Position Paper “Deploying Lightweight Processes”

Jens Coldewey
Coldewey Consulting
Curd-Jürgens-Str. 4
D-81739 München
Germany
Tel: +49-700-26533939
Fax: +49-89-74995703
email: jens_coldewey@acm.org
http://www.coldewey.com

Changing the way people do their daily work is one of the hardest tasks in consulting. If you not only want to make single persons change their habits but a complete organization you have to be extremely cautious not to fall in any of the numerous pitfalls. You have to take care to pick up the team where it currently is, you have to take care not to loose anyone in the team while you proceed, you have to take care not to loose the trust of the management, and last but not least the project has to deliver good code on time. Being successful here needs experience as well as sensitivity.

In my personal experience a smooth change works better than dogmatic preaching. Starting from where the team currently is you have to help them to do the change on their own rather than you doing the change on them. I use a special workshop series to deploy lightweight processes at clients.

The Initial Process Workshop

When a client engages you for a new project one of her questions probably is “*How do you plan to start?*” The answer is important for both you and your client. On one hand your clients expect you to have a plan how to start rather than just coming in and making up your mind on their expense. On the other hand you first have to find out where the team currently stands. Most people can only handle a limited amount of change, which contrasts with the gap between most installed heavy-weight processes and a light-weight process. And finally both management and QA department have to be confident that the new way to work does not harm quality or reliability of the projects. As a matter of fact, most organizations are quite unreliable with their heavy-weight processes and deliver rather mediocre software. However, most managers and QA specialists have been trained to blame lack of discipline and control for the failures, rather than too much of it, as a light-weight process proposes. Arguing about these different approaches beforehand usually leads to dogmatic discussions - and a loss of your engagement.

To deal with these issues, *I usually invite the team for a initial process workshop where they start to design their development process themselves.* There are two basic conditions: First we have to deliver after three months and second we have to maintain the software for the next years ourselves. Starting from these conditions I ask the team members to do a brainstorming on one question: “Remembering your past projects, what speeded you up and what slowed you down?” When all these aspects have been gathered, you have detailed goals for your process: Promote everything that speeds you up and avoid everything that slows you down. From here it is not too hard for the team to define an initial process that is based on the team’s experience and on their organization’s culture. It is almost sure that this process will not be Extreme Programming in its most dogmatic interpretation, but your chances that everyone in the team buys into the process is much better when the process is their own baby rather than something preached by an external consultant.

Still you have to sell the process to both management and QA. Because the process grows from the organization's culture, rather than being brought from outside, the resistance usually is not that hard. One trick is to sell this first phase as a prototype. Most managers and QA departments are more relaxed with prototypes than with a final delivery. This gives you some close season to proof that the process works, before the company's methodologists start for the hunt. And the best proof for a process is delivery on time. If you deliver working code before anyone expects you to, most managers are willing to cover you. And at least the result of the workshop is a well-defined process. It may not correspond to the companies process but who else uses a well-defined process to build a prototype?

One thing you have to take care for during this workshop is a good moderation. For instance, if the participants blame the QA for slowing down the project, the moderator has to help them analyze which detail of the QA's work was necessary in the end and which part needs improvement. A goal like "we'll do this project without QA" is not likely to lead to a delivery on time.

Escorting Change

Now that you have an initial process as a start, there are probably several techniques left, that belong to a light-weight process but were not valued by the team during the Initial Process Workshop. For instance, automatic tests are often considered too complex by many teams. So the next problem is *how to introduce these changes*. At first you have to value the results of the workshop. It wouldn't be too wise to end the workshop with an attitude such as "That was fun but now we're going to go the way I tell you." On the other hand the technique may just be new to the team. They cannot decide to use something they have never heard of. Again, you also have to be careful not to put too much change on the team. You have left well-trodden paths and it is important for the group to stick together now until they really know what they are doing.

What I do during the first increment is to *pick the one or two most important techniques and just start to practice them in public*. The rest of the team eventually watches me for example using automatic tests and the standard reaction is: "Hey, wait a minute! What are you doing there?" After a short while they start to try it on their own and to spread it in the team. At the end of the increment they sometimes even don't remember that the technique was introduced freshly.

There is one obvious restriction on the Escorting Change: If you need expensive tools you can't use a guerilla technique like that. To stay with our example, it is easy to download a tool like J-Unit and use it. If it comes to GUI tests you have to invest some ten thousand dollar and several weeks of time into the tools. You can still try to get free evaluation licenses and try to build a prototype before you suggest the investment to your client. However, you are much more likely to fail here than with a free tool or technique you just introduce by using it.

Post-Increment Workshop

After the team has gone through the complete development cycle of the first increment, they have learned a lot about their own process. They know what worked well, what needs improvement and what didn't work at all. So the question you're now faced with is *how does the team maintain their process?* Everyone in the team has her or his personnel experiences and a slightly different opinion about the past increment. Still you want to ensure that the team works on as a single team and not (only) as a collection of individuals. Additionally failed techniques may have failed for various reasons, among them the simple fact that they were not used correctly or consistently enough. New problems may have arisen the team didn't think about before. These problems may need changes in the process to be avoided in the future. And finally once the management and QA have accepted the fact that you follow a defined process, they want you to stick to the process. The least thing they want to see is a beautiful paper with the process in on every-

body’s desk and a bunch a hack-like-hell programmers who don’t care. Probably they have seen this constellation often enough before.

So to maintain the workshop I usually suggest *a new process workshop after each increment*. The workshop is similar to the Initial Process Workshop with the exception that you focus on the past increment rather than on the overall experience of the participants. After some increments you can pass the moderation of the workshop to other team members. Once they have learned that they can use these workshops to control their own work they usually don’t need external support anymore. The process has become part of the project’s deliveries. Soon the team members also discover that they can use these process workshops to solve human conflicts that came up. Over the time the workshops get a similar flavor as supervisions in social professions.

However, some managers tend to argue that you don’t need to sit together every few weeks and meditate over the process, after you have delivered so many times successfully. It is important to fight this stand and have these workshops regularly. After some calm increments you often find that completely new problems come up that may need significant adjustments to the way you work. The team may change or grow, the project goal may (or will) change, and so on.