6 days/week with 20hr of engineering/day and multiple releases in one day - Yahoo Search

YUKARI HASEBE, Yahoo!
MAI LE, Uber

Enterprise distributed agile model sounds very hard and complicated. Yes, it is not easy but there are ways to make it work. When we joined the Yahoo Web Search team, we inherited distributed teams working in a traditional development style with SE and QE located in Bangalore, Engineers in Taiwan, and Products in US. Our Agile transition started with a process change from distributed traditional model to distributed agile model. Our team was able to come up with a working model where we can release minimum of six days in a week with more than twenty hours of work hours a day between US and TW office.

1. INTRODUCTION

When we joined Yahoo Search team, we started from just two of us (Engineering Director and Program Manager) in the US, and engineering teams in Jordan and India. Yahoo Search is a very big organization within Yahoo, we own Yahoo Search Box, the Search Result Page, partner syndication, toolbar, browser extensions, and all backend processes such as ads federation, content selection such as wiki, local store listing, images then ranking and slotting information to the right place. Mai was in charge of all aspects of engineering including resource planning, hiring and training on top of product delivery. I was in charge of portfolio management, setting up the team structure and process, and coaching with an emphasis on quality. A few years later, we have self-organized and high performance teams located in both US and Taiwan. There were multiple organizational changes during this time. Regardless of multiple challenges to overcome, now our team's productivity has improved to 6 days a week of engineering, averaging 20 hours a day with CI/CD.

2. BACKGROUND

Yahoo is a very interesting place to work. We are known for many different reasons but Yahoo Search generates a good portion of revenue to the entire organization, with millions of search users everyday. Our goal is to grow user engagement that leads to more revenue. Meeting revenue goals is a lot of work with lots of pressure especially when our organization itself is going through major changes. We entered the Yahoo Search world without much knowledge of complexity in the existing business, technology and resourcing model. Both of us transferred to Search in fall of 2013 from the media organization who owns homepage and vertical pages such as News and Weather. We only knew that we always have Search Box on our pages but did not now much about what happens once you start typing words and hitting the enter key. We've learned that the tech stack has multiple deep layers and components to return information and ads. Search logic behind the box takes multiple information such as user information and locale into consideration to decide how to collect data, rank and select what to display and where to place it. We set a goal of establishing a sustainable high performance team and a process that would work for everyone. We've worked through multiple constraints and we have been able to come up with a model that works for us. Your organization may be able to adapt this model and gain some benefits. This is our two-year journey of building new teams across the world with engineering excellence.

Yukari Hasebe, email: yukari@yahoo-inc.com
Mai Le, email: mai.le@uber.com

3. WELCOME TO THE NEW WORLD

We entered Search distribution business with a mission to grow business. Where do we start? The first thing we did was ramp up. Learning products and organization by going over portfolio, products, teams, headcounts and process to understand the current status before starting anything new. We found out we did not have any dedicated Engineering resources in Sunnyvale headquarters. Teams were located in India, Taiwan and Jordan. Luckily, one Product manager was located in Sunnyvale and there were borrowed engineering resources with domain expertise in Sunnyvale and Taiwan. We then started massive hiring efforts, recruited from outside, inside, college grads and interns to build high performing engineering team in US headquarter; there was zero engineer in US when we joined. While we spent our energy on hiring, we also had a business to run. We started to look into engineering process and found multiple areas for improvement. One team is completely distributed across the world: Product in Sunnyvale, one engineer in Taiwan, one Quality Engineer and Production Engineer in India. The team had been releasing once or twice a month and there was no automation running in release pipeline. Team in Jordan had been maintaining the product but not developing any new features. Another team in India was delivering but not with any clear strategy. Little by little, the team grew bigger with lots of training. Our US team became a solid core team within six months and we added many more new team members across the world. We've reorganized function and team structure, initiated an agile transition and adoption which works across distributed teams. Introducing Agile/Scrum itself was not so hard but establishing sustainable best practices has always been a challenge. There were many iterations of trial and errors but we have been making continuous incremental improvement and hoping to stay on the right direction. We are sharing some of the challenges we've encountered and how we've addressed them along the way.

3.1 DIFFERENCES IN COMMUNICATION STYLE CAUSED MISUNDERSTANDING AND DELAY

When starting new teams, often times we forget to confirm what we think is “common sense” or “norm” and make assumptions without thinking too much. When assumptions are not true, we all get frustrated. Assumptions in communication are very dangerous especially across the time zones. For example, expressing “No”, “Not clear,” or “Do not understand,” is treated as weakness in some cultures, and people tend to keep silence. During the backlog grooming, people were not asking questions and we assumed the team understood enough to get going. However, it was not always the case, we found out things were blocked and left alone without any red “blocker” sign raised.

In another culture people tend to say, “Yes” to everything. People voluntarily took many stories during our planning meetings. So we set the WIP limit to three, but the next day when you looked at the board, the number had increased again. We found out people were taking up an unrealistic amount of work. And within a sprint, work was not getting done properly or work was completed differently from what was expected. We identified this type of communication challenge as something to be fixed right away. We have been experiencing many redundancies in communication. It was impacting turnaround time significantly. We were having a hard time understanding each other even when we are all in same room; adding distance, language barrier and culture can multiply the effect.

3.2 All major work and release overhead was staying in US team

When a new big feature came into the plan, often times the US team was requested to do the work. It is convenient from a coordination, management and product prospective. Important work had been executed in US while remote offices got lighter weight work. Another gap was also in the release process. The release overhead was heavy and unreliable; we've started a release master rotation across the country hoping to balance the load of release overhead. A release master is in charge of overseeing release activities end-to-end starting from checked in code going through unit test, all build and test jobs to deployment to production. When the build pipeline failed, the release master is the first person to troubleshoot issues and coordinate with engineers who broke the pipeline. However, US engineers were carrying more weight than remote offices due to gaps in knowledge and convenience. When an issue was found during the release, the engineer who is in charge of the release will look into it first but often times certain groups of engineers with deeper domain expertise were getting called anyways. While US engineers were working overtime feeling tired, remote engineers were feeling underutilized and under appreciated.

3.3 Priorities were mixed up across time zones

Priority keeps changing, this is considered as business as usual, especially at Yahoo. Sometimes it can change multiple times in one day. Remote teams had often taken the wrong stories to work on because urgent changes were not communicated clearly and Product assumed things were clear. This built frustration at both US and remote offices. The US team felt it took too much time to keep updating priorities, and remote teams felt they were not getting enough and up-to-date updates about changing priorities. Sometimes engineers were asked to quit what they have been working on to switch to higher priority work.

3.4 New team member cannot produce for a long time

Any new team member could not produce for a long time. Our tech stack is old and complicated with lots of technical debt. When a new team member joined the team, we did not have good on-boarding process to share, so it was taking weeks even for US engineers to even set up their own dev. box before learning the product itself. Imagine, what can go wrong with this situation with remote teams. It was not only taking a long time to ramp up but also leaving a big margin for error and misunderstanding. So basically, we kept building tech debt during the process.

3.5 Can't release as planned

Our testing and release process was 100% manual. Release issues often became the bottleneck and a release was put on hold until an issue was resolved. We've taken a big initiative to convert to CI/CD to mitigate this issue, that part was great. However because US engineers did most of the release work originally, the release was still blocked when CI/CD pipeline was broken during non-US business hours. Many tests were written by the US team without much knowledge transfer to remote offices therefore troubleshooting of any pipeline failure became a bottleneck for the release.

4. WHAT WE DID

What was listed above is not everything we had problems with; we have been encountering new challenges but we are determined and persistent in making continuous incremental improvement to areas that can be improved. We ranked these items based on feedback including ideas from a retrospective. Higher pain point items with lower efforts to fix were taken first. When introducing a new thing to the team, we need to make sure the team is not overwhelmed with too much information. As leaders, we have to assess the team's capacity and only provide what the receiver can digest. Otherwise, it will end up costing more as opposed to gaining benefits.

4.1 CHANGING COMMUNICATION STYLE AND MINDSET

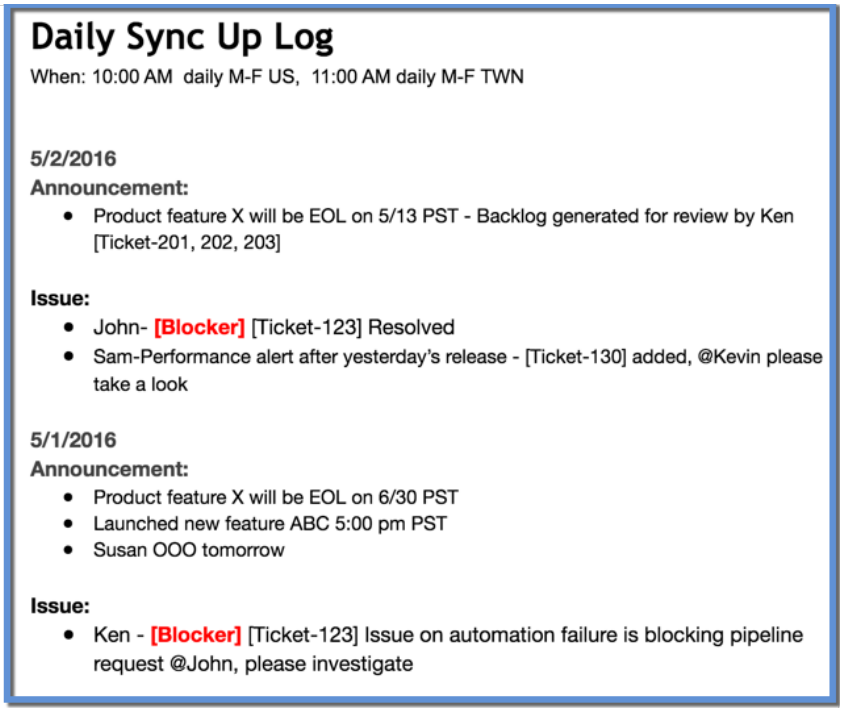
We've addressed our communication issues by doing the following things;
Encourage team members to speak up and communicate by providing safe environment to speak up without fear of management, influencing equal opportunity for the teams. Engineers usually like to code more than talking, and many engineers were not familiar with speaking up in front of everyone before daily stand ups were introduced. That was already a challenge. Speaking up and asking questions was not what they had done in the past. They were used to being assigned work as an order and just execute it. Questioning itself could be considered negative in some cultures. It could demonstrate weakness in skill or not cooperating. We've repeatedly encouraged open communication, explaining it is okay to ask questions whenever we can.

Second point we've addressed together with open communication is to promote end-to-end accountability by making each team member own stories from grooming to production delivery. This encourages people to ask the right questions at the right time. Engineers were used to delivering a "task" of one implemented story rather than delivering "value"". By reinforcing end-to-end accountability, engineers started to see and think differently. Good questions were started being asked during backlog grooming. This also helped making engineers only take on an amount of work they can deliver.

We utilize collaboration tools to make backlog and progress visible to everyone and help improve communication. A wall is great for a co-located team but not for distributed teams, therefore, we must take advantage of technology. We've used Google sheet, Rally and Jira to manage Scrum backlogs. Any collaboration

tool will help as long as clear guidelines are defined and used as a source of truth, otherwise it will just add another management overhead. Let's not duplicate the same information in different places to avoid confusion or introduce gaps. We've also used video conferencing, desktop sharing and chat rooms to enhance closer interaction. These tools significantly helped shorten distance. Productivity improvements between videoconference and teleconference are big. Observing body language enhances the communication experience.

Adding a shared daily sync up meeting log to keep track of conversation as reference point has been helping us a lot. Not everyone has the same way of understanding especially when English is not the first language. This document is very simple, it is not a comprehensive big log but only summarizing important points such as important announcements, new defects which have downstream impact, requests, dependencies and vacation plans, etc. This log captures the context of daily stand ups that happen in each location, and the level of detail can be adjusted as needed. We had more lengthy notes than what it listed in below at the beginning but as teams matured over the period of time, they no longer need the same level of detail. The teams use this document during daily stand up and provide important updates to the other teams.



Daily Sync Up Log

When: 10:00 AM daily M-F US, 11:00 AM daily M-F TWN

5/2/2016

Announcement:

- Product feature X will be EOL on 5/13 PST - Backlog generated for review by Ken [Ticket-201, 202, 203]

Issue:

- John- **[Blocker]** [Ticket-123] Resolved
- Sam-Performance alert after yesterday's release - [Ticket-130] added, @Kevin please take a look

5/1/2016

Announcement:

- Product feature X will be EOL on 6/30 PST
- Launched new feature ABC 5:00 pm PST
- Susan OOO tomorrow

Issue:

- Ken - **[Blocker]** [Ticket-123] Issue on automation failure is blocking pipeline request @John, please investigate

Example of Daily Stand Up Log

4.2 Knowledge transfer and streamlining work

It is important to provide consistent knowledge transfer to transition skills and domain knowledge across the team to minimize knowledge gaps. We've invested a lot in knowledge transfer. We send engineers to remote offices and remote office engineers visit HQ every quarter. One time checklist type knowledge transfer will not work on a technical transition. It has to be planned and prepared with ramp up time in order to digest, understand and apply new knowledge. We've utilized collaboration tools and videoconference to increase productivity. First we set up an overview session to provide high-level context followed by deep dive sessions. We've learned a face-to-face session is effective for deep dive sessions. Once engineers digest information, they may ask for more sessions. During ramp up time, it is important to fully support ramp up activities. Assigning a point of contact and setting up office hours is helpful to avoid excessive and redundant communication. A point of contact engineer cannot spend all day just to answer questions; he/she has to deliver, too. It is important to keep providing support as needed even after transition and not just cut off communications just because the transition is over. We ramped up remote offices one area at a time and gradually have been transferring more

heavy weight work to balance workload across the team. We were able to transfer release activities to remote offices and that was a huge victory for us. After the transition of release work to remote offices, productivity increased significantly. Now our releases can chase the sun!

4.3 Priorities were mixed up across time zones

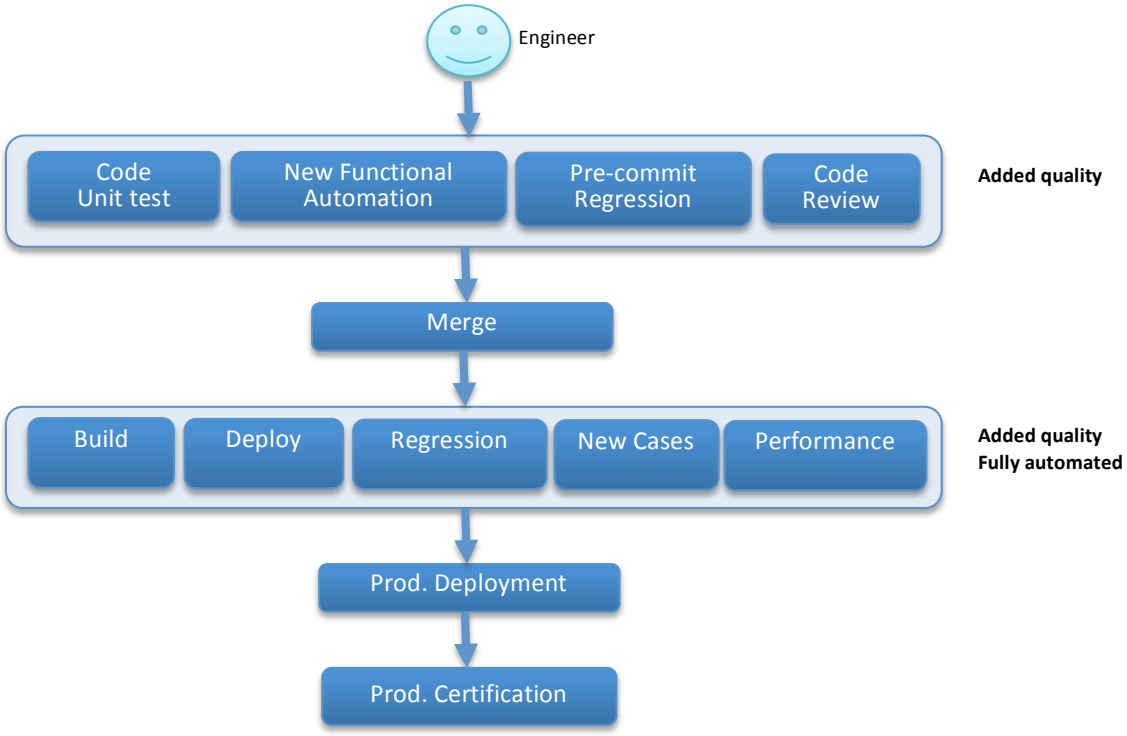
Keeping on top of priorities can be challenging especially when there are multiple product managers and business units are involved. In a perfect world, one company has unified priority, but that is not the case in the real world. Product team assumed everyone understands expectations and things are clear, which is not always the case. They wanted everything to be done by the deadline so prioritization was not important for them. They assume everything will be delivered so they wanted engineers to figure out what to do first, therefore backlog priority was not being maintained. One engineering team with multiple product owners who wanted everything confused our engineers. We've since educated the Product team to be on top of changes, to reflect them in tools accordingly, and to communicate priorities during sync up meetings. Now they are working together to agree on priority among Product first, before sharing with our engineers. This has helped minimize throwaway work.

4.4 Establishing our onboarding process

We've streamlined the on-boarding process to be more intuitive. There was no onboarding process when we started, engineers were asked to do lots of documents mining and reverse engineering. They found outdated documents not stored in one place but all over the place, and it took time to validate information in those documents. It was great if it worked, but if it didn't, they'd have to figure out if something wrong with them or the document was outdated. We've consolidated scattered information into one streamlined document and saved in team's network drive. New members are responsible to make improvements to it and add more information and suggestions based on their own experience. Now, setting up a dev. box, which used to take weeks, can be done in less than one day.

4.5 100% Manual to 100% automated CI/CD pipeline

With 0% automated release at the beginning, it took a lot of time and resource overhead to release. We were lucky if all test cases were run once per release. Regression test on different browsers, versions and countries are very time consuming. There were no unit tests, and uninspected code was getting merged. To achieve true CI/CD required discipline and mindset changes. The hardest part was to change the mindset to fail fast before things hit the main pipeline so a release does not get blocked. The team was used to a "just write and figure out at the last minute" release model. Releases were unpredictable because of this behavior. Then our team landed a contract with a 48-hour change request resolution guaranteed to the partner. This contract forced the team to go to CI/CD in order to be able to deliver within 48 hours. The team spent lots of effort to write reusable automation tests so anyone can run them anytime over and over. The CI/CD effort was big, the team worked very hard and within two months for this product area, we were able to run 3000+ test cases in less than 40 minutes. Now our "Definition of Done" includes unit tests, functional tests, pre-commit regression tests and code review. Engineers became accountable if they break the pipeline, so they started to test their code more and run pre-commit tests. The code reviewer is also accountable, so the entire team began to pay more attention to code reviews. This prevents bad code from getting merged into our pipeline. There were no checkpoints in the past and the release branches were often getting blocked. With knowledge transfer and a streamlined work process as mentioned above, the pipeline has been consistently monitored and maintained by all team. This model enabled a release to go out anytime.



Example of CI/CD Workflow

5. LESSONS LEARNED FROM EXPERIENCE

Key lessons learned from our experience are:

- Change communication style by audience type - We have to analyze and adapt our communication style in order to be effective. When teams are used to taking orders from managers and individuals are not used to being able to voice their opinions, then encouragement must start from the managers. Don't assume but rather confirm the details.
- Stay in touch and keep aligned - It takes a lot of time and efforts to align and stay aligned. Knowledge transfer and cross training must happen frequently to prevent gaps from becoming wider. Sync up frequency must be at least once a day to identify issues, fail fast and course correct immediately.
- Keep clear priority - Things change quickly and organizational priority overrides team's priority. This type of change happens often and even multiple times a day in some cases. Backlog priority needs to be clear at any given moment to avoid confusion and delay.
- Proof-of-Concept first then multiply - When trying new things, start a small POC first before adding-on/multiply. Rolling out new things to entire team at once is very risky. Technology upgrades, refactoring or changes in the pipeline can be tested by a small team first to identify gaps and support processed before rolling out to the whole team. It is always safe to start small so when something went wrong, both business and engineering impact can be minimized otherwise it is very risky especially in enterprise scale.
- Maintain a clean CI/CD pipeline – The pipeline must be clean with good quality coverage. Everyone is accountable for maintaining a clean pipeline. Accumulating bigger changes will incur more risks and support overhead so keep the release small and release frequently. CI/CD requires behavior change, otherwise it will not be successful.
- Effective onboarding process - Onboarding process should be tailored to the needs of the audience and must be regularly updated to be reusable in order to scale rapidly.

6. 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 ensured 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, reviews and edits.