Dynamic Cultural and Technical Transformation - Challenges for Engineering Managers

YUKARI HASEBE, Yahoo!
MAI LE, Uber

At Yahoo!, engineering managers are facing challenges everyday. While a team is working on product delivery, managers must encourage and guide the team for their career growth as well. Yahoo! has been experiencing organizational changes at least once a quarter for the last few years. Our portfolio is large and contains many products of different types. This report describes our experiences and how we overcame challenges and adapted to this frequently changing environment without slowing down. In order to be successful, cultural and technical transformation must happen together hand in hand. We must build both a culture and organizational structure where changes can be accommodated as part of doing business as usual.

1. INTRODUCTION

Our objective is to share common management and process issues in large-scale enterprises and how we at Yahoo! have overcome those issues. We’ve encountered challenges in multiple dimensions such as organizational, technical, financial, personal, process and motivational. Each challenge has different symptoms but we were able to identify patterns through different projects and come up with an adaptive model which gives us flexibilities in both process and team structure. This model works most of the time for us. We are calling this model our “Enterprise POD”. Agile POD usually refers to self-contained cross-functional team, Enterprise POD refers to multiple PODs forming a larger scale enterprise level POD. We’ve designed this model to empower the team and improve engagement, leveraging strengths from an existing model. This model has helped our teams and has significantly reduced management challenges.

2. BACKGROUND

Our company has traditional large-scale layers and structure that is not going to go away, at least as far as we can see. From the top there is the CEO, EVP, SVP, VP, Sr. Director, Director, Sr. Manager, Manager then independent contributors at different levels. Agile encourages flat organizations for fast turnaround, but the fact is that just not going to happen anytime soon in our organization. We have no control over that level of organizational changes. So our strategy always has been focus on what we can control then what we can influence next.

In our organization, engineering managers were used to the model of owning one product and managing engineers for that product. Frequent organizational and portfolio changes force teams to re-organize. This model of single product ownership is not flexible enough to move fast. When we joined the Yahoo Search team a few years ago, teams were aligned by Product line. One engineering team was responsible for one product. When a new engineer joined the team, this person only learned one domain area in-depth. Since we joined the search team there have been multiple re-organizations every quarter of different sizes and scope. Our delivery team also got restructured. Team members had to go through an onboarding process repeatedly. It was a big burden for both engineers and managers. We realized our old model was not effective and that we needed to
do something about it. Our team needed to welcome and adapt to changes quickly anytime. So that is why we decided to start our transformation and came up with Enterprise POD model.

3. TRADITIONAL WORKING MODEL AND BOTTLENECK FOR SCALABILITY

Before our transition, our working model looked like the diagram in below. The reporting structure and product execution were tightly coupled. Engineering managers were responsible for product delivery of their assigned product and also managing resources. Engineers were only working on the product owned by their own manager. This model was limiting engineers’ potential to make things better, bigger and be innovative in other areas. There was no reason for engineers to take work outside of their territory and there were benefits to be territorial within one’s own team since that was the only place where engineers can show their work. As a result, engineers had built in-depth domain expertise in their own area, which is both good and bad. The good part is there are “expert” engineers that managers can depend on; the bad part is an “expert” can also become a “single point of failure”. When the expert goes on vacation, a team gets stuck. In-depth domain expertise was praised and so encouraged a culture where only certain people were capable of doing certain types of work, and so information did not need to be transparent since no one else would do that work. This was considered “job security”. This behavior became a bottleneck to scale and adapt. It was impossible to work under this model with a fast delivery cycle. There was no CD pipeline back then therefore there was no proper checkpoint to test quality thoroughly. New code went back and forth multiple times between Dev and QA which was adding more risks of not getting enough test coverage and also increasing the possibility of causing other issues somewhere else. Often times requirements evolved during development and caused more delay. Not having transparency made managers’ jobs even harder to properly review their team’s work to provide proper feedback and performance review. Work done and efforts taken by each engineer were not clearly communicated and validating their work took a lot of effort. Yahoo has quarterly performance reviews and a manager usually has 15+ direct reports. Managers spent significant amount of time doing performance reviews.

![Old Engineering Team Structure](image)

Product managers were not involved too much in day-to-day operations. Requirements given to a delivery team were high level without enough details, so engineering managers gave directions to the team based on their own assumptions. At the end of our development cycle, the Product manager came back and told what was delivered was not what Product wanted so it took multiple cycles to get things “right”. Engineering managers were trying to fill this product gap and also direct the delivery team. A manager driven process will not promote team self-organization since conflict of interests exist. Team members did not get equal say as their manager as to how the work should be done. At the end of the day, since the manager does their performance review, an engineer would not go against their manager. This was a lose-lose situation.
Additionally, the product and business priority was not clear to the delivery team and kept changing. Because of managers being in the middle, the engineers were not kept in the loop about these conversations and cross team coordination was very painful. The product team and engineering team were not communicating enough face-to-face, just sending emails to each other. Consequently, delivery was taking longer than it should. Culture and dynamics at the time we joined the search team was do what was told and don’t take more work, it may come back and bite you later. Also, you don’t need to understand details upfront since it will change later anyway. Frequent re-organizations forced engineers to keep building relationships with a new manager and that was also adding another overhead. And engineering managers were not given enough time to groom engineers, as people moved around too frequently.

4. WHAT WE DID - NEW ENTERPRISE POD MODEL

We knew there were issues in multiple places. So we listed issues and tried to figure out what we could do. We went over each item and reviewed which items were under our control and which ones were not, then prioritized items under our control first. We have no control over organizational layers but we do have full control within our engineering team. And we can also influence our partners such as the Product team and the service engineering team. We decided to change our engineering team structure first. We knew this was a major core change but we wanted to establish a structure that could accommodate changes easily with minimum overhead. If we could do that, we would also be able to solve many challenges we have been experiencing day by day. We came up with a new model and tried it out, it is called “Enterprise POD”. This model allows flexibility in execution while maintaining an enterprise organizational view. Our enterprise POD model was built based on existing organizational functions. Importantly, this model allows us to move people quickly without changing managers. Our existing organizational functions included Engineering, Product owners, Architect, the QE team, the Service engineering team, a Build & Release engineer and Program manager. The model shown below was a starting point and we have been continuously making adjustments to optimize it.

In the Enterprise POD diagram below, Mai is POD leader. She is the leader for all engineering resources in this POD. Yukari is Program Manager for the POD who can influence the entire POD.
We also made changes to the role of Engineering manager. With our new model, Engineering managers work on multiple products across multiple delivery teams. Managers became servant leaders to support teams rather than owning one product and driving delivery. They give guidance on technology and career advancement, and groom engineers to get to next career level, while at the same time overseeing multiple delivery activities, coordination and problem escalation. This model enables managers to grow to handle larger responsibilities by acquiring a wider range of product and technical knowledge. They also started to learn how to prioritize, manage and resource a bigger portfolio. This was a huge benefit not only for managers but also for engineers. Now an engineer can change teams without changing managers. Relationships and track records stays with the same manager for a longer period of time, they don’t have to re-start from scratch with every product change.

Next we identified a Tech Lead (TL) for each team. Tech Leads are in charge of leading technical conversations and decisions. They are also responsible for sharing knowledge to the team. An “expert” in the old structure became a tech lead. Along with this change, we realized that behavior must change to be successful in our new model. Otherwise a TL will be once again become a single point of failure. This change has provided opportunities for engineers to move freely across teams and gives them opportunity to speak up without fear of management censure. We now find our teams have been collaborating better by sharing experience and best practices.

We’ve also embedded Quality Engineers (QE) under Engineering to remove the separation between coding and testing. It also provides better career advancement opportunity for QE. Our goal for the QE team was to become an engineer with quality expertise and thought leadership. Our engineers were not used to writing tests in the past, they just coded and passed it onto QE. Influencing engineers to raise their quality standards and awareness has been very important. It drives quality code, quality automation and lower maintenance cost. Most important, is that our customers will be happier.

However, this process and role transitions were not easy for managers who had been executing under the old model for many years. Some managers were having a hard time letting go of control and delegating decision making to the team. We started initially with a small team, and then expanded these role changes to a bigger scale. Once managers understood both the value and benefits of this model, especially after understanding the part about “ability to scale bigger easier”, then this transition got easier. Our organization model is really designed to welcome changes anytime with less overhead.

The architect is in charge of holistic design of the tech stack and supports tech leads in design, decision-making and technical grooming. We’ve established a forum where architects and tech leads gather together and discuss random agenda items. This forum enables the team to stay on the same page in technical direction, implementation details, Q&A and status updates. Tech Leads don’t always know reasons behind bigger technical decisions, so we find it is extremely important to have this forum; otherwise each team will make their own decisions which may incur tech debt that later can become expensive to remove.

Just like the Architect, there are other roles in this POD who serve multiple teams. The Build & Release engineer was previously in charge of build and release activities only. After the change, this role expanded to support architecting CI/CD design and flow and implementation. He talks to each team and helps drive CI/CD to completion, which is very hard to do. The Quality Lead is in charge of establishing our quality thresholds and automation framework. The Service Engineer role stayed mostly the same but interactions and communication has improved. They used to work as a separate operation team to support day-to-day operations and not a part of product delivery team. But now they are more involved and working as extended delivery team than operation only team. They are invited to all team meetings and involved with all communication channels, which was not the case before.

The Program Manager role is another horizontal role which is responsible for orchestrating all the activities listed above by helping teams to run day-to-day business without losing the enterprise viewpoint, establishing the business rhythms, and keeping the pace across teams, while guiding the teams to the same direction and finding ways to fill in gaps wherever needed.

The program manager does not force the team to follow any strict process but suggests minimum guidelines so even a new team can onboard and start quickly. The process should only be lightweight or have little overhead, the team must adapt these guidelines to be tailored to their needs and mature towards self-organization. The program manager always encourages positive attitudes, good communication, continuous
improvement and incremental delivery. Incremental small delivery with fast feedback loop cuts down overhead, it always helps the team to reiterate these messages.

Our enterprise POD model was built incrementally with lots of trial and error. After two years it is still evolving. When team dynamics change and evolve, our model also needs to be adjusted. For example, our current model does not require the QE Lead and Build & Release roles. Their skills, knowledge and mindset were transferred into each team. All engineers now are writing automation and maintaining the release pipeline. We will keep making adjustments as we go. When introducing new changes, they must demonstrate a value with measurable data point. In our case, each team completed full CD with our automation framework with good test coverage. We started with defining MVP test coverage by listing what typically QA does to certify a release. Those test cases became our MVP test suite, good enough to certify a release. Then we have been adding more test cases as we go. All new code must be accompanied by new automation. Now each team is in charge of maintaining the pipeline and keeping up test coverage. Shifting responsibilities to the delivery team with automated tool eliminates a need for a dedicated release engineer and QA lead.

5. HOW TO COME UP WITH IDEAS AND MAKE TEAM DECISIONS

We always want to the team to make their own decisions but when there are so many teams working together to deliver one product, there has to be common understandings across the group in order to be effective. Otherwise, you will have “too many cooks in the kitchen” experiences; no decision will be made in a timely manner. One team’s great idea may break something for some other team. That is the biggest difference of running an enterprise agile team versus small self-contained agile teams. It is impossible to reflect 100+ people's decisions and coordinate through one product in a timely manner without holistic alignment and coordination. In order to ease our pain, we’ve set up a forum where leads from each team and managers attend on a daily basis to share their team's voice, new ideas, experiments done, and also discuss potential collisions and conflicts across teams. Common changes and decisions made by this forum are based on feedback from each delivery team communicated through tech leads. This forum is where the Enterprise POD team aligns and makes collaborative decisions. We also invite team members from the
delivery team, as needed for deep dive discussions. This forum has no set agenda; the team dynamically comes up with agenda items. We utilize this forum to share any knowledge in general, announce things such as end of life product and tech updates, what not to do, etc. Information shared in this forum is echoed back to each delivery team. Decisions and changes agreed upon are always the result of collaboration following simple guidelines as opposed to strict and detailed rules. Each team adapts and makes further improvement. This forum has been extremely valuable to help everyone staying connected and providing an open feedback loop for the entire Enterprise POD.
Ground rules for the meeting were established to help align everyone and avoid confusion. When leads are not talking one language, the team gets confused and things get worse when information goes outside of the Enterprise POD. Here are our ground rules:

1. Be an advocate of the team and ground rules throughout the organization
2. Team discusses and makes agreed upon decisions first; then speak as “One Voice” to other outside orgs/teams
3. Silent is equal to agreement (including no comments to shared docs)
4. Create visibility and accountability among the team
5. Fully commit to the team’s strategies and decisions, internal conflicts stays within
6. Respect everyone on the team, assuming good intentions
7. Contribute to the team's success
8. Each meeting will have a clear agenda and objectives
9. Suggest vs. criticize
10. Come prepared and contribute
11. Docs are working docs that are owned by all members
   a. Draw designs, create proposals, contribute ideas, propose solutions, provide suggestions/alternatives, give feedback, suggest priorities, callout dependencies/risks
12. Share learning/findings to the rest of your team for transparency, org alignment and focused engineering execution

6. LESSONS LEARNED

Successful transition requires buy-ins from both top and bottom. First, leveraging influence is a key. We were fortunate to get buy-ins from our SVP and VPs who supported our ideas all the way. They had a big influence on the people involved; getting support from upper management meant getting more power to influence change. Then we focused on managers who can influence their engineers. Explaining and providing benefits to each stakeholder during the process of change is also very important. People will not change their behavior easily especially when they do not see any benefits to change. Our new model must mean something good for
them. Once a team gains momentum, then they will run fast and grow on their own. Our transformation job is almost done! The biggest lesson we learned from this experience is that changing process and culture requires everyone's buy-in. Start small and make sure everyone in this small group is aligned and good with the changes. People in this small successful group will positively influence others. Sustainable behavior change must happen without using authority.

With this experience, we've improved our:

- Ability to quickly move teams around without manager changes
- People management aspects by keeping same managers regardless of changes in delivery teams
- Managers ability scale bigger easier
- Delivery cycle by shortening it and delivering with higher quality
- Organizational alignment under Enterprise POD
- Transparency and knowledge transfer
- Collaboration, culture and team dynamics

7. ACKNOWLEDGEMENTS

We would like to thank the following people who helped made the transformation successful.

- **John Matheny, SVP Search Product, Engineering and Operations.** John was our executive sponsor who encouraged and fully supported the transformation to agile, to automation, to CI and CD. His endorsement was key to getting organizations aligned and resourced.
- **Glenn Beeswanger, VP Search Engineering and Operations.** Glenn provided the opportunities, budget and training plus any resources that we needed to rollout the transformation. He was our biggest cheerleader and our success could not have happened without his full engagement and support.
- **Mason Ng, VP Search Product Management.** Our transformation would not have been successful without the engagement, collaboration and flexibility of Mason and his organization. Mason's consistent presence in many of the cross-functional meetings showcased his support of the new working model which resulted in a much easier and faster adoption.
- **Our Web Search Engineering team who ensures our products are built with high quality, with full automation and end-to-end CI/CD.**
- **Rebecca Wirfs-Brock - Our paper’s Shepherd for her guidance, insight, review and edits.**