Organizational Culture Aspects of an Agile Transformation

Shlomi Rosenberg(✉)

Cisco Systems, Service Provider Video Software and Solutions, Jerusalem, Israel
shrosenb@cisco.com

Abstract. For an organization wishing to be more agile after working waterfall for years, it is not enough to just start learning and implementing new ways of working. There must be a parallel activity, at least equally important, of dealing with the organizational culture changes required to support this transformation. In Cisco I deal with those on a daily basis. An organizational culture is much harder to change than work methods. It involves feelings, perceptions and fears, so it is advisable to be aware of the importance and invest in dealing with it. This experience report details examples of these culture aspects, how we deal with them and some tips that can help make such transformation successful.

Keywords: Organizational culture · Agile transformation · Management · Leadership

1 Introduction

I am a senior development manager and part of the leadership of an organization with a staff of over 200 people, which is part of a much larger multinational engineering organization. This paper shares insights from my experience in being an active part of an amazing agile transformation that we have gone through and are still making progress with. I will focus on the organizational culture aspects of an agile transformation of this scale, which in the beginning I wasn't aware of as relevant. I came to realize their importance to success in the transformation and that it is at least as important as the agile practices themselves.

2 Background

For the past twelve years I have been an engineering manager at various levels, managing teams and activities related to development, integration and testing of complex systems in the digital TV industry. The entire development process was completely waterfall based. Our organization has always been structured in a way that is tightly coupled to clear functions. These domains can be based on job roles such as integrators, developers, QC engineers, etc.

They can also be based on components, projects, customers, etc. and all of these were divided between multiple sites worldwide. This type of organizational structure

DOI: 10.1007/978-3-319-18612-2_28

nourished an organizational culture that exhibited some underlying problematic characteristics such as ownership, territorialism, lack of trust, over-management and so forth. Actually, it was sometimes hard to believe that these domains were part of the same company or even that they were all working on the same project, a situation that is clearly harmful to efficiency and productivity. These aspects of organizational culture and various behaviors deriving from them not only make it difficult to improve effectiveness, but also need to be dealt with, specifically when considering such a large-scale transformation to Agile.

3 Ownership

Ownership in general doesn't indicate something negative. After all, when engineers feel ownership, they also feel responsible and it is clear who to go to with questions, support, and bug fixes required. These are things managers actually like and need. The question is what to build the ownership around?

3.1 The Culture Issue

The problem in our organization starts from the fact we develop complex end-to-end systems, which are built from numerous stacks running on various machines at different physical locations and each built from tens of components. Our organization was heavily based around component ownership i.e. Managers and their teams owned code.

When I refer to code ownership I refer to a culture of "no one but the component team is allowed to modify our code". It also nourished a management culture of focusing around what they own "physically", which allowed them to get some materialistic measure of their "power".

This direct correlation between code and teams was drastic and it caused strong dependencies and obstacles in our ability to move fast. For example, different time zones of component owners caused delays in progressing integrations. Some component owners would not even expose their code, so integrators could not debug by themselves. These dependencies were a big contribution for our heaviness as a development organization.

When we decided to transform to become more agile, one of our key principles was to enable our teams to progress feature development in as self-sustained way as possible. We decided that instead of owning code, the teams should own features. For supporting them in doing so, it meant we must reduce the code/component ownership to a minimum so when developing an E2E feature they'll need to develop and integrate all involved components in the feature's vertical flow.

This code ownership was, and still is in few cases, one of the biggest obstacles to our full Agile transformation. Component teams, owners and managers in particular, were nervous about this change. This code ownership is what defined their organization in many ways for so many years. Code is something tangible and is something you can "fight" for. Also, for many engineers, it was comfortable and "encouraged" them to be narrowly focused. Rather than building fast velocity feature development capabilities, they got used to focus on their narrow component domains. They could raise dependencies and have someone to “blame” for not progressing.

Also for those not owning components, specifically the integration teams, it nurtured frustration of not being able to progress effectively as they were constantly dependent on others. For others it was a comfortable situation, as their job was just to perform builds and report back if what they got is sufficient based various level investigations. This integration dynamic was comfortable for the owners and some of the integration teams.

For most people, something comfortable shouldn’t be changed, and they will fight for it. They focus on rationalizing this way of work by pointing the problems to other places and by focusing on the importance of people being experts in their function or component. They would also convince each regarding his role that “it’s too complex for anyone to do it or no one has the knowledge or experience to do what they do”. All valid points, but all can also change.

3.2 How We Dealt with It

The described situation impacts effectiveness as it virtually causes formation of teams within teams, limiting their agility. If not dealt with, it cripples the enablement and self-sustainability of the functional scrum teams turning them practically in to a semi-waterfall integration teams.

As a start, from team structuring perspective, as mentioned above, we defined vertical teams. These teams’ mission is to deliver end-to-end user stories and features. That is what they are measured by. We tried to limit as much as possible the number of horizontal (component centric) teams. The reason it’s limiting the number and not eliminating is because after all there are areas where it makes a lot of sense to remain horizontal (such as point products within the solution which are used by other solutions also, third party component porting, etc.).

To compliment that, decision wise, we made a decision and communicated to all teams, that components can be branched by vertical teams if they need to do so for progressing. This mainly disconnected the “only owners can modify code” from the component ownership. At the beginning we experimented with a vertical team to prove this increases velocity. We were proved right and use the concept more widely in other teams. Also, we are pressuring vertical teams not to be intimidated by new unfamiliar code. We are communicating clearly that we are not happy with them opening impediments for each bug they find in Horizontal components, but rather push to fix them.

3.3 Culture Tip

At early stages and continuously after, identify engineers in vertical teams who are both technically strong and willing to enthusiastically try the change. These engineers together will prove it is doable. This is quicker than depending on external parties. Success stories with results will be your best proof and motivator for others.

4 Territorialism

It’s the most trivial thing to have different job roles in a development organization. We have architects, developers, integrators and QC engineers. Our organization was built according to these roles i.e. each of the above roles was grouped also organizational-wise. So we had an architecture department, a development department, an integration department and a QC department. The flow of development was that architects defined a work-package, the development teams involved developed their part of it, the integration team integrated the components and then the QC teams tested it. Sounds reasonable, but it developed an underlying engineering culture that doesn’t go with agile development.

4.1 The Culture Issue

The underlying culture around our previous structure came mainly from prestige related reasons. It’s simple. Some of these roles subjectively have better reputation than others. Few examples:

- Developers want to develop code, they don’t want to do integration nor QC work.
- Integrators feel it’s beneath them to test.
- Developers don’t want integrators to find fixes for their bugs.
- Integrators feel threatened by QC engineers investigating the bugs they find.

For these reasons and more, people guarded their domains carefully. The engineers were territorial about their engineering function and would usually stay away from other functions.

One of our major goals of the transformation was building vertically enabled teams, who deliver end-to-end tested features. This was not a trivial change for many of our engineers. Suddenly an engineer is responsible to deliver an end to end feature. Together with his team, he is required to do much more than just write code, just integrate it or just test it. He needs to be involved and aware of all aspects of delivering the end-to-end user story. That is part of the new Definition of Done for the team’s user stories. This is a major change to engineering mindset.

4.2 How We Dealt with It

This proved to be a difficult change. It’s not easy for someone who always used to writing code and passing it to someone for integrating and then testing, to start delivering end-to-end working features.

First, we invest a lot in training to understand this concept and its importance as a key to our success. We are also pushing POs to minimize defining user stories, which correlate to these different functions as there was a tendency to define user stories such as “Write tests for functionality X” coming from pressure from within some of the teams. We constantly monitor to see we don’t have single task type engineers in the teams, even on the expense of in the short term sometimes slowing down some activities. Naturally, some engineers are stronger or more experienced in some functions than the others, which is a positive thing.

4.3 Culture Tip

Be persistent on this and work closely with the engineers to explain and convince them that the new way of work is better. Also acknowledge and praise high quality rather that high pace.

5 Managers vs. Leadership

Our command and control driven waterfall organization structure had a heavy management structure of roughly one manager to six employees.

The team responsibilities at the majority of cases were narrow related to the product the organization delivered. Also these team managers were responsible to all aspects of their team. This caused a lot of silos, which was slowing us down and limiting the variety of influence sources to the wide organization product perspective. The culture was also that only managers can lead or influence.

When we began our transformation we didn't focus on organization restructuring, we focused on activity restructuring. We didn't even discuss the organization structuring until we were confident with our program structure.

This proved to be a smart approach as it instantly eliminated all silos, which contributed greatly to the senior leadership to cooperate amazingly. This was crucial for leading the transformation. The leadership team literally didn't have anything to fight about in terms of private interests as with all the scrum teams' buildup it so happened that each manager had his reporting engineers spread between different teams, so teams were not associated with managers. Later, when we were confident with our program structure, the management organization restructuring was much clearer in terms of needs. We could restructure to fit our activities rather than organizing activities to fit a reporting structure. We divided the classical management role into three main categories, namely People, Activity & Technical:

– The people manager role is now focused on taking care of the Engineers and staffing the program teams rather than "interfering" with the teams work - **this is the reporting structure**.
– The activity management is done by the scrum teams and their PO (not a reporting structure).
– The technical management is a tech lead type network that continuously mentor, advise and lead technically the teams (not a reporting structure).

5.1 The Culture Issue

For an organization with such an embedded command & control mindset and culture, the strongest source of influence, power and leadership came from the reporting structure. The common perception was that leadership equals management.

The downside that started building up slowly was that our people managers, which until transitioning to working agile had clear responsibility, didn't find their place in the system. The activities and team management was provided by the scrum teams themselves and the program management, so “all” that was left for them is the people management, which was unclear. The perception was that being a people manager is just a bureaucracy role. It started causing tension and insecurity, specifically for the first line managers. On the other hand, the engineers also gradually showed signs of confusion not understanding who to go to for what kind of issues. This situation alerted us, the leadership team, that we needed to move forward with the reorganization in order to fill all these gaps.

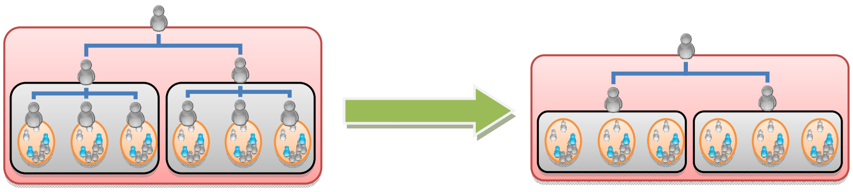

Fig. 1. Alignment of organization structure to support agile

When starting to communicate the structure change shown in Fig. 1, managers, (mainly first line managers) got nervous. At first look and considering the management culture we were coming from, it looks like are eliminating a level of seniority. If they are not chosen as people managers they perceived it to mean they are demoted and they are not part of our leadership anymore. It could be perceived as limiting for carrier development. Although I am confident in that structure, it’s not that easy to make people understand that. Many people who were managers in the old organization structure are no longer managers. It doesn’t mean they can’t be leaders.

5.2 How We Dealt with It

There are actually many more influential positions available at this structure once you understand people manager is not the only way to develop your carrier. It is also not necessarily always the most visible leadership position in terms of the program day-to-day performance.

Saying all that, it’s not that trivial to explain, specifically to a mature organization with such different organizational culture. One of the main actions we took is to consider scrum masters, product owners and architects (we try to assign an architect for each team) as part of the organization leadership. They take an active part in shaping our way forward. As senior leaders, we respect and keep the well-defined boundaries between people management issues, which will be dealt with people managers, and scrum team management, which is dealt with the scrum masters or product owners.

There is a continuous learning and improvement process for shaping the different leadership roles, empowering the different roles to lead their domain.

5.3 Culture Tip

From early stage start communicating the difference between functional management and people management. Also focus on the importance of the different roles in an agile organization. These things take time to absorb and it will drastically reduce frustrations when you apply your reorganization.

6 The Buzzword Trap

My last topic involves a communication related aspect to our agile transformation. Anything new usually involves a lot of buzzwords. Agile is no different and as the name implies, they do indeed create a buzz. In a large scale transformation such as we went through, you must create a buzz, or a sense of excitement, in order to pull everybody on board. However, if you don't walk the walk at the same pace you talk the talk, it is easy to achieve the opposite effect. What my experience showed me is that you can't answer questions or give guidance with buzzwords if you want them to transform the way people work. You must focus on the methodology and only afterwards relate it to its name if you want. Another issue related to this is that I observed people actually turning agile to be their goal rather than the means to achieve their goal.

6.1 The Culture Issue

As agile methodologies are so different in so many ways from waterfall, mentioning words such as sprint, scrum, agile and retrospective as the solution for all the problems we face is far from being enough to motivate a large organization to change. For someone taken out of their comfort zone by such change, specifically many engineers, buzzwords are not comforting. In fact, overloading people with new terminology can be quite intimidating and cause the opposite effect of contempt.

At the start of our transformation, anyone who wanted to sound like he is fully agile "compatible" used a lot of agile buzzwords. However, it became clear that different people have different interpretations of the different words. This can easily cause damage to what you want to achieve as it causes misalignment. For example, you show up for a demo and find that the team was working on it for three days rather than showing their current raw state. Another example is teams being managed by their scrum master by him asking for status at each standup, commenting on it and giving directions for next day. The problem with these behaviors is that it is easy to derail from what we try to achieve as these are command and control behaviors masked by agile structure and neglecting those can easily regress the work culture back to what you originally want to change.

6.2 How We Dealt with It

This requires a lot of self-discipline. We started being aware, mainly when guiding the teams not to use these words. E.g., many times we actually ban the word agile from our discussions. We focus on the methodology and what we want to achieve.

The focus is on understanding the problems we want to solve and discussing the way we believe it can be solved. We don't focus on coupling our solutions with an agile framework. We created a culture, which is open for improvement and feedback. We invest in agile coaching. We continually conduct classes and to complement these, we have coaches joining the teams, even the more experienced teams and providing feedback on what to do more of, what to do less of and what to change. We focus on continuous improvement rather than on achieving some ultimate goal.

6.3 Culture Tip

When going agile and passing the messages needed for this transformation, don't focus on agile related buzz words it is not sufficient to achieve a real transformation. Focus on the characteristics and qualities that come with it and why you need them.

7 What did I Learn from This Experience

My agile transformation experience has shown me the importance of the culture aspects involved in such a deep change to the way we work. It also proves to me and makes me confident that being aware to it and addressing it, is an absolute key factor to a sustainable culture change. I think the following should be an integral part of any leadership team's agenda when transforming to agile.

- Your organization's culture aspects are equally important to succeeding with your transformation. They are more difficult to deal with.
- Be persistent on explaining the advantages of multidiscipline engineering within the teams and value high quality over fast pace.
- Communicate the difference between functional management and people management. Focus on the leadership aspects of different roles
- Focus on the characteristics and qualities that you believe are important for your culture, not on agile buzzwords.
- Choose the right leadership to lead your transformation. Make sure they trust each other, work well together, and understand that organization success is their success.
- Your leadership at all levels must encourage change and support change.

Acknowledgements. I'd like to thank Cisco for the opportunity to be part in leading such an agile transformation. I would also like to thank my colleagues at the leadership team running this transformation.

Lastly, I would really like to thank my experience report shepherd Ken Power for encouraging and really helping bringing this paper together. *Thanks Ken for all the help. It probably wouldn't come together without you!*