Applying DevOps for Distributed Agile Development: A Case Study

Asif Qumer Gill and Devesh Maheshwari

University of Technology Sydney, Ultimo, NSW 2007, Australia
e-mail: asif.gill@uts.edu.au

Abstract Agile software engineering principles and practices have been widely adopted by the software-intensive organizations. There is an increasing interest among organizations in adopting DevOps for improving their distributed agile software environments. However, the challenge is how best to adopt and integrate DevOps in their software development environments – especially in distributed agile environment. This paper presents one such successful case study of DevOps adoption by the distributed agile teams for the development and deployment of a real-time high-performance gaming platform. (1) Small teams, (2) trust, (3) active communication and collaboration culture, (4) shared product vision and roadmap, (5) continuous feedback and learning culture, (6) appreciation and excellent senior management support are some of the key success factors of DevOps. The experiences and learnings discussed in this paper can be used by other organizations to effectively plan and adopt DevOps for their environment.

Introduction

Agile approaches have fundamentally changed the way organizations develop and deploy software [17]. Agile approaches focus on iterative development and continuous improvement [1]. Agile development team aims to incrementally deliver the working software to operations team [18]. The operations team is then responsible for putting the software into the production environment. Despite the recent success with agile development, operations at large still work in isolation and slow compared to agile development teams [2]. Being agile in development, and not agile in operations, is one of the major concerns of agile teams. Lack of alignment and synchronization between the development and operations could lead to the problem of slow release and longer time to market of software solutions [3]. Isolated and slow operations in traditional settings could be collectively seen as a bottleneck in the overall value stream of software delivery [4]. In order to address this important concern, an alternative and integrated DevOps (development and operations) approach is emerging and getting vast attention from software-intensive organizations [5].

DevOps seems to be an interesting approach. However, its adoption in distributed agile software development is a challenging task [15, 19, 21]. This paper aims

to address this important concern and presents a case study of DevOps adoption in distributed agile development environment. This study provides interesting insights and key success factors of DevOps adoption such as (1) small teams, (2) trust, (3) active communication and collaboration culture, (4) shared product vision and roadmap, (5) continuous feedback and learning culture, (6) appreciation and excellent senior management support.

This chapter is organised as follows. Firstly, it describes the DevOps concepts. Secondly, it presents the DevOps case study results. Finally, it presents the discussion and conclusion.

DevOps

DevOps is defined as "a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality" [4, 16]. The integrated DevOps brings together both the development and operations teams and seems to address the bottleneck of slow releases of software into production environment [20, 22]. The integration of DevOps is not a straightforward task and poses several technical and non-technical challenges [6]. This paper presents a case study of DevOps adoption in an Australian software-intensive organization (ABC - coded name due to privacy concerns) for the distributed agile development and deployment of a real-time high-performance gaming platform. The experiences and learnings from this case study will help the readers to understand the DevOps process, its implementation, and key learnings.

DevOps Case Study

The ABC is an ASX listed entertainment organization. It offers gaming software solutions in Australia. Its vision is to create entertainment experiences where the passion, thrills, and enjoyment of the Australian way of life comes alive. It is one of the world's largest publicly listed gaming firms. It runs multiple gaming brands and has the ability to handle huge amount of daily real-time transactions (over million) on a high performance platform and their digital capability allows them to deliver the same at the fast pace by using an agile approach. It has a flat organization structure, which is augmented by continuous feedback and learning culture.

Analytical Lens

ABC has been using DevOps in their distributed agile environment for more than 3 years. ABC DevOps case has been analysed and reported by using the "Iteration" management capability layer (see Figure 1) from the adaptive enterprise project management (AEPM) capability reference model [7]. The AEPM capability reference model specifies the services for adaptive or agile portfolio, program, project, release, and iteration management layers. DevOps is one of the services embedded in the bottom "Iteration" layer of the APEM, hence, it has been deemed appropriate and used here as an analytical lens to systematically analyze the ABC DevOps case.

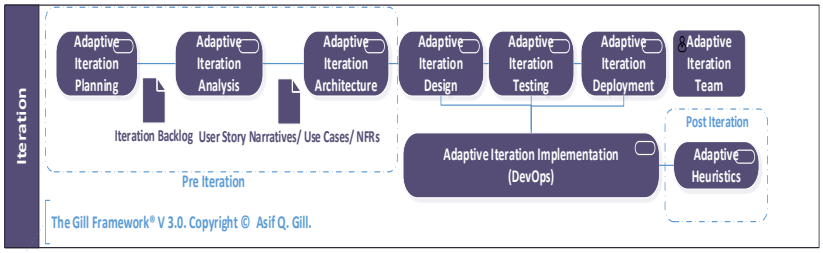

Fig. 1. AEPM – Showing Iteration Management Layer (adapted from [7])

An iteration is a short time-boxed increment of a software. Iteration has embedded services, which are organized into three parts: pre-iteration, iteration implementation and post iteration implementation. Iteration team employs practice and tools to realize these services. Pre-iteration services include adaptive iteration planning, analysis, and architecture for the upcoming iteration. Iteration implementation refers to the integrated development and operations (DevOps) of current iteration in hand. It also involves automated testing and continuous deployment (CD) services [8]. The CD also includes continuous integration services (CI) [9]. The deployment covers the deployments in development, test, staging and production environments. Code is design, which emerges as the DevOps progresses in small increments. However, design service can be used to document the technical design, if required. Heuristics refers to continuous learning or adaptation through iterative feedback and reviews.

Iteration Management

ABC is using an agile release cycle, which involves the development of prioritized product features in two weeks increments. Release cycle includes inception stage, which includes release planning, vision, and scope. Release cycle spans multiple iterations that frequently release working software increments into production. The

ABC release cycle has been analyzed and detailed below using the iteration management capability layer items (see Figure 1).

Iteration Team

ABC has 70+ IT team members working on the gaming platform, which are organised into small geographically distributed feature teams. These teams are located across Sydney, Melbourne, and Brisbane. Each feature team size ranges from 6-9 people. These teams are supported by 3 DevOps engineers. DevOps engineers create standard scripts, lay out foundation for execution and guide teams to move in the right direction. It is important to note here that development teams take ownership to the larger extent to deliver features including DevOps tasks. DevOps engineers mainly facilitate the feature teams to smoothly deploy product increments into production environment. Feature teams continuously deploy code in test environment. However, code is deployed into production twice in a week. Hence, teams delivering features take the ownership and responsibility of taking the code through to production and support it.

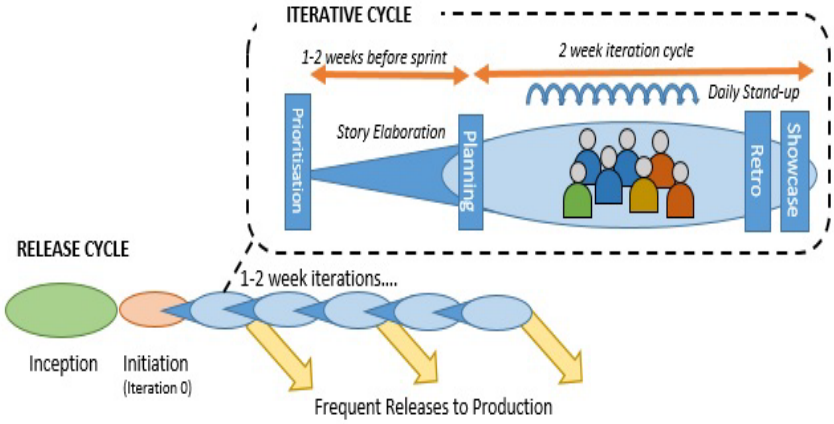

Fig. 2. APEM – ABC Iterative Cycle

Pre-Iteration

The iteration cycle of ABC has iteration 0, which is called the initiation stage. Iteration 0 is also a pre iteration for next iteration (iteration 1). It means that iteration 1 planning, user stories and architecture (story prioritization and elaboration) are detailed in iteration zero (0). Similarly, iteration 2 planning, user stories and architecture are detailed in pre iteration 1. This enables the team to have the user stories ready (analysed, planned and architected) before the start of next iteration.

Iteration Implementation (DevOps)

The bulk of the work is done during iteration implementation. Product increment user stories are implemented by the distributed agile teams using automated DevOps practices and tools. Development is done by using the Microservices Architecture style, in which application is decomposed into small independent fine-grained services in contrast to traditional coarse-grained service [10]. The application Microservices are deployed in the cloud (Amazon Web Services), which is also integrated with the ABC company infrastructure. Automated testing, functional and non-functional, is built into the DevOps process. The development team develops the code and automated tests, which are required to complete the user stories. Any new scenario identified by the team during the iteration implementation is also estimated and prioritized and, if required, is incorporated into the current or upcoming iterations. Code is a design. However, additional technical design, if required, is also done as a part of the user story implementation. The artefacts, other than the code, are captured on the source control wiki for information management and sharing.

Code is frequently checked into the version control system and is also peer reviewed. Once code has been peer reviewed and automated build on CI server is passed, it is merged into the mainline repository. Once the code is checked into the version control, the automated tests are run by the CI server again to verify that the change has not had any adverse impacts on the rest of the solution. This is to ensure the quality and integrity of the solution. There is a high level of automated test coverage. It is made sure that the relevant acceptance criteria have met and execution of the exploratory tests is done for any edge cases. Any identified issues are captured as comments in the story tracking tool for a given story and are fixed straight away by the person who developed the story, and then re-checked by the person verifying the story. It is the mindset of the team that all issues or defects have high priority and need to be fixed as early as possible. This is done to avoid the possibility of hanging over issues and technical debt.

In a nutshell, user stories cannot be deployed or considered 'complete' until they are tested as a part of the automated test suite. User stories, acceptance tests and defects are captured and tracked using the agile tool, which is called Mingle. The CD employs CI to ensure that the code base is always in a deployable state and that regression defects have not been inadvertently introduced. The CI is enabled using the "GO" CI integration server [11], which is responsible for deployment orchestration. It is also supported by the Github repository [12] for version control for both code and test scripts. Confluence (wiki) [13] is used for capturing supporting information that is not recorded in Mingle or Github. Ansible is used for preparing configuration [14]. Further, active communication and collaboration among distributed agile teams are enabled using the HipChat communication tool. Each user has their own login, every change is recorded showing who made the change, and for each check in, the associated Pivotal Tracker ID is referenced. Figure 3 summarizes the DevOps value stream.

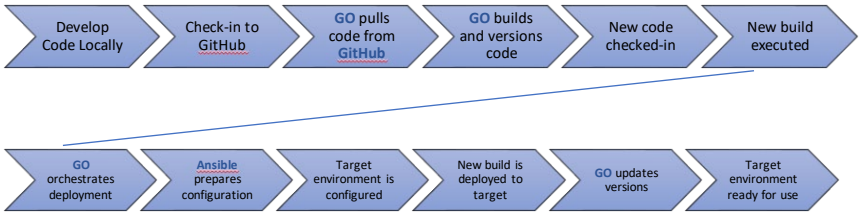

Fig. 3. DevOps Value Stream

The CD of the overall DevOps process involves deployments in five different target environments: local development, shared development, testing, pre-production, and production environments. Local developments are done on the standalone machine or laptop. Shared development environment involves one or more components. Testing environment is for functional testing such as UAT. Pre-production is a production like environment for performance testing and related bug fixing. Finally, production is a customer facing environment, which is duly monitored, operated, and supported by the DevOps team. Deployment pipeline can be traced from Github to Base AMI (Amazon Machine Instance) to Web AMI to Web Deployment (see example in Figure 4).

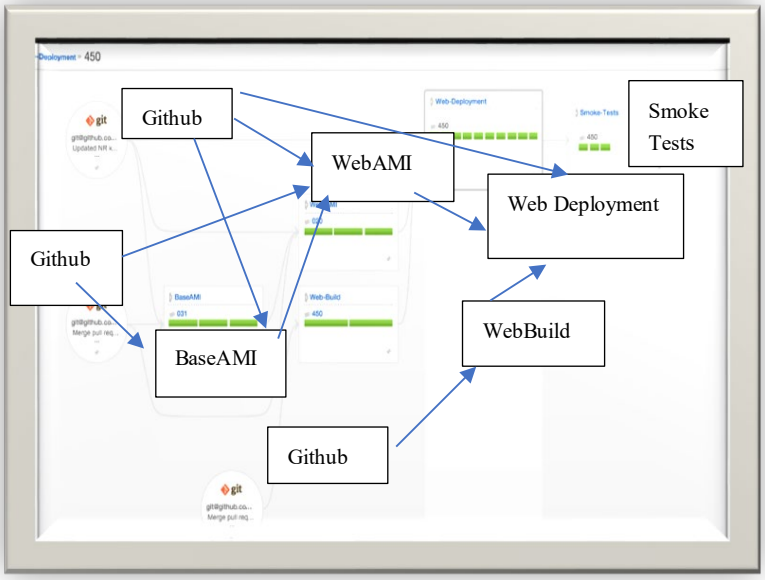

Fig. 4. Deployment Pipeline

One of the goals of the DevOps is to make available the working solution into production environment as quickly as possible. However, the quality of the deployed code or solution is very important from target stakeholders' perspectives. Therefore, ABC DevOps implements the additional automated production checks (see example in Figure 5). It involves automated sanity test, which runs every few minutes and send alert alarm on mobile to check any customer impact due to a deployment. These checks have been divided into 5 minutes and hourly checks based on their criticality and execution time.

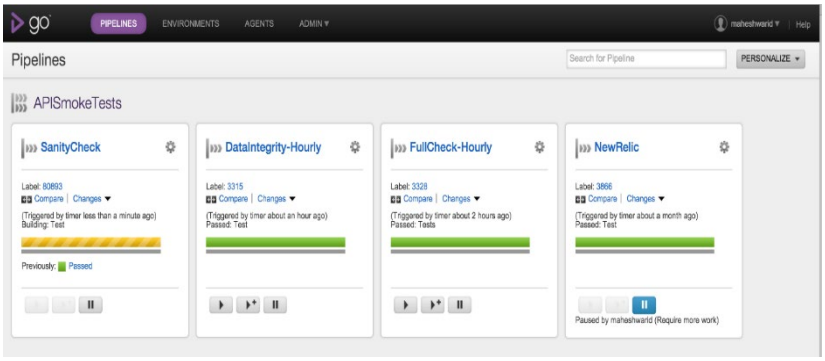

Fig. 5. Automated Production Checks

Each iteration involves at least two showcases, one for technical understanding to internal team and one is for business external to customer. This enables the team to quickly identify and address any technical and business requirements related concerns during the iteration. In addition to product owner, customer care team is also involved during the business showcase. Further, in order to keep the distributed agile feature teams aligned and synchronized, ABC maintains a card wall or portfolio of features (shared vision) and roadmap organized into next 3, 6, 9 and 12 months. This helps the distributed feature teams to understand the holistic picture (shared vision and roadmap) while working on their local features.

Post-Iteration Implementation (Heuristics)

Post-iteration heuristics involves iteration retrospective. In addition to traditional retrospective, it also involves process self-assessment. The secondary process owners run regular self-assessments to ensure conformance to the mandates and records identified by the team. This is achieved by sighting the content in the nominated repositories against the self-assessment checklist. The Quality Manager, on a periodical basis, reviews the completed self-assessments and raise Improvement Tickets for any non-compliance issues that cannot be justified.

The ABC organization's DevOps case study analysis results summary is presented in Table 1. It is clear from the analysis that ABC has a well-established

DevOps environment within the overall distributed agile development. We also learnt that APEM reference model elements provided us with a structured mechanism or checklist to systematically analyze the DevOps case study and ensure that the important points are not overlooked.

Table 1. DevOps Case Study Analysis Results Summary

Iteration	**Services**	**Practices**	**Tools**	**Key Team Roles**
Pre-Iteration	Planning Analysis Architecture	Planning Prioritization User Story Elaboration	Mingle Confluence (wiki) HipChat	Development Team Iteration Manager Product Owner SME UXD
Iteration Implementation	Design DevOps Testing Deployment	Technical Design Automated Testing CD CI Code Peer Review Change Handling Technical Showcase Business Showcase	Mingle Confluence (wiki) GO Github Ansible HipChat	Development Team DevOps Engineer Iteration Manager Product Owner SME UXD Customer Care Team
Post-Iteration Implementation	Heuristics	Retrospective Improvement Tickets	Mingle Self-Assessment Checklist	Development Team DevOps Engineer Iteration Manager Quality Manager Product Owner SME UXD

Discussion and Conclusion

Setting up with smaller and trusted features teams to deliver features gave the ABC organization the flexibility to try out various mechanism and technologies for delivering software. Active communication and collaboration culture, and shared product vision and roadmap helped the ABC distributed agile teams to stay align and synchronized. Further, continuous feedback, learning, appreciation, and senior management support helped the teams to stay motivated to successfully implement the DevOps in their distributed agile environment over a period of 3 years. Microservices Architecture and DevOps are considered as a strong combination. However, interestingly, the ABC digital delivery lead mentioned that “with growth we

realized that we made lot of decisions like splitting monolithic application into multiple smaller services and created lot of micro services. On one hand we are seeing advantages of having micro services but there is also a risk of having too many services which will in future create more work of maintaining deployments, risk of having things implemented differently on each of the services, risk of having each services only serving few routes". This seems to suggest that organizations should procced with great caution when considering Microservices Architecture. Security could be an issue in a flexible DevOps environment. ABC deals with this issue through monthly security audit reviews on DevOps. ABC is currently looking at which fine-grained Microservices can be combined or consolidated into more coarse-grained traditional services. This chapter presented a DevOps implementation case study in a relatively a different context of entertainment gaming industry. This case study provided us several insights which could be applied to other industrial contexts. It is clear from the case study analysis that DevOps is not all about technology, it is a mix of both technology and non-technology elements. DepOps is an emerging approach for digital innovation and transformation, and marks the need for more empirical studies in this important area of practice and research.

References

1. T.Dybå and T. Dingsøyr. 2009. *What Do We Know about Agile Software Development*? IEEE Software, Sep/ Oct 2009.
2. M. Huttermann. 2012. *DevOps for Developers*. Apress, 2012.
3. M. Virmani. 2015. *Understanding DevOps & Bridging the gap from Continuous Integration to Continuous Delivery.* Fifth IEEE International conference on Innovative Computing Technology (INTECH 2015), 2015.
4. L. Bass, I. Weber and L. Zhu. 2015. *DevOps: A Software Architect's Perspective.* Addison-Wesley, 2015.
5. L.F. Wurster, R.J. Colville and J. Duggan. 2015. *Market Trends: DevOps — Not a Market, but a Tool-Centric Philosophy That Supports a Continuous Delivery Value Chain*. Gartner Report, 2015. Available: https://www.gartner.com/doc/2987231/market-trends-devops--market
6. M. de Bayser, L.G. Azevedo and R.F.G. Cerqueira. 2015. *ResearchOps: The Case for DevOps in Scientific Applications.* IFIP/IEEE IM 2015 Workshop: 10th International Workshop on Business-driven IT Management (BDIM), 2015.
7. A.Q. Gill. *Adaptive Cloud Enterprise Architecture*. World Scientific, 2015.
8. S.J Humble and D. Farley. 2010. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional; 1 edition, 2010.
9. P.M. Duvall, S. Matyas, A. Glover. 2007. *Continuous Integration: Improving Software Quality and Reducing Risk*. Addison-Wesley Professional; 1 edition, 2007.
10. S. Newman. 2015. *Building Microservices Designing Fine-Grained Systems*. O'Reilly Media, 2015.
11. Go. Continuous Delivery. Available: http://www.go.cd/. Access Date: April 06, 2020.
12. GitHub. Where software is built. Available: https://github.com/. Access Date: April 06, 2020.
13. Confluence. Available: https://www.atlassian.com/software/confluence. Access Date: April 06, 2020.
14. Ansible. Available: http://www.ansible.com/. Access Date: April 06, 2020.

15. Y.I., Alzoubi, A.Q., Gill, and A., Al-Ani. 2015. Distributed Agile Development Communication: An Agile Architecture Driven Framework. JSW, 10(6), pp.681-694.
16. G. Bou Ghantous, and A., Gill. 2017. DevOps: Concepts, practices, tools, benefits and challenges. PACIS2017.
17. A. Qumer, B. Henderson-Sellers. 2007. Construction of an agile software product-enhancement process by using an agile software solution framework (ASSF) and situational method engineering. In 31st Annual International Computer Software and Applications Conference (COMPSAC 2007) (Vol. 1, pp. 539-542). IEEE.
18. A.Q., Gill, A. Loumish, I., Riyat, and S., Han. 2018. DevOps for information management systems. VINE Journal of Information and Knowledge Management Systems.
19. Y. I. Alzoubi and A. Q. Gill. 2020. An Empirical Investigation of Geographically Distributed Agile Development: The Agile Enterprise Architecture is a Communication Enabler. IEEE Access, vol. 8, pp. 80269-80289, 2020, doi: 10.1109/ACCESS.2020.2990389.
20. G.B. Ghantous, and A.Q. Gill. 2018. DevOps Reference Architecture for Multi-cloud IOT Applications. In 2018 IEEE 20th Conference on Business Informatics (CBI) (Vol. 1, pp. 158-167). IEEE.
21. G. B. Ghantous, & A. Q. Gill. 2019. An Agile-DevOps Reference Architecture for Teaching Enterprise Agile. International Journal of Learning, Teaching and Educational Research, vol. 18. no.7.
22. R., Macarthy, & J. Bass. 2020. An empirical taxonomy of DevOps in practice. 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).