DevOps adoption in Junior Enterprise: an experience report of software development

Peterson Rodrigues[1], Juliano R. Macedo[1], Pedro H. França[1]
Luiz P. Franz[1], João Pablo S. da Silva[1], Jean F. P. Cheiran[1]

[1]Federal University of Pampa (UNIPAMPA)
Caixa Postal 118 – 97500-970 – Alegrete – RS – Brazil

{petersonlrr,juliano.rmacedo,peu06,luizpaulofranz}@gmail.com

{joaosilva,jeancheiran}@unipampa.edu.br

Abstract. *Adopting new approaches to increase software development success rate in junior enterprise context is a constant challenge for this kind of company. In this paper, we report the experience on adoption of DevOps foundations to integrate continuous delivery in software development process. This adoption had three main moments: company diagnosis of maturity, application of tools to reach DevOps maturity level, and analysis of benefits of integration through one project in the company. Results indicate better transparency in project status through interactive communication provided by unifying development and operation areas, and faster software delivery.*

1. Introduction

In the last decade, the manner in which software is delivered has drastically evolved. While corporations like Amazon, Google, Netflix and Facebook provide new features for their clients transparently, many times a day and without shutdown their services, other companies need months to update systems. The key to success in features delivery is the way in which company runs the development cycle [Duvall 2012].

Evolution of software development cycle provided by agile methodologies, and Lean Software Development above them all, has culminated in 'micro delivery' concept. These deliveries are product backlog items split into small activities that are developed and placed in a Quality Assurance (QA) pipeline until they are approved and delivered to final user [de Lange et al. 2016].

DevOps culture puts together Development (Dev), Operations (Ops) and Quality Assurance (QA), and it focuses in collaboration and integration of all company divisions, from developers to stakeholders [West and Groll 2017]. Moreover, Junior Enterprises (JE) present a leading role to disseminate and update knowledge acquired in Brazilian universities. Taking the importance of DevOps culture propagation into consideration, JE allow undergraduate students to learn practices and cultures of modern software industry, and among them is DevOps [Bogo et al. 2014].

The Junior Enterprise (JE) presented in this report is composed of a multidisciplinary team of Software Engineering and Computer Science undergraduate students. Like many other JE, our enterprise is affected by a high turnover of staff, impairing directly our capability to deliver software. Regarding these difficulties, adoption of DevOps

culture could improve maturity of development cycle, software quality and knowledge dissemination among the team.

This paper has goal to report the experience of adoption of DevOps concepts in a JE placed at Alegrete campus of Universidade Federal do Pampa (Unipampa) in Brazil. To reach this goal, we identify JE maturity level, we propose and apply a DevOps workflow with automation tools to support software development process, and we collect JE team perceptions about the impact of DevOps on our project.

Text is organized as follows: at section 2, we present main concepts of DevOps movement; at section 3, we describe our JE in which our project was run; at section 4, we present diagnosis of JE maturity and its analysis; at section 5, we report integration of DevOps movement on JE; at section 6, we discuss lessons of DevOps adoption on our project; and at section 7, we present our findings, conclusions and learnings.

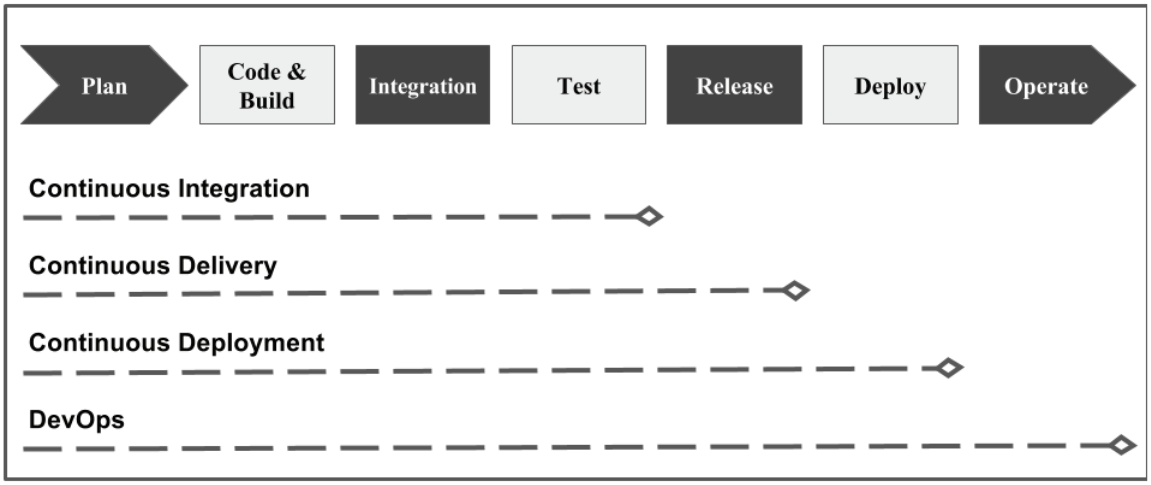
2. DevOps Movement

Software project usually present clear division of teams into roles and responsibilities. Developers create and evolve software products by adding value to business, while operations team is responsible for keeping production environment where product is deployed and consumed by clients [Sato 2014].

In this context, there is a conflict of interest between development and operations, since the first introduces new features and consequent change and the second value stability of offered services by avoiding unavailability and technical problems. This divergence affects release timing by impairing agile principles and by compromising product scheduling [Hüttermann 2012].

Agile movement describes a set of principles for software development in which requirements and solutions evolve through a collaborative effort of auto-organized, multifunctional teams [Beck et al. 2001]. DevOps movement appears as a natural evolution of agile movement and it aims to mitigate conflicts, approximate responsibilities and improve communication of Dev, Ops and QA [Hüttermann 2012, Sato 2014]. At Figure 1, we present an overview of DevOps workflow followed by detailed description.

Figure 1. Collaboration Diagram. Adapted of [Rathod and Surve 2015]



Continuous Integration (CI): It is automation of code **build** and test after every change. It is supported by tools integrated to code repository that instantly report failures on build and test tasks. In doing so, the main goal of CI is to provide a rapid response to developers by alerting when a bug is introduced on source code [Virmani 2015, Rathod and Surve 2015].

Continuous Delivery (CDel): it is partial automation of generation of new system releases. It covers CI phase, and it also includes submission of changes approved by automated testing to a QA team for review. It usually means that application deploy is manually done shortly after QA approval. This approach is commonly applied to safety critical, money critical or information security systems which depend on high availability [Humble and Farley 2010].

Continuous Deployment (CDep): unlike CDel, CDep brings complete automation to system release and deploy. After adding new features and after running automated tests of CI, a new release is automatically sent to deploy system and it is directly deployed at production environment. This strategy is used by most of Software as a Service (Saas) applications like social networks, email servers, etc [Claps et al. 2015].

DevOps: it is complete integration and automation of previous approaches, aiming the ability to develop and deliver software continuously through a transparent, repeatable process. This process promotes full collaboration of Dev, Ops and QA teams in order to allow continuous delivery and infrastructure automation which easily respond to software evolution. DevOps movement is largely used on industry, and it recently has reached academy [West and Groll 2017].

De Lange et al. [de Lange et al. 2016] performed an experiment at RWTH Aachen University that aims to teach DevOps movement to undergraduate students. They investigate what is the technical preparation level of students to perform functions on DevOps-compatible start-up companies after learning about DevOps. Authors also highlight that the experiment allowed to students to reduce the gap between university learning and industry practices.

Perera et al. [Perera et al. 2016] present a study performed on Sri Lanka companies that has found correlations between quality, business adaptation needs and agility in DevOps adoption. In order to do this, they performed interviews with Information Technology (IT) professionals who knew DevOps movement.

Both aforementioned studies do not cover JE peculiarities and this kind of material is deeply scarce on scientific databases, so experience reports on this environment are pioneer and necessary.

3. Our Junior Enterprise (JE)

Ideiah Soluções em Software JE was founded in 2012. The company consists of Software Engineering and Computer Science undergraduate students, and it currently has seven members on staff and one adviser professor.

Since 2014, our JE has adopted agile approaches for flexibility and standardization in product development. So, a Scrum team [Schwaber et al. 2016] composed of five developers, one Product Owner and one Scrum Master continuously perform Extreme Programming (XP) practices [Beck 2000] along software development.

Employing daily Scrum (or daily meeting) and other techniques, we noticed deficiency in used tools for software coding, testing and delivering and we also observed a centralized responsibility on tools that had caused delivery delays. Particularly, we had not promoted homogeneous autonomy on tools to avoid the bottleneck caused by a largely concentrated infrastructure.

Moreover, tools were not mutually integrated, resulting in inefficient communication between development team and infrastructure management team. Since developers seek to deliver software as soon as possible and infrastructure team needs to keep production environment without issues, it is obligatory to reduce this gap.

These problems resulted in many delivery delays and therefore improvement of JE processes is necessary. In order to address problems with tools and responsibilities, we perform a brief literature review on strategies and instruments to identify JE maturity level. Due to this search, DevOps movement emerged as an approach to mitigate issues and to collaborate in supporting agile philosophy of our company. Results of diagnosis of JE maturity is presented at next section.

4. Diagnosis of Maturity

Diagnosis of maturity seek to identify practices and tools adopted by JE and their integration. With this in mind, we can recognize what is missing to effectively implant DevOps movement. Once DevOps has evolved from agile movement, JE already meets the requirement to embrace agile culture.

Figure 1 was used to guide diagnosis process and to implant DevOps concepts integrally. We performed intensive, direct observations [de Andrade Marconi and Lakatos 2010] during an entire development cycle, mapping practices and tools. At the end of the cycle, we compiled information and compared it to DevOps principles and workflow.

Continuous Integration (CI): our JE uses version control and remote distributed repository as integration resources. Tools for test writing and automation are also used, but there was not integration to repository or other tools. In addition to that, there was not policies and support on test reuse and there was not support on test rerunning by coverage analysis tools.

Continuous Delivery (CDel) and Continuous Deployment (CDep): there was no software delivery support tool. As a result, we were heavily overloaded with manual software delivery to QA and with manual software deployment into production environment.

DevOps: JE did not applied a clear, repeatable process of software delivery. It was necessary to manually deploy software and to manually adapt infrastructure after each release to support software evolution. Also, communication between development, operations and QA teams was not efficient.

Since agile culture was met and collaboration of JE divisions seems fulfilled, the adoption of new development workflow focused on environment integration and automation should be sufficient to support DevOps and to mitigate aforementioned problems.

5. Implanting DevOps

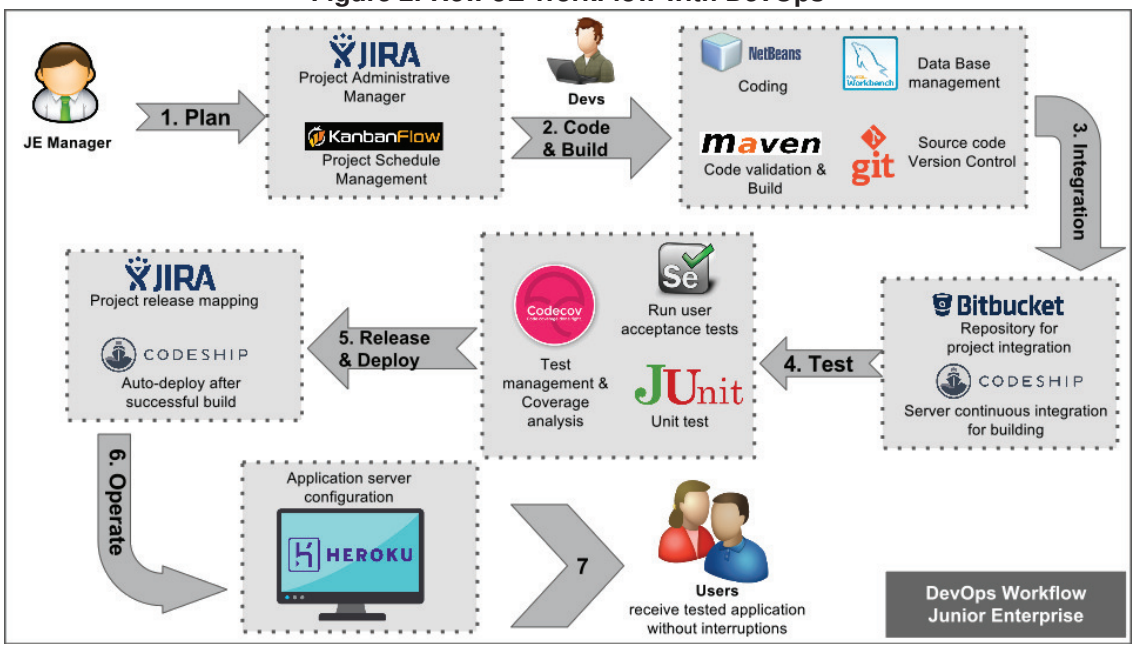
Selection of tools to support integration and automation for a JE commonly has a serious constraint. Because the usual JE educational, nonprofit profile and because the lack of institutional funding, we need to search low cost – or completely free – tools. Even though there is eventual incomes from external sources, JE business model does not assure continuous support to maintain non-free software. Besides that, it is also necessary to choose "easy to learn" tools in order to minimize adaptation problems caused by a high turnover of staff.

JE members from outside the project in which DevOps would be implanted searched tools that fit our context, verified their limitations and costs, and performed pilot tests on another JE software development project.

5.1. Tools identification and selection

Through literature review focused on needs highlighted at section 4, we select and test tools to support software development process. At Figure 2, we draw the new JE workflow with chosen tools[1] and the phase they are applied.

Figure 2. New JE WorkFlow with DevOps



Tools at Figure 2 interact each other in favor of complete integration and automation – from coding to deployment.

We perform the entire project planning (stage 1) with JIRA and KanbanFlow by providing software scope, scheduling milestones and defining production strategies. NetBeans, MySQL Workbench, Maven and GIT cover development and change control

[1]Tools available at `http://jira.atlassian.com` — `www.kanbanflow.com` — `www.netbeans.org` — `http://maven.apache.org` — `www.git-scm.com` — `www.mysqlworkbench.org` — `www.bitbucket.org` — `www.codeship.com` — `www.junit.org` — `www.seleniumhq.org` — `www.codecov.io` — `www.heroku.com`

(stage 2). New or changed artifacts are sent to Bitbucket repository, and it allows CodeShip to try to build the project (stage 3). CodeShip itself runs test scripts (using JUnit and Selenium in our project) while building, and CodeCov runs tests again to keep updated the coverage analysis and other statistics (stage 4). If building is succeeded, CodeShip does an auto-deploy to application server Heroku and JIRA update project status when the new release is detected at repository (stage 5 and stage 6). At the end of this process, users receive a new version of software without service interruption (stage 7).

5.2. Adoption of new workflow

The new workflow implantation was done in two moments: (1) pilot implantation with a small team and (2) full project implantation with the whole team.

In order to test the new workflow, we ran a pilot study during a small increment in a stable project. So, we could identify adaptations in development team practices and particularities of brand-new tools. Workflow implantation time was short, since the team uses branches to organize development. As result, new tools tutorials and workflow overview document were created to facilitate understanding of all.

Furthermore, DevOps workflow was completely integrated into a project for developing an Enterprise Resource Planning (ERP) software and it involved the whole team. Tutorials in fact made implantation easier even for a small, agile team that embrace the change. Detailed information about implantation in production environment is showed at section 6.

6. Lessons Learned

The main project of this report is running for four months using DevOps principles, and it was already running for twelve months when DevOps adoption started. Since DevOps implantation, we performed regular diagnostic evaluation sessions with the team to collect feedback about the new workflow. Besides discussing lessons of DevOps in our JE business environment, we analyzed test coverage statistics to investigate collateral benefits of new tools.

Opinions gathered in diagnostic evaluation suggest positive lessons and team satisfaction from DevOps adoption. Not to mention the overall perception of more effective software delivery, we point out the following main learned lessons:

Agility in identifying bugs: bugs in the project are quicker identified and addressed, since tools run tests more frequently and they allow transparency along the entire workflow to the team. Moreover, new tools allow to monitor real-time project status for developers and managers.

Issues reduction: full test automation prevents developers to skip test phases and assure that failures to be detected earlier that production.

Better communication between Development and Infrastructure teams: even in our agile context, there was a natural conflict between Dev and Infra. When responsibilities are decentralized, communication intensifies and conflict decreases.

Continuous deployment: automated release and deployment increase available time to perform acceptance testing. So, development team and infrastructure team contribute directly to QA.

Learning new tools: there is a motivational impact on JE team, because the very educational nature of a JE environment. Since there is no incomes for its members, learning is a key value to keep staff and to develop professional abilities.

Quicker delivery: as a result of eliminating the gap between end of coding and software delivery, each software increment is delivered quicker.

Improving business value: spreading responsibilities and intensifying communication foster team cohesion to reach a goal, increasing business value.

Code coverage and acceptance testing information reinforce benefits observed on team opinion. There is a significant increase in code coverage assessed through CodeCov before new workflow started (3% coverage) and after four months on new workflow (68% coverage). 3% initial coverage represented only tests on critical parts of the software, so 68% current coverage represents a meaningful increase, considering the complexity of reported project (an ERP software).

Software acceptance rate also increased after DevOps. Before new workflow started, one in four releases had been accepted by product owner according requirements and by team according quality criteria. After new workflow, three in four releases was accepted, and the rejection of the first release was possibly caused by new technological challenges and stricter quality criteria.

As a major lesson, we notice the importance of an organization open to change and the increase of competitive factor through adoption of new technologies and techniques even if it demands technical effort and cultural changes by team members.

7. Final Considerations

We presented in this paper an experience report describing DevOps movement adoption in a software development Junior Enterprise (JE). Still, we presented main DevOps features to locate our JE agile practices into DevOps workflow.

Results indicate DevOps is characterized not only by behavioral features native of agile approaches but also by productivity increase through automated tools. As subject, we present a new software development workflow with stages and tools that promotes deployment automation and communication improvements.

Our main findings reinforce benefits of agile approaches integrated to DevOps culture. So, we has achieved (1) quicker failure detection caused by bugs and by infrastructure issues, (2) more visibility to project status, (3) better communication between development team and infrastructure team, (4) continuous deployment, and (5) quicker delivery.

An important limitation to this paper is project confidentiality clauses that prevents detailed data presentation in order to support statements and findings. Nonetheless, we believe this report solidly contributes to DevOps movement literature by presenting new case study in a very particular kind of company (Junior Enterprise) and by detailing workflow information.

Future work includes a controlled experiment in our JE with explicit authorization to use project artifacts, data and statistics, so we could reevaluate the new workflow applicability.

References

[Beck 2000] Beck, K. (2000). *Extreme programming explained: embrace change*. addison-wesley professional.

[Beck et al. 2001] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.

[Bogo et al. 2014] Bogo, A. M., Henning, E., Schmitt, A. C., and Marco, R. G. D. (2014). The effectiveness of junior companies from the viewpoint of engineering students at a brazilian university. In *2014 IEEE, EDUCON*, pages 745–750. IEEE.

[Claps et al. 2015] Claps, G. G., Svensson, R. B., and Aurum, A. (2015). On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software technology*, 57:21–31.

[de Andrade Marconi and Lakatos 2010] de Andrade Marconi, M. and Lakatos, E. M. (2010). *Fundamentos de Metodologia Científica*. Atlas, São Paulo.

[de Lange et al. 2016] de Lange, P., Nicolaescu, P., Klamma, R., and Koren, I. (2016). Devopsuse for rapid training of agile practices within undergraduate and startup communities. In *ECTEL*, pages 570–574. Springer.

[Duvall 2012] Duvall, P. (2012). Breaking down barriers and reducing cycle times with devops and continuous delivery. Online. Acessado em Julho–2017.

[Humble and Farley 2010] Humble, J. and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education.

[Hüttermann 2012] Hüttermann, M. (2012). Beginning devops for developers. *DevOps for Developers*, pages 3–13.

[Perera et al. 2016] Perera, P., Bandara, M., and Perera, I. (2016). Evaluating the impact of devops practice in sri lankan software development organizations. In *Advances in ICT for Emerging Regions (ICTer)*, pages 281–287. IEEE.

[Rathod and Surve 2015] Rathod, N. and Surve, A. (2015). Test orchestration a framework for continuous integration and continuous deployment. In *2015 International Conference on Pervasive Computing (ICPC)*, pages 1–5.

[Sato 2014] Sato, D. (2014). *DevOps na prática: entrega de software confiável e automatizada*. Editora Casa do Código.

[Schwaber et al. 2016] Schwaber, K., Sutherland, J., and Beedle, M. (2016). The definitive guide to scrum: the rules of the game. *Scrum Guide*.

[Virmani 2015] Virmani, M. (2015). Understanding devops & bridging the gap from continuous integration to continuous delivery. In *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on*, pages 78–82. IEEE.

[West and Groll 2017] West, D. and Groll, J. (2017). The convergence of scrum and devops. Online, Scrum.org and DevOps Institute. Acessado em Setembro–2017.