Accelerating Organizational Growth with Inspiration from Parliamentary Procedure

ANASTAS STOYANOVSKY, IBM Watson
WILLIAM CHAPARRO, IBM Watson

Transitioning a growing development team into smaller, cross-functional sub-teams is challenging, particularly until a collaborative decision-making process for shared concerns is agreed upon. Within the framework of XP, we hypothesize that a core part of this challenge comes from a fracturing of shared understanding as those sub-teams form and storm. In order to navigate this process and re-establish that shared understanding, we take inspiration from *Robert's Rules of Order, Newly Revised* (RONR), which is the most widely used manual of parliamentary procedure in the United States.

Some of the specific inspirations we drew from RONR were to establish a formal charter and bylaws, treating our wider development team as a formal organization; we reimagined and redefined these and other formalities in terms of agile ceremonies and XP practices. Herein, we describe the integrated framework we developed and will share our experience implementing it at IBM Watson to foresee and improve organizational scaling efficiency, accelerating the onboarding process and allowing our organization to grow more efficiently.

1. INTRODUCTION

Organizational efficiency and scaling are universal challenges, due in part to communication costs that increase as up to the square of team size (Brooks, 1995). In this experience report we will focus on our approach to the specific case of scaling a growing team of software engineers to multiple cross-functional teams. After our team had grown to 14 engineers, we split it into three smaller, cross-functional teams. We immediately began observing inefficient communication of significant design and architecture decisions after they were made, and sometimes deadlock in even reaching agreement on such decisions. We hypothesized that, in the terminology of eXtreme Programming (XP) (Beck, 1999), the shared understanding of the initial team had been eliminated by the team split, and that the observed problems would worsen after another significant increase in headcount. To test this hypothesis, we looked for an approach to re-establish, maintain, and spread shared understanding, and then tested this approach during another onboarding event after which head count increased by 50%.

Our approach was directly inspired by *Robert's Rules of Order, Newly Revised* (RONR), which is a handbook on formal rules for a wide range of types of organizations that was first published in 1876 and has been continuously revised since (Robert III, Honemannn, Balch, Seabold, & Gerber, 2011). Based on parallels between common agile/XP practices and rules described in RONR, we constructed an integrated framework that draws on empirically effective forms of self-organization described in both these sources. We defined a team charter defining the team's mission, in analogy to a formal organization's charter, and codified tribal knowledge of the team's development process, in analogy to a formal organization's bylaws. We focused on addressing three specific areas: (1) efficiency of onboarding, (extending shared understanding to new team members); (2) establishing efficient communication, (maintaining shared understanding), and (3) establishing efficient decision making, (efficiently creating new shared understanding). We report moderate success with respect to (1) and (2) and mixed results with respect to (3).

2. RELATED WORK

There has been much written about scaling agile development since soon after the publication of the Agile Manifesto (The Agile Manifesto, n.d.) (Eckstein, 2004), including named frameworks such as AUP (Ambler S. , 2019), DAD (Ambler & Lines, 2012), and SAFe (Leffingwell, Knaster, Oren, & Jemilo, 2018), which try to address organization past the single team level. These and others tend to build on the Agile Manifesto and/or XP. Another approach to scaling agile organizations has been given in what is often referred to as "the Spotify

Anastas Stoyanovsky: astoyan@us.ibm.com
William Chaparro: wchaparr@us.ibm.com

model" (Kniberg, 2014). An interesting example of large-scale self-organization of teams working on open-source software is the Kubernetes governance model (Kubernetes Governance Model, 2019).

3. BACKGROUND

IBM Watson Discovery Service is a platform that provides information retrieval capabilities as a cloud service and is implemented in a microservice architecture. It is staffed by ~5 largely autonomous teams that coordinate only on high level business goals and each of which has essentially complete ownership over the entire software development lifecycle for the microservices they own. The authors' team is responsible for runtime information retrieval components, as well as the machine learning model training and serving lifecycle underlying the involved artificial intelligence capabilities.

The initial release of this set of services was implemented by a team of 6-8 engineers. After that release, another team of approximately the same size joined, as well as its development manager. At first, the development team was not structured cross-functionally, in that an engineering team would consist entirely of engineers reporting to the same manager. After approximately 1.5 years in production, these two reporting chain based teams, comprised of 14 people total at that time, were reorganized into three smaller cross-functional teams whose structure was not related to reporting chains. Of these 14 engineers, only five had been on the team throughout the entire product's development; these five, together, had complete knowledge of the entire product, but were distributed amongst these three new teams, at least one per team.

Within days of separating into cross-functional teams that each consisted largely of junior members, organizational problems surfaced. There was no clear decision-making process: whereas architectural decisions had previously reached consensus and been recorded with little controversy, there was no longer a clear path towards consensus in cases of disagreement. This is exemplified by a technical proposal made by one team regarding a decision affecting all teams and that received 56 comments, most of the last 30 being an argument between two people from different teams. This discussion reached a stalemate for weeks.[1]

Each individual team had incomplete familiarity with the entirety of the architecture; a recurring problem was one team designing new functionality without being aware of its implications on components owned by a different team, leading to slow design processes and/or design flaws. It became clear that alignment between teams could not be relied upon to happen naturally and that it was necessary to implement some remedy in a timely manner. At the same time, a third development manager and his direct reports were preparing to join the growing organization as well; adding one or two more members may not have had a large impact, but increasing head count by another 50%, before having solved these fundamental problems, would likely have caused significant disarray in the organization and significantly delayed its development roadmap.

4. AIMS

Under the abstract hypothesis that the root cause of our observed organizational problems was a fracturing of shared understanding, we chose three specific short-term Aims with respect to which we would be better able to directly measure success:

- Aim 1. Minimize onboarding time for new team members.
- Aim 2. Ensure that understanding of our development process exists amongst all current team members and propagates efficiently to new ones, especially that understanding which had previously existed as tribal knowledge.
- Aim 3. Ensure that significant design, implementation, and architectural decisions are informed, reached through consensus with the necessary stakeholders, and have a method for dissemination once made.

To work towards these Aims, we drew from RONR in order to extract sets of concepts and principles that are empirically effective for facilitating self-organization of large groups and then integrated those with agile/XP practices.

[1] The stalemate can be best illustrated with a comment of "It doesn't sound like there is a solution that we will agree on, so I'd like to move this to some sort of internal arbitration process that can impose a solution without requiring a consensus," being met with "This is the internal arbitrage process, right here."

[2] RONR discusses corporate constitutions and bylaws: "In general, the constitution or the bylaws - or both - of a society are the documents that

5. MOTIVATION

While discussing how to deal with emergent organizational inefficiencies and prevent further ones, we noticed a number of parallels between XP/agile practices and RONR. The initial motivating observation was that RONR specifies that "the bylaws of an organized local society usually provide that it shall hold regular meetings at stated intervals... and also usually provide a procedure for calling special meetings as needed" at which those bylaws can be modified.[2] Historically, our team had used agile retrospectives as an opportunity to reflect on and experiment with our development processes; if one views the development process of a team as being analogous to an organization's bylaws, then agile retrospectives would correspond to those periodic meetings. A fundamental part of onboarding new team members would, in this terminology, involve familiarizing them with those bylaws. This prompted us to look for more analogies with the structures and principles described in RONR.

5.1 Robert's Rules of Order

RONR begins by defining and describing several different types of deliberative assembly: mass meetings, local assemblies of an organized society, conventions, legislative bodies, and boards. Of these, the convention and the legislative body were not relevant to our case.

- A **mass meeting** is "a meeting of an unorganized group that is announced as open to everyone (or everyone within a specified sector of the population) interested in a particular problem or purpose defined by the meeting's sponsors, that is called with a view to appropriate action to be decided on and taken by the meeting body."
- A **local assembly** of an organized society is "the highest authority within such a society or branch (subject only to the provisions of the bylaws or other basic document establishing the organization), this body acts for the total membership in the transaction of its business."
- A **board** is "an administrative, managerial, or quasi-judicial body of elected or appointed persons that differs from several of the other principal types of deliberative assembly as follows:
    - "boards are frequently smaller than most other assemblies, and
    - "while a board may or may not function autonomously, its operation is determined by responsibilities and powers delegated to it or conferred on it by authority outside itself."

RONR recounts that new organizations are generally established at a mass meeting. It goes on to give general principles of definition for formal organizations, suggesting, at a minimum, a charter and a set of bylaws; an organization's charter defines the organization's mission and cannot be changed, while its bylaws describe its daily functioning and can only be changed at periodic local assemblies.

6. IMPLEMENTATION

In order to work toward Aims 1-3 for what had become multiple teams and whose total headcount was about to increase by 50%, we first established a source control repository in which to encode tribal knowledge of our development practices. The first pieces of text committed to this repository were drafted collaboratively among the more senior team members, and that draft was then brought to all 23 members to finalize and endorse during what roughly corresponded to a mass meeting. The initial commits to the repository added a charter, a list of bylaws, and a list of component architects; the latter is described in *Component Architects and Expertise* below.

We were careful to not attempt to overextend analogies with what is described in RONR. For example, the rules of order for conducting a deliberative assembly seemed like they would have leaned more towards "process over people" for the size of our team, in contradiction with the "people over process" value given in the Agile Manifesto. Those rules of order include rigid mechanisms for bringing motions to order, seconding motions, and numerous other details which we deemed overly constrictive for us at that time.[3] We did,

[2] RONR discusses corporate constitutions and bylaws: "In general, the constitution or the bylaws - or both - of a society are the documents that contain its own basic rules relating principally to itself as an organization, rather than to the parliamentary procedure that it follows. In the ordinary case, it is now the recommended practice that all of a society's rules of this kind be combined into a single instrument, usually called the bylaws...." We opted to follow this recommended practice.

[3] There are further, finer-grained formalities, such as agendas and meeting minutes, which are already common in some form and are used to varying degrees to suit the working habits of the people within a particular organization. The concept of a call to order, for example, manifests

however, identify some constructs that seemed potentially useful to codify were the organization to grow larger; those are discussed below in the *Possible Extensions* section.

6.1 Charter

The first step we took was to collaboratively define an explicit team charter, with the goal of keeping it to one sentence. Per RONR, a formal organization's charter cannot be changed for organizations that exist legally; we felt that, in principle, the concept of a charter corresponds to an engineering team's identity, purpose, and purview. Since these can change, we do not expect to treat our charter as immutable, but rather as slow to change.

Based on previous experience, effectively communicating the team's general mission, scope, and responsibilities during onboarding of new members would normally have been a half hour of improvised, unorganized communication of tribal knowledge, followed by a question and answer session. This would have added up to over one work day's worth of time, totaled across 23 people in this case. Instead, after having taken time to craft a clear and concise charter[4] that had been endorsed by all previously existing team members, we were able to establish that shared understanding with a one-way message that was delivered in seconds. This was a small but significant first step towards Aims 1 and 2; obviously, minimizing time spent in meetings that include 23 people is advantageous.

6.2 Bylaws

As mentioned above, RONR defines the bylaws of an organization as "the documents that contain its own basic rules relating principally to itself as an organization" and we chose to define our bylaws as our development and support practices and processes. RONR also specifies that "the bylaws of an organized local society usually provide that it shall hold regular meetings at stated intervals" at which its bylaws can be modified, which, again as mentioned above, seemed to correspond naturally to our agile retrospectives.

The first draft of our bylaws was written collaboratively by the more senior team members and their development manager, and was finalized during an hour-long meeting including all members of teams in our larger organization. An early decision to make was regarding what level of detail to target for our bylaws; keeping in line with the Agile Manifesto value of "people over process," we kept our bylaws concise and minimal, and we tried to arrive at them descriptively rather than via an opinionated mindset. We attempted to encode methods for efficient decision making (see *Component Architects and Expertise* below) and for low overhead of communication of decisions after they are made (details given below). Finally, we were careful to avoid restricting team autonomy in any way we felt not strictly necessary for achieving the above goals. Our bylaws are given in *Appendix I*, redacted as per confidentiality requirements.

6.3 Component Architects and Expertise

We had previously found it beneficial to identify "component architects," that is, one to two engineers per component who have ownership over its design and evolution (Keeling, Runde, & Gala, 2018). In particular, lack of deliberate stewardship of a component had often led to indecision during new feature design and to architecture violations during implementation. We identified the concept of officers of a formal organization, as given to RONR, as an analogy to this concept.

Our bylaws encoded that significant design, technical, and architectural decisions would be reached by consensus that includes both external stakeholders and component architects. In particular, significant architectural decisions would continue to be recorded in Architectural Decision Records (ADRs) (Nygard, 2011), which had already been an established practice for our team (Keeling & Runde, 2017). Taken together, these worked towards Aim 3: we expected that requiring consensus among all stakeholders, who include component architects, would encourage informed decisions. Recording those decisions in ADRs provided a straightforward way to disseminate that information with low overhead.

Alongside the "Component Architects" documentation, we included an "Expertise" section whose goal was to serve as a reference for junior engineers to use to solicit advice on specific technical topics ("Distributed Systems", "Machine Learning", etc.). At the time of these sections' writing, people from the original team were

---

casually when a meeting begins with the statement, "it's time to get started." While perhaps uninteresting on their own, we mention these correspondences as further evidence that the approaches in RONR give name and definition to natural ways that humans effectively self-organize.

[4] While we cannot share our exact charter due to company policy of revealing information on internal structure, its sentence structure is, "we implement, ship, and maintain functionality of type *X* for product *Y* that provides significant customer value." We were surprised at the amount of discussion required to arrive at a wording of our charter that satisfied all existing team members.

included; unfortunately, new team members did not update the "Expertise" section to include themselves, despite repeated requests to do so.

7. EXTENSIONS

The practices and values described in XP are sometimes accompanied by a description of common roles on software development teams. These roles - such as "programmer", "tester", and "coach" - are descriptive in that they are observations of the types of roles that people on successful programming teams tend to naturally assume: "The ones you see today are there because they worked and the other ones didn't." (Beck, 1999). Since RONR is descriptive of structures within and principles of successful organizations, we looked for further analogies between XP/agile practices and RONR.

7.1 Temporary and Permanent Committees

Temporary committees are defined in RONR as small, autonomous groups that form in order to establish a particular task or carry out a particular investigation. A similar idea already exists under the name "tiger team" (Dempsey, Davis, Crossfield, & Williams, 1964), illustrating another convergence between concepts described in RONR and existing self-organizational practices.

One example of a way that this concept might manifest is the establishment of a temporary committee of people with architect roles in order to design a new system. Interestingly, this bears similarity to the "guild" structure in the Spotify model (Kniberg, 2014). We speculate that creating a "guild" type construct is an unnecessary degree of formal structure until an organization grows past a certain size which ours has not yet reached, and so we have not explored this.

After implementing the approach we describe in this report, a permanent committee naturally formed when a team member organically proposed a permanent "tactical squad" whose team members would rotate and whose responsibility would be to maximize support quality. We agreed to adopt that construct at an emergency team-wide meeting that was called to address specific support issues, after which we amended our bylaws accordingly. We note that this is another case of self-organization which emerged in a form described in RONR.

7.2 Board

The drafting of a charter and bylaws was carried out collaboratively by a group of more senior team members. During that time period, it might be argued that this group functioned as a board, notwithstanding that the goal of that team was to make a board-type structure unnecessary. As the organization grows further, it is possible that it will become necessary to establish a formal board; we do not speculate under what conditions that would be the case for an engineering organization contained within a larger corporate structure.

8. RESULTS

We sent an anonymous survey to 20 organizational members (including engineers and development managers, but excluding ourselves) asking their assessment of whether our approach helped them:

1. effectively onboard,
2. understand how the organization operates, and
3. understand how to effectively reach informed decisions that impact others.

We chose question types 1, 2, and 3 to be in direct correspondence with Aims 1, 2, and 3. The survey population was composed of two groups. Our team was composed both of team members who were onboarded using our experimental approach (Group A) and of team members who had previously been onboarded but were present for the onboarding meeting in which we presented the framework described here to new members (Group B).

For Group A, we asked variant A of question types 1-3 (Questions 1A - 3A, respectively) asking how effective they think our approach was; for Group B, we asked variant B of questions 1-3 (Questions 1B - 3B, respectively), asking them to assess how effective they think our approach would have been for them during their previous onboarding.

8.1 Aim 1

Our first Aim was to "minimize onboarding time for new team members."

Question 1A: "Was this an effective way for you to onboard?"
Question 1B: "Do you think this would have helped you onboard?"

	1 (Strongly Disagree)	2 (Somewhat Disagree)	3 (Neutral)	4 (Somewhat Agree)	5 (Strongly Agree)	Mean
Question 1A	0	0	3	4	0	3.6
Question 1B	0	1	2	5	2	3.8

8.2 Aim 2

Our second Aim was to "ensure that understanding of our development process exists amongst all current team members and propagates efficiently to new ones, especially that which existed as tribal knowledge."

Question 2A: "Did it help you understand how the team operates?"
Question 2B: "Do you think this would have helped you understand how the team operates?"

	1 (Strongly Disagree)	2 (Somewhat Disagree)	3 (Neutral)	4 (Somewhat Agree)	5 (Strongly Agree)	Mean
Question 2A	0	1	1	1	2	3.8
Question 2B	0	0	2	6	2	4.0

8.3 Aim 3

Our third Aim was to "ensure that significant design, implementation, and architectural decisions are informed, reached through consensus with the necessary stakeholders, and have a method for dissemination once made."

Question 3A: "Did it help you understand how to effectively reach informed decisions that impact others?"
Question 3B: "Do you think this would have helped you know how to arrive at informed decisions that impact others?"

	1 (Strongly Disagree)	2 (Somewhat Disagree)	3 (Neutral)	4 (Somewhat Agree)	5 (Strongly Agree)	Mean
Question 3A	0	2	1	4	0	3.3
Question 3B	0	0	3	6	1	3.8

8.4 Other Survey Results

We asked all survey participants, "If you have been onboarded to another team previously, how does this onboarding compare?", asking them to respond on a scale of 1 ("Much Worse") to 5 ("Much Better").

	1 (Much Worse)	2 (Worse)	3 (About the Same)	4 (Better)	5 (Much Better)	Mean
Responses	1	0	0	1	2	3.8

8.5 Qualitative Feedback

We included one open-ended question: "What other aspects of the onboarding process could be improved? Do you have any further feedback or considerations?" The relevant responses are reproduced in *Appendix II*, with redaction of our internal team name, of references to specific product features, and comments that are not directly relevant.

9. DISCUSSION

The most commonly mentioned issue in feedback to our open-ended question regarding onboarding experiences was that the technical onboarding documentation was out of date, as was the breakdown of domain expertise by person. The remedy to the former is obvious. With regard to the latter, we noted in the

Component Architects and Expertise section above that we asked the new team members to add themselves to the listings but that they did not do so. Perhaps new team members did not feel comfortable openly self-assessing and declaring their own expertise. We have since removed the "Expertise" section from our documentation.

The mean survey results are positive for Aims 1-3, though to varying degrees; in particular, according to the responses we received, Aim 3 regarding decision making seems to have not been well achieved. It is possible that the failure of the "Expertise" section led to a net negative effect towards Aim 3, especially considering the open-ended feedback mentioned above. Survey results indicate success with respect to Aims 1 and 2.

In response to the survey question, "If you have been onboarded to another team previously, how does this onboarding compare?" there was one vote for "Much worse". Since the survey was anonymous, we could not follow up privately to understand that response; we chose not to extend an open invitation for that respondee to come forth out of a concern that doing so would discourage team members from responding genuinely to surveys in the future. We conclude that our approaches to Aims 1 and 2 served most team members well, but not all, and that, in particular, our approach for Aim 3 needs improvement.

10. CONCLUSION

After splitting a growing development team into smaller, cross-functional teams that were each led by one or two of the previously existing senior team members, we observed a breakdown in communication and decision-making processes. We hypothesized that this was due to a fracturing of the shared understanding that existed on the original team. Due to an upcoming 50% increase in headcount that would likely worsen the situation, we tested our hypothesis, using inspiration from organizational principles given in RONR, by composing an organization charter and bylaws, adopting those across all teams, and using them to onboard new team members. We supplemented this with "Components Architects" and "Expertise" sections, whose goal was to identify architects of specific codebases and experts in specific technical areas, respectively.

We surveyed the team with regard to whether they felt the approach helped them - or, if they had already been onboarded would have helped them - (1) effectively onboard, (2) understand how the organization operates, and (3) understand how to reach informed decisions and communicate them to others. The survey showed positive overall responses for (1) and (2), but only slightly positive overall responses for (3) and with multiple responses in the negative. We used these results as proxy metrics for whether our approach was able to re-establish, spread, and maintain shared understanding of our development process. Based on the results, it seems that our approach was generally effective but can be improved, especially with respect to (3). In particular, the "Expertise" documentation seemed to have served as a net negative, perhaps due to a general hesitation in new team members towards self-declaration of specific expertise.

Finally, based on our chosen proxy metrics, we conclude that the maintenance of shared understanding while scaling one engineering team to several smaller teams is essential to avoid accumulating organizational inefficiencies during that process. Furthermore, we conclude that the approaches to organizational definition and governance suggested in RONR do significantly and effectively converge with agile/XP practices; in particular, deriving an integrated framework based on this convergence both is effective at the organizational scale we describe and is a promising perspective to use when growing to larger organizational scales.

11. ACKNOWLEDGEMENTS

We would like to thank our team at IBM Watson for their open feedback on our approach described here and for their general willingness to experiment with development processes. We would like to thank Jutta Eckstein, our shepherd for this experience report, for her thoughtful, insightful, and patient feedback during the composition of this experience report.

REFERENCES

Ambler, S. (2019, April 13). *The Agile Unified Process*. Retrieved from http://www.ambysoft.com/unifiedprocess/agileUP.html

Ambler, S. W., & Lines, M. (2012). *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise* . IBM Press.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change.* Addison-Wesley Professional.

Brooks, J. F. (1995). *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley Professional.

Dempsey, J., Davis, M., Crossfield, A., & Williams, W. (1964). Program Management in Design and Development. *SAE Technical Papers*.

Eckstein, J. (2004). *Agile Software Development in the Large: Diving Into the Deep.* Dorset House.

Keeling, M., & Runde, J. (2017). Architecture Decision Records in Action . *SATURN*.

Keeling, M., Runde, J., & Gala, C. J. (2018). Architectural Hoisting: Or How I Learned to Stop Writing Breaking Code and Love the Architecture. *SATURN*.

Kniberg, H. (2014, 03 27). *Spotify engineering culture (part 1)*. Retrieved from https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/
Kruchten , P. (2000). *The Rational Unified Process: An Introduction, Second Edition.* Boston, MA: Addison-Wesley Longman Publishing Co., Inc.
Kubernetes Governance Model. (2019, 04 13). Retrieved from https://github.com/kubernetes/community/blob/master/governance.md
Leffingwell, D., Knaster, R., Oren, I., & Jemilo, D. (2018). *SAFe reference guide: Scaled agile framework for lean enterprises.* Boulder, CO: Addison-Wesley Professional.
Nygard, M. (2011, 11 15). *Documenting Architecture Decisions*. Retrieved from http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions
Robert III, H. M., Honemannn, D. H., Balch, T. J., Seabold, D. E., & Gerber, S. (2011). *Robert's Rules of Order Newly Revised.* PublicAffairs.
The Agile Manifesto. (n.d.). Retrieved from https://agilemanifesto.org/

Appendix I: Bylaws

Article I. **Operational Principles**

1. All significant architectural decisions will be recorded as an architectural decision record (ADR).
2. All code committed to any… organization repository master branch will have either gone through code review and been approved or have been written during a mobbing session.
3. All significant design and/or architecture decisions are arrived at through a consensus that includes domain experts and/or component architects.
4. All members who routinely contribute production code support that code by going on call.
5. All support work will follow [our] support policy; see *Article III*.

Article II. **Best Practices**

1. All functionality will be covered by automated verification tests that run after every deployment. Exceptions to this rule are either arrived at through group consensus or are clearly marked as technical debt."
2. All significant new functionality will be performance and/or load tested under conditions that mimic customer usage as realistically as possible before being released for general availability (GA).
3. All deployable components will be characterized in their READMEs as targeting specific quality attributes (QAs).
4. The state of deployed… services will be defined in code ("code as configuration").
5. Cases that do not follow the above are marked as technical debt.
6. All significant new feature development will go through example mapping before implementation work begins.

Article III. **Support Process**

Not reproduced for reasons of confidentiality.

Article IV. **Tactical Squad**

Not reproduced for reasons of confidentiality.

Article V. **Altering Bylaws**

1. Any part of these practices, principles, and processes are subject to modification at a team retrospective consisting of at least half the team, at a team meeting called to address a team-wide problem, and at emergency meetings.

APPENDIX II: OPEN-ENDED SURVEY RESPONSES

1. *"The documents within the repo were helpful to point to during activities while new members were coming up to speed. But the… repo itself didn't help us come up with a process for doing the actual onboarding…."*
2. *"There wasn't really a formal onboarding process for me. It was pretty unofficial, but worked out fine."*
3. *"[The repo] is never reviewed and can easily go out of date. A fraction of the team is listed on the expertise page and that is not due to lack of expertise, but as result of not reviewing the contents as a team occasionally. It does not help at all for technical setup, information about the [team-owned] components, or where to find the repos."*
4. *"Some Suggestions for Improving: - Better [team] education - unified content that actively gets updated as changes happen - Need to do a better job at keeping the docs/READMEs up to date…."*