Technical Debt Through Silos

Anthony Escalona
City of New York - DoITT
2 Metro Tech 5th Floor
Brooklyn, NY, U.S.A 11201
aescalona@doitt.nyc.gov

***Abstract*—This paper explores organizational change management (OCM) as an integral part of DevOps adoption to break silos and remove boundaries between business, development, QA, and operations groups. It attempts to depict the characteristics of DevOps. The paper describes how DevOps can be applied to software delivery. It also seeks to analyze and explore its interplay with a view on several perspectives. These include: Conway's Law, Theory of Constraints, and Thinking Processes.**

I. Introduction and Motivation

I understand that the organization is not yet aligned to provide end-to-end agile. However, during my stint here, I feel we have as a whole reached an impasse where much of our challenges are not technical but rather cultural and operational.

In organizations, such as DoITT, that are not already steeped in DevOps, the practical differences in the way the operations and development groups approach their work can be a source of tension. It will be difficult to architect an agile infrastructure platform like Continuous Delivery (CD) without aligning goals and objectives and embracing the same cultural idioms that have enabled agile software development in private industries. There are core values and concepts that must change to shape organizational efforts to succeed. Insufficient consideration of these cultural elements will make automation (improve operational effectiveness) or modernization initiatives more difficult, if not doomed to fail. Cohesion is necessary to create synergy in the platforms we build. There must be a connection among the individual functional groups such that the output of the whole is greater than the output of each individual function. In other words, the interaction of elements that when combined produce a total effect that is greater than the sum of the individual elements. Here, Development and Operations are part of different hierarchal structures within an organization. They both have different visions, missions, and SLAs metrics, making it quite challenging to bring together people, processes, and technologies but both share the same goals.

This paper attempts to explore Devops. It will further explore the combined role of individuals and technology. It will as a matter of core significance explain its characteristics, environment, conditions, and practice. Is it possible to replace people with robots like chatbots?

II. Conway's Law

Among computer science literature, Melvin Conway has published many technical papers over the past few decades. Most notable to the Empirical Software Engineering community is his 1968 article titled "How Do Committees Invent?", which presents a statement on the sociology of system design. He describes how the communication paths between functional groups inevitably influence the final product design. He codified it to what now become known as Conway's Law:

> *Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.*

As Conway notes, when technologists break down problems into smaller blocks to delegate, they introduce coordination problems. In many ways, formal communication structures or rigid hierarchy appear to solve this coordination problem but often lead to inflexible solutions. In organizations with functional silos, management divides teams to make their Human Resources department happy without much regard to engineering efficiency [1]. Although each group may be good at their part of the design (e.g., ops, app dev, or network), to release a new capability or feature, all three teams must be involved in building the capability. Organizations typically optimize for efficiency for their immediate tasks rather than the more abstract, strategic goals of the agency, particularly when under schedule pressure. Infrastructure development is a collaborative process reflecting the ideas of the people involving communication structures across different organizations

III. Silos

The core idea of DevOps is to break silos and remove boundaries between business, development, and operations teams. With decentralized, self-organizing groups, there is a tendency to create a centralized center of excellence to establish standards and guidelines that leverage best-practices and mitigate the risk of inconsistency when using various methods and tools[2]. DevOps and removal of departmental silos extend CD through continuous integration and promotes consistent, reliable and automated capabilities with frequent iterations and fast feedback loops. CD tools provide visibility to operational environments, streamline their workflows and automate some build, release and deployment steps. In speaking with some of the folks, I conclude we are all incongruent in eliminating silos to improve agility, reliability, testability, repeatability, and sustainability, etc. However, efforts might be marginalized or stymied due to the lack of inclusion and support because of the many protected silos. For individual contributors, breaking these barriers are ways above their regular stratum. I'm fearful the lack of cross-functional communications will impact or impede our ability to solve architectural and engineering solutions methodologically thus incurring additional technical debt for the entire organization.

IV. DevOps

Specifically, in the context of Infrastructure and Operations, DevOps addresses coordination by underpinning the three widely accepted principles:

1) System Thinking and Flow:
   a) Create fast flow of work as it moves from Development to IT Operations (e.g. 1:1 parity between development, test, and production environments)
2) Feedback Loops:
   a) Demonstrate the ability to shorten and amplify the feedback loops so issues can be fixed at the source and avoid rework (e.g. Fitness Functions and Telemetry).
3) Experimentation and Learning
   a) Somehow create a culture that simultaneously fosters experimentation, learning from failure, and the importance of repetition and practice are needed.

DevOps practices address all of these coordination challenges by establishing collaborative cross-functional development, test, and operational teams that share their responsibility for maintaining the overall system running the software and further prepares the software to run on a specific system with greatly increased quality feedback as well as automation. This engineering efficiency will help with automating key build processes, deployment, and management tasks to speed up the deployment process. DevOps is a client-centric approach enabling rapid delivery of capabilities to them, which requires collaboration across the software delivery value stream that includes teams spanning across business, development, qa, and operations. DevOps brings together all these groups, thereby making process of development very lean.

V. Theory of Constraints

One of the difficulties in shifting organizations towards a more systemic model like DevOps is that constraints can impede one's thinking. A paper presented by Pegels and Watrous [3] reflects a case study of the successful application of the theory of constraints (TOC), a productivity improvement tool proposed and developed by Goldratt and Fox [4]. TOC was in the forefront at that time and continues to be actively used in industry because of its considerable potential to (1) identify throughput problems, (2) serve as a guide to correct the throughput problems, and (3) generate significant improvements in productivity and efficiency. TOC views processes within organizations as "chains", wherein the entire system is only as strong as its weakest link. The purpose of TOC is to identify the weak link (constraint) within an organization and to strengthen this link to the point where it is no longer the limiting factor in determining the strength of the chain organization. TOC is a continuous improvement process where a system is viewed as a flow, and the objective of the process is to strengthen it by identifying and eliminating all bottlenecks continually.

The TOC includes a sophisticated problem-solving methodology called the Thinking Processes. The Thinking Processes are optimized for complex systems with many interdependencies (e.g. manufacturing lines). They are designed as scientific "cause and effect" tools, which strive first to identify the causes of undesirable effects (UDEs) and then remove the UDEs without creating new ones.

Just understanding that adopting DevOps practices and expanding capacity upstream to improve system flow will not necessarily increase the overall capacity of the system unless you identify and increase capacity in the bottlenecks identified through the TOC. If the weakest link is the speed of delivery of new functionality or the systems' operational behavior, then DevOps has considerable benefits to offer. As TOC continued to evolve, it became clear that a limiting factor to the organization's performance is not always a bottleneck resource, but can be of different nature, like cultures, policies, decisions, etc.

VI. Coordination

In software engineering, for example, there are tasks or operations that are time-critical such as code development. Usually the problem to be resolved is directly linked to a long trail of related and successive tasks. The incidents show that a problem with one of these tasks will result in a standstill of the entire development process. The delay in waiting for a resolution of the problem consequently translates to a loss of time and waste. In order to better illustrate the problems to be addressed, the case study, DoITT was chosen as an example. The choice of DoITT, as a basis, for the assessment of coordination is based on the following: (1) DoITT provides efficient and effective delivery of IT services, and infrastructure. (2) It utilizes the of people and technology. (3) It is going through a massive modernizing efforts by implementing state-of-the-art information technology to improve services to its agencies. Given the enumerated reasons, DoITT therefore serves as a logically ideal ground for such discourse.

VII. Evolution

Many opportunities exist to modernize some of the coordination (i.e., intake processes) challenges by taking a leaner approach by identifying bottlenecks of system flows. For instance, constant continuous improvements of DevOps practices have further solved some of the coordination problems locally by replacing snowflakes with Immutable Infrastructure as Code (IIAS). Snowflake servers are ones that have been manually crafted by an operations person, and all future maintenance is done by hand [5]. As a result, making it hard to reproduce and replace, making things like scaling and recovering from issues difficult. IIAS refers to systems defined entirely programmatically. All changes to the system must occur via the source code, not by modifying the running operating system. Thus, the entire system is immutable from an operational standpoint — once the system is bootstrapped, no other changes occur. Importantly, IIAS remove needless variables thus eliminating many unknown factors and complexities as possible. Snowflake servers would be virtually impossible to build [6] [7] fitness functions [8] that can provide metrics and telemetry how the latest patch of the operating system might affect the application. Fitness functions are used to provide feedback how close a solution is to achieving the intended design. They encompass a variety of techniques to ensure the architecture does not change in undesirable ways, including metrics, tests, and other verification tools [1].

VIII. Agile

A move closer to agile adoption might help where parallel execution amongst all silos in every project is essential. It will eliminate the inherent problems that stem from each stage of infrastructure release life-cycle. With many back-and-forth exchanges, incomplete or misinterpreted information moving up and down the chain, and Project Management or lack thereof trying to keep everyone coordinated, there are too many hand-offs, ticketing, and linear processes causing the challenges and delays we are seeking to reduce, and ultimately eliminate, with Agile. Parallel execution can significantly diminish the dissipation of effort that comes with the lack of coordinated communication across teams. In practice, this will mean that each functional group involved in a particular release will be working off a common backlog, and should adopt the same Agile processes. There is enough context (not control) here to have a charter that sets expectations, quality, metrics, and vision around a capability or feature. For example, continuous delivery gives us context through automation. If expectations are through automation (process as code), then people have freedom within that context to accomplish great things through experimentations. And when things do fail which often they do, then we asked ourselves what context, fitness functions, telemetry, or processes have we failed to set? Without putting any blame on the individuals or divisions; we just automatically make those corrections through code.

IX. DevOps AI

This paper presented boundaries between business, development, and operations groups involving software delivery. DoITT has extensive amounts of organizational silos across various departments and divisions. Even with Agile adoption, coordination will still result in slow processing time of software delivery. Furthermore, the evolution of immutable infrastructures have exacerbated delivery through delays and missed deadlines. Generating chat bots using AI small to represent different organization groups to improve operational coordination and efficiency is worth exploring. Business constraints, organizational constraints, and other coordinational constraints are potentially solvable through AI automation. Lee[9], in their paper, have identified problems of scalability under the explosive growth of Internet traffic and high-speed access. They presented a Hadoop-based traffic monitoring system that performs analysis of the network traffic in a scalable manner. Similarly, how can AI help with our constraint problems?

X. Summary

This short paper represents an attempt at exploring DevOps and improving cultures. It highlights the study of Conway's with the aim of shedding light on how process and structures are formed.

Many innovative experimentations are still needed to bring these AI capabilities forward. It is best achieved by eliminating boundaries, bottlenecks, and improving coordination in how we work. To this end, I propose an AI chatbot to speed up operational processes and lower constraints by using machine learning to improve operational decisions.

XI. Future Work

Bots are relatively new comers and more research is needed to determine the efficacy on using AI or machine learning to interact with humans through bots to reduce organizational silos.

Acknowledgment

My sincerest gratitude to Eric Ip and Kim Truong for reviewing the paper.

References

[1] N. Ford, R. Parsons, and P. Kua, *Building Evolutionary Architectures: Support Constant Change*. Sebastopol, CA: O'Reilly Media, Inc, 2017, iSBN-13: 978-1491986363.

[2] G. Shrikanth, "Deepdive: Devops and its impact on software development," Online, 2017, https://fit.summon.serialssolutions.com, last access October 2017.

[3] C. Pegels and C. Watrous, "Application of the theory of constraints to a bottleneck operation in a manufacturing plant," Online, 2005, https://doi.org/10.1108/17410380510583617, last access October 2017.

[4] E. Goldratt and J. Cox, *The Goal: A Process of Ongoing Improvement*. New York, NY: Routledge, 1986, iSBN-13: 978-0884271956.

[5] H. Virdo, "What is immutable infrastructure," Online, 2017, https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure, last access October 2017.

[6] N. Li, A. Escalona, and T. Kamal, "Skyfire: Model-based testing with cucumber," in *Software Testing, Verification and Validation (ICST), 2016 IEEE International Conference*, ser. ICST'17, Chicago, IL, April 2017.

[7] N. Li, M. West, A. Escalona, and V. Durelli, "Mutation testing in practice using ruby," in *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference*, ser. ICST'15, Graz, Austria, April 2017.

[8] M. Harmon and J. Clark, "Metrics are fitness functions too," in *Software Metrics, 2004. Proceedings. 10th International Symposium on*, 2004.

[9] Y. Lee and Y. Lee, "Toward scalable internet traffic measurement and analysis with Hadoop," ACM SIGCOMM Computer Communication Review, vol. 43, (1), pp. 5, 2012. . DOI: 10.1145/2427036.2427038.