Might be a useful thing for the students to read?
rcd

Begin forwarded message:

From: "Cutter Consortium" <kletourneau@cutter.com>
To: Robert Duvall <rcd@cs.duke.edu>
Subject: A Three-Tier Model for Guiding Your Agile Implementation

The beauty of Agile software methods is that they enable us to focus with a singularity of purpose on the iteration management and project management aspects of the software delivery process. Numerous other aspects of software delivery, such as those illustrated in Figure 1, are, of course, of critical importance. Yet, it is the sustained and continuous focus on how we perform iteration management and project management that leads to eventual success with Agile.

Figure 1 -- An holistic view of software delivery.
(Source: Gat, Israel et al. "Reformulating the Product Delivery Process."
The potential pitfall of too strong a focus on iteration management and project management is that we might lose the holistic view of how the "sausage" is made. In particular, we may find that various aspects of Agile are hard to fit in a specific context. For example, a certain Agile tenet such as "test early and often" might not quite fit well in the financial context. It might be as prosaic as lacking the financial resources to invest in an infrastructure that can satisfy test early and often on the required scale. Prosaic as it is, such a deficit could easily undo an Agile implementation.

To address the risk of losing the holistic view, I started using a simple three-tier framework in my engagements, as illustrated in Figure 2.

![Figure 2 -- The method-process-realization framework.](image)

In this figure:

- **Method** is the abstract definition of how one chooses to carry out software development. For example, one might choose waterfall, Scrum, Crystal, or Kanban. Ultimately it is about the fundamental tenets (as distinct from the nuts and bolts) of the software method.

- **Process** is the way the method is specifically implemented in the client context. For example, a client might decide to use two-week sprints and conduct code reviews, but not to implement Behavior-Driven Development (BDD).

- **Realization** is the way tools and resources are applied to the code. For example, a company might not be able to build often enough.

The effectiveness of this model is in shedding light on incoherence(s) between the tenets of Agile (at the method level), the way these tenets are implemented in a specific client context (at the process level), and the hard facts with respect to the actual tools and resources available for the Agile implementation (at the realization level).

It is my repeated experience that many/most of the pitfalls of Agile have their roots in incoherence(s) across these three levels. Here are a few examples from actual Agile engagements in which I applied the model:

**Company A**, implementing the Scrum method, used five-week sprints. Under this process, stakeholders and executives were not able to provide timely feedback through the sprint demos. Hence, company A instituted various supplementary feedback and oversight ceremonies on top of the
Scrum process. In the course of doing so, the decision-making algorithm became incoherent and the “self-organizing” teams of Scrum lost much of the empowerment one would expect to attain through Agile methods.

Consider the company A experience in terms of the three-tier model discussed above. At the model level Scrum does not specify the length of the sprint. For a variety of reasons, at the process level company A implemented sprints that were too long for both its business needs and its operational needs. Rather than adding ceremonies on top of Scrum, the company should have stayed within the boundaries of the method, reducing the sprint length at the process level to (probably) a couple of weeks. In planning the Agile rollout, the length of the sprint should have been determined as an explicit, indeed crucial, part of the (method -> process) mapping.

Company B, as part of its Agile rollout planning, made explicit decisions to: (1) adopt the Scrum method; and, (2) use two-week Scrum sprints. The teams were actually “Scrumming” pretty well, but they had a hard time responding to the needs of their very vibrant community of users. The conflict between the urgent needs of the community versus the two-week sprint length led (unfortunately) to numerous “religious” wars between biz and dev about the software method. Once the folks in this company applied the method-process-realization framework, they realized the heart of the issue was inadequate realization. Only a fraction of unit testing was automated, the continuous integration infrastructure left much to be desired, and they would not have been able to do continuous deployment before they developed adequate tools and appropriate disciplines for selective deployment (e.g., deploy to 0.1% of community members or to community members in just one region) and quick rollback mechanisms in the event of unsuccessful deployment. In analyzing the failure to satisfy the needs of their community from a narrow process perspective, the folks in this company missed the crucial process -> realization mapping.

Company C was saddled with a huge code base of ~20M lines of code. Very unfortunately, it was pretty bad spaghetti code that required regressing the whole code base prior to shipment. The long period of regression testing pretty much rendered their “done” definition useless: most of the testing, and the concomitant fixing, were done at the tail of the development process. As hard as the teams tried, the test early and often tenet of Agile could not be fulfilled.

The main point to keep in mind with respect to applying the method-process-realization framework in your company is that it shifts the focus from a theoretical debate of the fine points of one software method or another to the way the software method is actually implemented and realized in your context. By so doing, it generates actionable insights into how to resolve the real constraints your company wrestles with.

I welcome your comments about this Advisor and encourage you to send your insights to comments@cutter.com.

-- Israel Gat, Fellow and Director, Agile Product & Project Management Practice