but they are having an increasing impact on the IT and SE. Also, while these efforts were initially applied to larger companies, they have trickled down to the smaller ones who supply these larger firms.

Having a documented process has therefore become a mandatory requirement for software development. It isn't sufficient to say "we are doing Waterfall, Iterative or Agile; go away and let us get our work done". Like it or not, everyone needs to be able to articulate the process they use for software development. It is time for developers to take ownership of the process side of things so that they can stop fighting it. Process is like democracy: if you don't participate, you get the process you deserve. However, there is no need for process tension to obstruct productive development!

2 SOFTWARE CULTURE YES - SOFTWARE PROCESS NO

Ask any software developer and they will tell you if they feel they are working in a good software culture, and most who answer no will say that they would like to be. However, ask most developers how they feel about their company's software process and they will complain bitterly. Yet the best software cultures always have very disciplined processes which developers follow almost to a fault. Indeed the culture is defined by a unique combination of people and process. The open source Apache foundation is an example of a successful software culture that has very disciplined practices. In fact, all of the successful software cultures I know of have disciplined and valued rites and rituals. How can cultures be so respected while the software processes that help drive them are so repugnant to developers?

Let Them Eat Process

When articulated by the CEO or VP of engineering, the process initiative brings tears to your eyes: clearly we all want to improve. However, too often these speeches take the form of political promises sounding like free universal health care and a good life for all without seriously looking at how they are funded or implemented in the organization. My most recent irritation is the attempt by some to move to Agile development without first making the investment in a continuous integration and test environment.

Traditionally software processes have been dictated top down by the Process or Quality department with little input from the analysts or developers. Searching for the usual quick fix, many companies have adopted commercial processes in the hopes that they could provide the solution. Unfortunately this has created a perception gap which says that process is bad and something which gets in the way. This persistent adherence to a top-down approach is surprising when history has shown that quality and change is best introduced bottom up sharing experiences across teams so that the organization learns from experience. Instead of evolving with the teams' learning experience, the process often remains a static pile of out-of-date manuals or a huge website which is seldom referenced. Top-down processes emphasize artifacts, roles and linear production where as bottom-up processes tend to focus on skills, practices and incremental improvement.

Too often the appearance of process is more important than actually having a process which works! Middle management people have become experts at gaming any measurement system, and can often obtain their quality bonus without actually doing anything to improve quality. The measurement system is gamed to give the same good news results as before. Consider for example the hotel room quality control card that says "if you can't give us at least 9 or 10 call me, the manager, at my home number". Clearly this hotel chain doesn't really care about quality! Nor does it care about improvement since reporting real problems is actively discouraged.

Unfortunately many managers and executives often lack modern real world development experience. Few, if any, have developed and deployed a new major application or product; hence they have maintenance processes at best, which are designed to reduce risk through incremental fixes and features. They believe in all best intensions that if there is a documented process that development will improve. There is little said about first-time software where expert education and mentoring are so important to acquiring new practices. The problem is compounded by process implementations by managers and project leaders that do not provide for learning, proper time allocations for activities etc. In many cases the process is simply a set of check boxes and artifacts to be produced in addition to the software asset (with no additional time to do so).

Culture – The Invisible Implicit Process

What one really wants is a learning organization with a culture that is tightly aligned with an almost invisible process. In such a software culture things just get done according to a seemingly implicit process because that is the culture. In such organizations people follow the process because they just do it, not because it is posted on the wall, distributed in a large manual accompanied by draconian gate reviews, piles of project charts etc. We need to move beyond large complex process definitions to implicit practices which are intrinsic to a productive culture.

Agile Development (www.agilealliance.org) is an increasingly successful example of a bottom-up people/team center process which can be assimilated quickly by small teams to improve software predictability and quality. Agile Development quickly builds a common culture of simple, day-to-day practices which developers live and breathe. Unfortunately, the lack of Agile process descriptions and the disdain which Agile developers have for process creates unjustified concerns about the lack of discipline of developers. Agilists need to meet the need by providing the straightforward process documentation and measurement which organizations need.

3 AGILE PROCESS DEVELOPMENT – MAKE IT BETTER EACH SPRINT

Process descriptions don't need to be lengthy, they just need to describe the vocabulary essence of what people do, what is produced, the sequence in which things are done and how the process can be observed by management or customers who are not actually inside the development team. Since there are lots of publications available, there is no need write a massive document; instead, the documentation needs only to provide sufficient information and appropriate references, ideally to online documents, to allow a newcomer to understand how it is that things are done in your software culture.

Process doesn't have to be built in a single day. It is really just a case of Agile writing where pairing, incremental releases, and collaborative reviews to find bugs can easily produce a simple website. Many Agile teams do this already in one way or another in their team wikis, blogs or websites. If you are concerned about the quality lingo of Six Sigma, or the software metric police, take the time to ask them what information they need and why they need it. Just asking them, indicating you may even care if they can be successful at their job, can often lead to a ground swell of cooperation that may have process experts writing and editing your process for you as you speak.

Leverage Open Space Communities of Practice

Virtually everyone accepts that Agile works inside a small team, but large organizations need to have common language across the organization so that people can communicate using the same vocabulary and so that people can be moved to from team to team and new employees can be brought up to speed quickly. Building process consensus across teams is best facilitated by communities of interest who meet informally in open space to discuss how they can improve their practices, be they in developing stories, acceptance tests, unit tests, interfaces, acting as scrum master, measuring progress etc. Grow your process and culture bottom-up, sharing what works and doesn't for your team and your organization.

Measurements – Let Code Report Itself

Agile advocates' providing constant visibility to both customers and developers through collaborating as well as visible progress through burn down and velocity charts. The Agile manifesto values working code over voluminous documentation. Similarly, small releases and continuous integration are essential practices. Since the code is so important, make sure the code base is instrumented and stories' implementations, unit, and acceptance tests documented so that rather than using cumbersome project reporting tools one can just produce accurate information from the code base. "Luke, trust the code". Why fake it with tedious reporting tools if you can collect the information at every build directly from the code base (feeding whatever inane PM tools are still in use auto magically). Project information can best be obtained from simple qualitative web based questionnaires filled out each iteration by the scrum master, with quantitative measurements for the code base.

Towards Open Process Development

To be fair, the process community is very concerned about the gap between process and programmer and they are working to find ways to improve the situation. Many process champions have tried everything they can to publish processes on the web, wikis and other friendlier formats which may appeal to developers. Recently, organizations have even been situating process people with development teams to try to document the processes actually used by the development teams. However, their efforts are seldom supported by development organizations, which see all process as evil overhead.

Recently Randy Miller (http://blogs.msdn.com/randymiller/archive/2005/05/10/416021.aspx) and others at Microsoft (http://msdn.microsoft.com/vstudio/teamsystem/msf/msfagile/default.aspx) have made efforts to incorporate processes integrated with development tools.

In October Eclipse has announced the Beacon Eclipse Process Framework (http://www.eclipse.org/proposals/beacon/proposal). Per Kroll, principle author of the proposal, () promises EPF will provide open source process definitions and tooling to define such processes. In doing so, the process community is seeking to provide a common place repository for process descriptions as well as tools for defining and publishing process descriptions. The existence of open process documentation should reduce the effort and expense needed to create process description and ideally offer the opportunity for communities of interest to describe practices, including typical pitfalls.

4 SUMMARY

It is time for developers to embrace and extend software processes into a healthy description of your organization's software culture. All that is needed is the vocabulary, practices, measurements, lessons learned and references to places to learn more. Agile software has better ways of measuring and predicting software development than classic PMI. Now we need to make the efforts to articulate these new ways to others. We also need to automate our code bases so that the artifacts themselves can provide the information that organizations need to manage and plan their software.

About the author

Dave Thomas is cofounder/chairman of Bedarra Research Labs (www.bedarra.com), www.Online-Learning.com and the Open Augment Consortium (www.openaugment.org) and a founding director of the Agile Alliance (www.agilealliance.com). He is an adjunct research professor at Carleton University, Canada and the University of Queensland, Australia. Dave is the founder and past CEO of Object Technology International (www.oti.com) creator of the Eclipse IDE Platform, IBM VisualAge for Smalltalk, for Java, and MicroEdition for embedded systems. Contact him at dave@bedarra.com or www.davethomas.net.