TREXIN INSIGHT PAPER

July 2016

DEALING WITH ORGANIZATIONAL DEBT WHEN IMPLEMENTING AGILE

Understanding the culture of the organization is paramount to successfully standing up an Agile team

I have been implementing Agile, specifically Scrum, for over 12 years. I've worked with small 4-person shops to very large teams where the customer confused stakeholders with team members, so they insisted on assigning 74 people on a team. To quickly establish a well-formed productive team, I have found that one of the key pillars of success is to address Organizational Debt early on.

We define Organizational Debt as *an organization structure that does not align with the principles of an Agile mindset*. These being interactions over process and tools, working software over documentation, collaboration over contracts, and responding to change over following a plan.

There may have been good and decent reasons that an organization was originally structured the way it was. It may have been created for command and control or to enable process maturity or, most likely, to solve for resource management for skills and process alignment.

Those reasons become hurdles to a well-formed Agile team because the team will tend to focus on the extensive documentation and process flow gates required of their functional area expertise instead of looking for the minimum viable solution to delivering a product release.

Having delivered many successful Agile projects, I have found that one of the many keys to success is to deal with Organizational Debt up front and start to evolve the organization quickly to a new frame of reference. Let's take a look at the Agile Manifesto and think about how a classical org structure is set up to help us untangle Organizational Debt.

The classic waterfall SDLC (Software Development Life Cycle) focuses on establishing repeatable activity steps resulting in approval gates at the end of each step. The idea is that no development work is done on a product until all of these gates have been completed. The vision of the end result being a nice and tight project plan and work breakdown schedule that everybody follows. This vision has led many organizations to structure their workforce around this concept. The advent of the PMO, EPMO, QA,

Business Requirements groups, and a Development team can all trace their lineage back to this basic concept. People working in functional teams and tossing their "deliverables" down the pipeline to the next group is the clearest example of Organizational Debt we see on a daily basis. Quickly responding to change is not an inherent value in this model as change needs to be continually renegotiated back up the organization line. Instead of creating a large interdependent plan, an Agile team concentrates on the highest priority items and makes commitments in 2-3 week sprints to deliver feature enhancements. This way, the team isn't doing work against things that are likely to change

Take a look at your process steps and ask yourself if the end product you are working to build is an artifact that supports the building of the product or to document what is being built to support a discussion with senior leadership, which is a different problem to solve. The best Agile teams have the ability to merge the process steps and perform them concurrently. People are defining test cases, writing user manuals and release notes in unison with building requirements and developing code. Small, bite-sized pieces of functionality are released in a steady stream of improvements instead of waiting for annual releases. The idea is to be able to produce and review working software at the end of each sprint. This is a cornerstone concept to force the team to think in terms of product delivery as the ultimate goal of a Sprint. If the work you are doing does not materially contribute to releasing the code, then you are focusing on a non-value add artifact.

Organizational Debt in the form of requiring overly rich requirements documentation tend to support the concept of not having to do "re-work" and a contract negation which doesn't account for the fact that things change. Maybe in a week, maybe in a year, maybe in 5 years. Software isn't static. New features are requested to address new business needs, so we need a process that welcomes this change. An organization and process structure that rewards commitments for change gates and long term plans are missing the point that all things change and never does a long range software development plan make it to the end without many modifications. Why spend the upfront time on a detailed plan when we know it will change. This is not to say that having a general idea of dependencies and direction aren't warranted. It's just to say that the likelihood of new priorities outweigh the cost of building a concrete plan.

Coupled hand-in-hand is the concept that while tools and processes are often enablers of being able to seamlessly repeat good practices, it doesn't replace eye contact and face-to-face communications for making and keeping commitments. The goal is to build a product, not update a tracking system. Documentation can be as light as a picture of a white board and a tool as easy as sticky notes on a wall. These can be hard concepts to embrace when moving from the idea of an 18-month roadmap to a prioritized backlog approach. There is certainly a need to define long- and mid-term visions for a product, but these are not part of the Scrum rituals. Those are handled outside of the framework by the Product Owner using tools of their choice. Only when

these backlog items become prioritized do they become visible to the Scrum team. When embracing Scrum, the organization needs to take a hard look at its tooling and process steps and ensure that they can support the idea of "less is more."

Lastly, our Product Owner needs to be a part of the team. (S)He is not an external entity who only accepts the deliverable. (S)He is also responsible for defining "done" at the beginning of the Sprint. (S)He is aware of all of the decisions and tradeoffs during the Sprint and is continuously kept abreast of the progress. Our organizational structure needs to embrace the reality that it's a full time job.

OVERCOMING ORGANIZATIONAL DEBT

Here are the recommendations:

1. Break your program management needs out from your project management needs. What you need to do for managing a program or a product may not fit nicely into what you need to do from a Scrum ritual perspective. Solve for each separately.
2. Agree on the Dashboard requirements for each of these areas up front before you start sprinting so that people know how they will be measured.
3. Assign people full-time to a Scrum Team. In the beginning, you need the heart and soul of the people to be empowered to take advantage of the framework.
4. Drop the pretense of functional areas until you get a feel for any organization impediments.
5. Force compliance with the reporting and dashboard tools. You can't manage it, if you can't measure it.
6. Train, train, train. Make sure that the Product Owner knows their role and responsibilities. Hire a coach that is in tune with your organizational goals, and not just a framework guru.
7. Start your effort by assigning 2 different people to lead the change. While one person concentrates on the Agile Team rituals and cadence, the other is looking at the program roadmap and traditional PM responsibilities. Together they will be able to bridge the PMO gap more effectively.
8. Start small. You may want to concentrate on incremental value such as retiring aspects of technical debt and get into a cadence with a development team before opening up the backlog to the business teams.
9. Beware that even good Agile teams will lapse into old behaviors in time of stress. Make sure to coach actively and empower the team to make decisions for themselves.

This TIP was written by Rick Hailey, a Senior Principal serving in Financial Services and Healthcare & Life Sciences. Rick welcomes comments and discussion on this topic and can be reached at rick.hailey@trexin.com