The Issue of Monorepo and Polyrepo In Large Enterprises

Nicolas Brousse
brousse@adobe.com
Adobe
Emeryville, California

ABSTRACT

Product and engineering teams' speed of producing high-quality results is critical to ensuring enterprise competitiveness. Additionally, one can observe an increase in IT systems complexity driven by the adoption of service-oriented architecture, micro-services, and serverless. Therefore, many large enterprises benefit from a mono-repository for source code management because of the improved team cognition that results from eroding barriers between teams and from influencing enhanced teamwork quality. This paper, first, reviews the characteristics of a multi-repositories structure, a mono-repository structure, and a hybrid model. Second, it discusses why some manage source code in a multi-repositories structure, either by choice or because of the organic evolution of large enterprises. Third, it reviews how mono-repositories in large teams, beyond the technical arguments, can drive high efficiency and enhanced product quality through improved team cognition.

CCS CONCEPTS

• **Software and its engineering → Software libraries and repositories**; **Programming teams**; *Distributed systems organizing principles*; *Rapid application development*.

KEYWORDS

programmer experience, source code management, mono-repository, multi-repositories, enterprise, devops

ACM Reference Format:
Nicolas Brousse. 2019. The Issue of Monorepo and Polyrepo In Large Enterprises. In *Companion of the 3rd International Conference on Art, Science, and Engineering of Programming (Programming '19), April 1–4, 2019, Genova, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3328433.3328435

Programming '19, April 1–4, 2019, Genova, Italy

ACM ISBN 978-1-4503-6257-3/19/04…$15.00
https://doi.org/10.1145/3328433.3328435

1 THE ISSUE OF MONOREPO AND POLYREPO

Information technology (IT) systems are becoming more complex to the extent that machine learning has become necessary to manage that complexity [5, 37]. The rapid adoption of microservices has increased the complexity of systems [15, 39]. If the complexity of modern systems requires more advanced technology to manage them, then one may wonder how this increased complexity impacts organizations.

Software design patterns and architectural patterns are *general, reusable solutions to a commonly occurring problems* [48, 50]. Unfortunately, there remains lack of such patterns to address the challenges of managing source control repositories for large teams or projects. Version Control Software (VCS) has existed for decades [49]. Nevertheless, the practice of using certain software development strategy with a specific repository structure has been only recently qualified with names such as monorepo [19], polyrepo (also named manyrepo or multirepo), and more recently metarepo [30, 41]. The different repositories structure indicated in table 1 present an example of the notable adopters for each design.

There is an insufficient number of academic research on the topic of monorepo [4] while there is an abundance of online material comparing the different structures [24, 36, 42, 43] or concerning the benefit of monorepo [33]. It has become more noticeable since a series of public posts from companies such as Google [38] or Facebook [17] and their unique use of a monorepo. There is now a website dedicated to the adoption of monorepo [16] as well as a public curated list of monorepo tools, software and architectures [7].

Despite the widespread adoption of monorepo structures in large enterprises, the struggle is salient [31], as some went back-and-forth, such as Uber [29, 32]. A case study at Google [25] concluded that the choice regarding the use of a monorepo or polyrepo architecture involves a *comparison of tradeoffs* while still highlighting that monorepos *"encourages consistent and high-quality code, and empowers engineers*

Table 1: Source Code Repository Structures

Structure	Implementation	Notable Adopters
Multi-repositories (polyrepo)	One source repository for each component and library	Amazon, Netflix, Lyft
Mono-repository (monorepo)	One source repository for a whole company or very large product line	Google, Facebook, Microsoft, Uber, Twitter, React, Angular, Babel, Kubernetes
Hybrid Poly-as-Mono	Updates are made to multiple repository but managed like a monorepo	Android, Chrome
Hybrid Mono-as-Poly	Updates are made into a monorepo but then split into read-only polyrepo for build or distribution purpose	Symfony, Shopsys

to study and learn from the institutional knowledge of their company, crystallized in the form of source code".

2 MONOREPO AND CULTURE IN LARGE ENTERPRISES

The technical benefits of monorepos may partially be a fallacy, since some of the problems associated with polyrepos still must be resolved in some manner in a monorepo [28]. Additionally, one cannot ignore the large adopters of a polyrepo models such as Amazon and Netflix. The complexity of the technical choice and the lack of clarity in the tradeoff causes some to alternate back and forth, as in Uber's case. Ironically, Lyft also adopts a mixed approach, with a strong polyrepos approach on most of its systems, but apparently a monorepo model at the Android project level [51]. If the technical benefits are not obvious and vary in terms of many external factors, then, what makes monorepos highly successful in most large enterprises or projects? It is clear that the benefits of either implementation appear to depend upon other factors aside from a pure technical choice.

A key element in the initial Google publication [38] is the mention of the word *culture*. The four occurrences emphasize on the open collaboration and how it encourages code quality. This focus on quality is expected, since part of Google's philosophy claims that *"It's best to do one thing really, really well"* and *"Great just isn't good enough"* [18]. Twitter, who also adopted a monorepo structure, claims on their career page *"We Are One Team"* [45].

The relation between culture and choice of repository structure is not limited to monorepo. Netflix, who adopted polyrepo, favors a culture of *"freedom and responsibility"* [22] that *"empowers engineers to craft solutions using whatever tools they feels are best suited to the tasks"* [35]. However, this would need to be balanced against Netflix's "keeper test" and culture of fear [46]. Another polyrepo example is provided by Amazon, who is also known for its unique culture and for making teams working on the same projects and compete with each other [1, 26, 44].

Finally, one can examine Microsoft, which has transitioned from polyrepo to monorepo. Satya Nadella, Microsoft CEO, has transformed the company culture, in his words *"we needed a culture that allowed us to constantly refresh and renew"* [34]. The culture change is accompanied by a change in the choice of source code repository structure. Microsoft scaled Git [21] to handle the largest Git monorepo in the world [20]. Ultimately, Microsoft observed that the polyrepo structure fostered a siloed culture and that transitioning to monorepo allowed them to provide a better developer experience [9].

3 MONOREPO FOSTER TEAM COGNITION

Functional silos have been identified for a long time as an inhibitor of team productivity, velocity, and innovation [12, 13]. It is critical to enable permissionless innovation, which is *"the ability of others to create new things on top of the communications constructs that we create"* [2]. Permissionless innovation enables autonomy while maintaining accountability. Autonomy does not imply independence, but rather that systems and people work well within their environment (being self-directed as opposed to not dependent upon anything) [8, 10, 40]. Large enterprises have long sought to reduce team frictions and reduce their silos in order to increase their competitiveness. Large enterprises tend to have many mergers and acquisitions activities as part of their regular operations, which require them to employ certain consolidation efforts to minimize the risk of increased numbers of silos.

The experience reports from Microsoft, Google, and other large companies confirms that team cognition is an important factor in the choice of repository structure with an effort to gear toward a holistic team cognition rather than collective team cognition [27]. How we communicate and share information exerts a direct impact on a team cognition, which is a *"critical mechanism for facilitating knowledge activities within the software development process"* [23]. It has been demonstrated that *high quality software depends on high quality collaboration* within a team, with influencing factors such as trust, value-sharing, and coordination of expertise [47].

To accomplish holistic team cognition, it becomes necessary to *tear down the silos* between teams. This helps produce a shared vision, facilitate communication and collaboration. Ultimately, those efforts align with the recent movement around DevOps methodologies [14]. The 2018 DevOps report highlights the benefit of *"influencing organizational culture through autonomy, leadership and learning"* [11], with Google being cited as yet another example of how their environment provides psychological safety and a sense of safety when taking risks. The report also establishes that one can change a company's culture by changing how work is performed. At this point, one can assert that enterprise can leverage the *Inverse Conway Maneuver* [3, 6] by designing a workflow based on a monorepo to drive critical culture changes that break down silos and enable competitiveness.

4 CONCLUSION

This paper has identified different source code repository structures and well-known adopters. Ultimately, it has established that the technical arguments for one model or another always correspond with certain tradeoffs and may not be a clear-cut. However, it has observed that the benefit of monorepo is the cultural impact. It has asserted that a monorepo facilitates cultural change and enables a holistic team cognition that ensures high quality work and improves communication, while preserving necessary autonomy. These are critical elements to help a large enterprise remain competitive and deliver innovation and high quality products rapidly. Ultimately, more research is necessary on source control repository structures and their direct impacts on team cognitions. The industry would benefit from defined patterns that establish the technical tradeoff and culture impact of each model.

ACKNOWLEDGMENTS

The author would like to thank the site reliability engineering and software engineering teams from the Adobe Advertising Cloud for their help and contribution to this work. The author would also like to express their deepest gratitude to Julie Lee for her insightful comments and suggestions.

REFERENCES

[1] Amazon. 2019. Leadership Principles. Retrieved Jan 25, 2019 from https://www.amazon.jobs/en/principles

[2] Jari Arkko. 2013. Permissionless Innovation. Retrieved Jan 25, 2019 from https://www.ietf.org/blog/permissionless-innovation/

[3] Jason Bloomberg. 2015. DevOps Insights into Conway's Law. Retrieved Jan 25, 2019 from https://intellyx.com/2015/06/22/devops-insights-into-conways-law/

[4] Gleison Brito, Ricardo Mingarini Terra, and Marco Tulio Valente. 2018. Monorepos: A Multivocal Literature Review. *CoRR* abs/1810.09477 (2018). http://arxiv.org/abs/1810.09477

[5] Nicolas Brousse. 2018. Top 5 Machine Learning and Self-Healing Techniques used by SRE. Retrieved Jan 25, 2019 from https://www.linkedin.com/pulse/top-5-machine-learning-self-healing-techniques-used-sre-brousse

[6] Melvin E. Conway. 1968. How Do Committees Invent? *Datamation* (April 1968). http://www.melconway.com/research/committees.html

[7] Uriel Corfa. 2019. Awesome Monorepo. Retrieved Jan 25, 2019 from https://github.com/korfuri/awesome-monorepo

[8] Edward L. Deci and Richard M. Ryan. 1985. Self-Determination Theory. Retrieved Jan 25, 2019 from http://selfdeterminationtheory.org/theory/

[9] Scott Densmore. 2018. How "Mono-repo" and "One Infra" Help Us Deliver a Better Developer Experience. Retrieved Jan 25, 2019 from https://blogs.msdn.microsoft.com/vsappcenter/how-mono-repo-and-one-infra-help-us-deliver-a-better-developer-experience/

[10] DifferenceBetween.com. 2015. Difference Between Autonomy and Independence. Retrieved Jan 25, 2019 from https://www.differencebetween.com/difference-between-autonomy-and-vs-independence/

[11] DORA. 2018. *Accelerate: State of DevOps Strategies for a New Economy.* Technical Report. DORA.

[12] Phil Ensor. 1988. The functional silo syndrome. *AME Target* 16, Spring Issue (1988), 16.

[13] Phil Ensor. 1988. Organizational Renewal-Tearing Down the Functional Silos. In *AME Study Group on Functional Organization.* AME Target, 4–16.

[14] Nicole Forsgren, Jez Humble, and Gene Kim. 2018. *Accelerate: The Science of Lean Software and DevOps Building and Scaling High Performing Technology Organizations* (1st ed.). IT Revolution Press.

[15] Martin Fowler. 2015. MicroservicePremium. Retrieved Jan 25, 2019 from https://martinfowler.com/bliki/MicroservicePremium.html

[16] gomonorepo.org. 2019. All You Always Wanted to Know About Monorepo But Were Afraid to Ask. Retrieved Jan 25, 2019 from https://gomonorepo.org

[17] Durham Goode. 2014. Scaling Mercurial at Facebook. Retrieved Jan 25, 2019 from https://code.fb.com/core-data/scaling-mercurial-at-facebook/

[18] Google. 2019. Google Philosophy. Retrieved Jan 25, 2019 from https://www.google.com/about/philosophy.html

[19] Paul Hammant. 2018. Trunk Based Development: Monorepos. Retrieved Jan 25, 2019 from https://trunkbaseddevelopment.com/monorepos/

[20] Brian Harrys. 2017. The largest Git repo on the planet. Retrieved Jan 25, 2019 from https://blogs.msdn.microsoft.com/bharry/2017/05/24/the-largest-git-repo-on-the-planet/

[21] Brian Harrys. 2017. Scaling Git (and some back story). Retrieved Jan 25, 2019 from https://blogs.msdn.microsoft.com/bharry/2017/02/03/scaling-git-and-some-back-story/

[22] Reed Hastings. 2009. Netflix Culture: Freedom & Responsibility. Retrieved Jan 25, 2019 from https://www.slideshare.net/reed2001/culture-1798664/2-NetflixCultureFreedomResponsibility2

[23] Jun He, Brian S. Butler, and William R. King. 2007. Team Cognition: Development and Evolution in Software Project Teams. *J. of Management Information Systems* 24 (2007), 261–292.

[24] Joel Parker Henderson. 2019. Monorepo vs. polyrepo. Retrieved Jan 25, 2019 from https://github.com/joelparkerhenderson/monorepo_vs_polyrepo

[25] Ciera Jaspan, Matthew Jorde, Andrea Knight, Caitlin Sadowski, Edward K. Smith, Collin Winter, and Emerson Murphy-Hill. 2018. Advantages and Disadvantages of a Monolithic Repository: A Case Study at Google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP*

'18). ACM, New York, NY, USA, 225–234. https://doi.org/10.1145/3183519.3183550

[26] Jodi Kantor and David Streitfeld. 2015. Inside Amazon: Wrestling Big Ideas in a Bruising Workplace. Retrieved Jan 25, 2019 from https://www.nytimes.com/2015/08/16/technology/inside-amazon-wrestling-big-ideas-in-a-bruising-workplace.html

[27] Preston Kiekel and Nancy Cooke. 2005. HUMAN FACTORS ASPECTS OF TEAM COGNITION. *The Handbook of Human Factors in Web Design* (01 2005).

[28] Matt Klein. 2019. Monorepos: Please don't! Retrieved Jan 25, 2019 from https://medium.com/@mattklein123/monorepos-please-dont-e9a279be011b

[29] Gautam Korlam. 2017. One for all, all for one - The Journey to Android Monorepo at Uber. Retrieved Jan 25, 2019 from https://speakerdeck.com/kageiit/one-for-all-all-for-one-the-journey-to-android-monorepo-at-uber

[30] Burke Libbey. 2017. Monorepo, Manyrepo, Metarepo. Retrieved Jan 25, 2019 from http://notes.burke.libbey.me/metarepo/

[31] Cheng Long. 2017. Multirepo vs Monorepo. Retrieved Jan 25, 2019 from https://chengl.com/multirepo-vs-monorepo/

[32] Aimee Lucido. 2017. Mono-repo To Multi-repo And Back Again. Retrieved Jan 25, 2019 from https://www.youtube.com/watch?v=lV8-1S28ycM

[33] Dan Luu. 2015. Advantages of monorepos. Retrieved Jan 25, 2019 from https://danluu.com/monorepo/

[34] Harry McCracken. 2018. Transforming culture at Microsoft: Satya Nadella sets a new tone. Retrieved Jan 25, 2019 from https://www.intheblack.com/articles/2018/06/01/satya-nadella-transforming-culture-microsoft

[35] Netflix. 2016. How We Build Code at Netflix. Retrieved Jan 25, 2019 from https://medium.com/netflix-techblog/how-we-build-code-at-netflix-c5d9bd727f15

[36] Mike Nikles. 2017. One vs. many - Why we moved from multiple git repos to a monorepo and how we set it up. Retrieved Jan 25, 2019 from https://hackernoon.com/one-vs-many-why-we-moved-from-multiple-git-repos-to-a-monorepo-and-how-we-set-it-up-f4abb0cfe469

[37] Seth Paskin. 2018. What is AIOps? Artificial Intelligence for IT Operations Explained. Retrieved Jan 25, 2019 from https://www.bmc.com/blogs/what-is-aiops/

[38] Rachel Potvin and Josh Levenberg. 2016. Why Google Stores Billions of Lines of Code in a Single Repository. *Commun. ACM* 59, 7 (June 2016), 78–87. https://doi.org/10.1145/2854146

[39] redhat. 2018. Microservices: What's a service mesh? Retrieved Jan 25, 2019 from https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh

[40] Dave Schools. 2016. Autonomy and Accountability: How Leaders Achieve Miracles With Their Teams. Retrieved Jan 25, 2019 from https://www.viget.com/articles/autonomy-and-accountability-how-ceos-achieve-miracles-with-their-teams/

[41] Patrick Lee Scott. 2017. Mono-repo or multi-repo? Why choose one, when you can have both? Retrieved Jan 25, 2019 from https://medium.com/@patrickleet/mono-repo-or-multi-repo-why-choose-one-when-you-can-have-both-e9c77bd0c668

[42] Peter Seibel. 2017. REPO STYLE WARS: MONO VS MULTI. Retrieved Jan 25, 2019 from http://www.gigamonkeys.com/mono-vs-multi/

[43] Chetan Sharma. 2018. How to Structure Code Repositories: Multi, Mono or Organic? Retrieved Jan 25, 2019 from https://blog.socialcops.com/technology/engineering/code-repositories-multi-mono-repo/

[44] Brad Stone. 2013. *The Everything Store: Jeff Bezos and the Age of Amazon.* Atlantic/Little, Brown, Boston, MA, USA.

[45] Twitter. 2019. Twitter Career. Retrieved Jan 25, 2019 from https://careers.twitter.com/en.html

[46] The Week. 2018. Netflix's culture of fear. Retrieved Jan 25, 2019 from https://theweek.com/articles/805123/netflixs-culture-fear

[47] Emily Weimar, Ariadi Nugroho, Joost Visser, Aske Plaat, Martijn Goudbeek, and Alexander P. Schouten. 2013. The Influence of Teamwork Quality on Software Team Performance. *CoRR* abs/1701.06146 (2013).

[48] Wikipedia. 2019. Comparison of version-control software. Retrieved Jan 25, 2019 from https://en.wikipedia.org/wiki/Design$_{p}$attern

[49] Wikipedia. 2019. Comparison of version-control software. Retrieved Jan 25, 2019 from https://en.wikipedia.org/wiki/Comparison$_{o}$f$_{v}$ersion-control$_{s}$oftware

[50] Wikipedia. 2019. Design pattern. Retrieved Jan 25, 2019 from https://en.wikipedia.org/wiki/Architectural$_{p}$attern

[51] Artem Zinnatullin. 2018. Android builds at Lyft. Retrieved Jan 25, 2019 from https://mobiusconf.com/en/2018/msk/talks/2dbsfjgmi8ecuec8kwmqwi/