3rd International Conference on Higher Education Advances, HEAd'17
Universitat Politècnica de València, València, 2017
DOI: http://dx.doi.org/10.4995/HEAd17.2017.5607

Teaching Agile Development with DevOps in a Software Engineering and Database Technologies Practicum

Mason, Robert T. [a]; Masters, William [b] and Stark, Alan [c]
[a]College of Computer & Information Sciences, Regis University, USA, [b]College of Computer & Information Sciences, Regis University, USA, [c]Center for Scholarship and Research Engagement, Regis University, USA

Abstract

DevOps is a new concept for Software Engineering. Teaching DevOps can be challenging with the limited resources that are available at many universities. This paper exams how to teach of an Agile Development Methodology using a DevOps approach for the Regis University (RU) M.S. in Software Engineering and Database Technologies Practicum. With faculty support, heavy stakeholder involvement and RU Information Technology Services department (Operations Support) mentoring, students were able to successfully follow the Agile Development methodology to create an application that was incoporated into the RU Web-site infrastructure.

Keywords: *Teaching DevOps, Agile Development*

1. Introduction

DevOps is a new term for the subject of Software Engineering that combines the areas of Software Engineering (a.k.a. Software Development) and Operations Support into one methodology. Traditionally, the development of software by Software Engineering organizations has followed the Systems Development Life Cycle (SDLC). This is a methodology that has been used by project management for the last 40 years to manage software development projects. The SDLC usually includes a feasibility study (planning), analysis, design, coding, testing and then deployment of the application to the production environment. After deployment, the maintenance of the new application is often handed off to the technical support personnel in the Operations Support organization. Unfortunately, if the new software has defects, then Operations Support is tasked with resolving the defects (e.g. fixing the errors or finding work arounds). Over the years, this separation of concerns has created barriers and tension between Software Engineering and Operations Support.

Jabbari, Ali, Peterson and Tanveer (p. 18, 2016) completed a systematic mapping study of 44 papers from six electronic databases (e.g. ACM, IEEE, Inspec, etc.) on the topic of DevOps and then synthesized the following definition:

"DevOps is a development methodology aimed at bridging the gap between Development (Dev) and Operations, emphasizing communication and collaboration, continuous

integration, quality assurance and delivery with automated deployment utilizing a set of development practices."

Fitzgerald and Stol (2014) recognized that DevOps is the integration between software engineering and operational deployment and that it should occur on a continuous basis.

The term Continous Software Engineering includes the entire software life-cycle sub-phases of Business Strategy & Planning, Development and Operations. Continuous processes can include deployment, planning, integration, testing, and run-time monintoring.

Continuous deployment can be a radical shift in comparison to the conventional waterfall release cycle that can take months to deploy new software into a production environment. Spencer (2014) proposed an example of a agile release cycle that involves:

- Small code changes are made, tested & deployed on a daily basis
- Functional requirements are updated every other day
- Operations Support testing occurs 1 – 2 times per week
- Hardware installation done as needed
- Early Life Support (ELS) and Continual Service Improvement (CSI) is done on a daily basis

Humble and Molesky (2011) characterized the principles of Continuous Software Engineering and DevOps as a shared responsibility for collaboration between Development and Operations Support that uses automation and measurement to improve software quality.

The DevOps handbook the elaborates on the three ways for implementing DevOps that are listed below. (Kim, Humble, Debois, Allspaw, & Willis, J., 2016). The three ways were initially presented in an IT novel by Kim, Behr, & Spafford (2014). The 1st DevOps principle is the implementation of an accelerated delivery flow from Development to Operations Support to Production (Customers) and includes:

- Controlling the queue size of Work in Progress (WIP) to be very small.
- Small batch sizes of work – less WIP, faster lead times, error detection and less rework.
- Test environments that built in advance and continually updated.
- Loosely coupled architectures, such as Sevice Oriented Architecture.
- The 2nd Principle is feedback that enables the creation of ever safer systems of work and includes:
- Defects or significant deviations are quickly found and acted upon by the team.
- Feedback loops are created at all levels, Project Management, Development, QA, Info. Security and Operations Support.
- A swarm approach by all team members is taken to solve problems with whoever needs to be involved. This involves learning and immediate resolution to the problem.
- The 3rd Principle is Continual Learning and Experimentation and includes:
- Changing the culture of the organization to value organizational learning and institutionalize the concept of improving daily work
- Change the culture to accept the fact that failures will always occur in complex systems
- Encourage an open dialogue about problems across departments
- Local learning should be shared globally

2. Teaching DevOps and Agile Development

Christensen (2016) outlined his challenges for teaching DevOps and proposed teaching techniques focused on DevOps. Teaching DevOps can be challenging because of the following reasons:

- Limited Operations Support experience of the faculty teaching DevOps
- Most university courses are focused on development or operations, but not both topics
- DevOps is focused on hands-on skill competency, such as coding, database configuration, etc.
- Creating a realistic production environment for testing applications can be a daunting task because of financial constratints at many universities
- Evaluating student work can be difficult without actual testing in a production environment, especially performance tuning for high volume transactions

Based on these challenges, Christensen (2016) developed a Cognitive Apprenticeship and Story-Telling Approach for teaching DevOps within a Cloud Computing course that is taught at Aarhus University in Denmark.

- Students designed and coded a simple distributed system called "SkyCave"
- Student deliverables included the documentation and code for the application
- Grades were based on the amount and quality of the work submitted
- No specific deadlines were enforced except for the final end of course deadline
- Feedback for the class exercises was completed with 24 hours of the student submission

Sjodin and Barnes (2016) hosted a panel discussion at the 25th Annual CCSC Rocky Mountain Conference Program at RU in Denver, Colorado. They discussed an Agile Development Methodology in Computer Science courses offered by CC&IS. This type of rapid software development is in alignment with what Spencer (2014) was suggesting. The panel proposed teaching methodologies for both online and classroom courses that included fully automated real time validation, continuous testing and deployment. Highlights from the panel discussion on the Agile Development Methodology are as follows:

- Project Charter – should consist of a one paragraph executive summary, identification of stakeholders, use case diagrams based on business requirements
- Initial Planning – Use Cases are prioritized, a tentative schedule is developed, and the product backlog is reviewed
- Iterations – Elaboration of specific Use Cases (descriptions, write ups, fully-dressed)
- Design - UI: Site Map, Wire Frame Mockups, Class Diagrams, Sequence Diagrams
- Configuration Management using Git, such as the Gitlab or Github tools
- Iterative Development (1 – 2 weeks) with Unit & Functional Testing
- Standups (i.e., providing status with the 3 Ps: progress, problems, plans)
- Incremental, Continuous Deployment as suggested by Fitzgerald and Stol (2014)

3. RU CC&IS Software Engineering and Database Practicum

The RU College of Computer and Information Sciences (CC&IS) Software Engineering & Database Practicum is comprised of two courses that satisfy an exit strategy requirement for the Master of Science in Software Engineering and Database Technologies (MS SEDT) program. A total of 36 credit hours (12 courses) are required to graduate from the program. After completing many of the software engineering and database courses, students participate in the SEDT Practicum for one semester (two concurrent eight week terms - Practicum I and II).

The SEDT Practicum II course facilitates a real-life, hands-on learning experience via the development of operational software. Students participate on teams of 4 – 6 students and follow an Agile Development Methodology as outlined by Sjodin and Barnes (2016). The SEDT Practicum Project allows students to demonstrate Software Engineering and Database Design skills that been acquired in earlier course work.

The SEDT Practicum I course provides additional training in the form of lab exercises on the topics of Microsoft SQL Server, MS Visual Studio using C++, MVC 5 Scaffolding, Linux and Oracle Enterprise Manager. In addition to technology training, students participate in an Operations Support environment working as Database Administrators. Students create and manage all of the databases that are used in the undergraduate and graduate courses within the RU CC&IS. Students staff a help desk and resolve a variety of operation support issues for the databases within the Linux OS and Windows OS environments. Tasks include space management, creating user IDs, granting access privileges, installing vendor software and performance tuning the databases.

3.1. RU Scholar Database Project

In the Fall 2016, the administrator for the RU Center for Scholarship and Research Engagement (CSRE) approached the lead faculty for the SEDT Practicum. He asked if the practicum students could participate in the development of a web-based database application to assist with the management of Research materials for the five colleges within RU. Normally, this project would be developed by the RU Information Technology Support (ITS) Operations Support team, however ITS was unable to work on the project for at least a year based on a huge backlog of other projects with higher priorities.

The Regis Scholar Database Project was selected for the practicum because the initial release size was small and would support the Agile Development Methodology described by Sjodin and Barnes (2016). For example, the initial project release involved the design and creation of eight relational database tables and their supporting web-pages. The primary function of the application is to allow prospective students, media, faculty scholars and others to quickly and easily link scholars with related research interests. The home application webpage (shown in Figure 1) was added as a link from the main RU Library Digital Commons webpage because it is a location that faculty and students use to find Research materials. Therefore, it made sense to co-locate this new web-application with other commonly used Research links, such as the online Research Databases (e.g. ACM, IEEE) and links to other library support services.

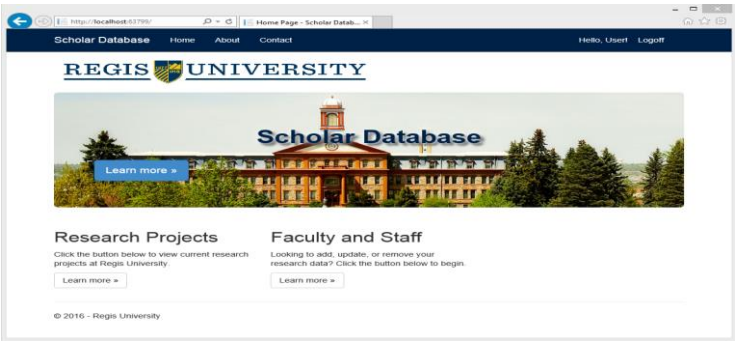

Figure 1. The home page of the RU Scholar Database application (2016).

4. Results of using the Agile Development Methodology with DevOps

The following chronology of events documents the results of using the Agile Development Methodology with a DevOps Approach within the MS SEDT Practicum. Students and Faculty met at online Skype meetings 2 – 3 times per week to discuss and review the project deliverables. Students within the program are located throughout the USA, therefore online meetings are the only feasible way to meet as a group synchronously.

Project Charter and High-Level Documentation – Students developed a Project Charter that consisted of a paragraph for an executive summary, identified the stakeholders (RU CSRE staff, Faculty, external entities, etc.), documented the Business Requirements and then created the Use Case diagrams based on the Business Requirements. These deliverables were reviewed for approval by the Faculty Lead and then the RU CSRE staff.

Initial Planning – The Use Cases were prioritized by Faculty Lead and the students. Some Use Cases were deferred to future product releases which made the scope of the project very small. Subsequent approval of the release size was obtained from RU CSRE staff via an online meeting. Because the project was using an Agile Development approach, one Use Case was selected for initial development. The Use Case was elaborated upon with descriptions, write ups and basic logic to meet the criteria for a Fully-Dressed Use Case and reviewed by the lead faculty. A very basic ERD was created for the first Use Case and a few Wire-Frames (Web-page designs), Class Diagrams and Sequence Diagrams were also created within a few days.

Initial RU ITS (DevOps) Touch Point - Prior to beginning coding of the first use case, the Faculty Lead met with the RU ITS Operations Support team to collaborate on the development platform and development coding standards. Since RU ITS would later support the software in production, they provided suggestions for the development platform, ideas for testing the code in the Integrated Development Environment (IDE) and they shared their development process and coding standards. ITS agreed to conduct code review and thus provide mentoring for the students on Best Practices. The development platform was as follows:

- Microsoft (MS) Visual Studio (VS) Community Edition 2015 with C++ using MVC and Scaffolding. ASP .net is not used by Regis ITS. The VS Community Edition 2015 is free for download and provides a good IDE for coding and unit testing. VS also has a local database that simulates a connection to MS SQL Server.
- Microsoft SQL Server was used as the Relational Database Management System because it is the standard database used by RU ITS in production.
- Gitlab was selected for Configuration Management. Gitlab was choosen instead of GitHub because it allowed the projects to be concealed (hidden) from the public

and the web application is free to use. This product was recommended by RU ITS because they use a local Gitlab repository to house all of their development work. Thus, it was easy for ITS to review the code as it was being developed and checked into the repository and then incorporate the deliverables into their local Gitlab permanent repository. Also, Gitlab allows for separate development tracks for each student that can later be merged with the main development track after unit testing.

- Unit Testing was accomplished using local Windows machines owned by students and by Windows Virtual Machines (VM) on the RU Network that the Faculty Lead created for the students. A VM Windows Server 2012 was created and SQL Server was installed in that machine.

DevOps review of initial code – After students developed and tested the initial code for the first Use Case, RU ITS was asked to review the code to provide feedback. This enabled the students to get immediate feedback and thus incorporate the lessons learned into the code for the remaining Use Cases.

Begin Small Development Iterations - The remaining Use Cases (within the project scope) were elaborated upon with descriptions, write ups and basic logic to meet the criteria for a Fully-Dressed Use Case. Completed Use Cases were reviewed with the Faculty Lead and then the RU CSRE staff. Initial review by the Faculty Lead allowed the students to make minor corrections to the Use Cases prior to the RU CRSE review.

Remaining Design of other Components – The design stage leveraged the remaining Use Cases to develop a complete Entity Relationship Diagram (ERD) in third normal form. This diagram was reviewed by the students and the Faculty Lead several times. Initial drafts of the ERD were sent to the Faculty Lead via email and then the final draft ERD was reviewed during an online meeting. Eight relational tables (six new tables) were then created with DDL from the ERD. After ERD approval, students created Wire Frame Mockups (Web-Page Designs), Class Diagrams and Sequence Diagrams for the remaining Use Cases. The Wire Frames Mockups were reviewed with the Faculty Lead and then the RU CSRE staff for approval. The students presented the Wire Frame Mockups at a University-Wide Faculty Presentation by the RU CSRE staff in regards to the new application. The Class Diagrams and Sequence Diagrams were reviewed during the online meetings with the Lead Faculty. All of the design documents were added to the Gitlab project and shared with the development team so that team members could learn by reviewing other students work.

DevOps review of remaining code - As mentioned previously, RU ITS agreed to mentor the students by performing code reviews of the code for the remaining Use Cases. Verbal and written feedback was given to the students regarding adherence to the RU ITS development standards and best practices.

Code Adjustments – were made to the application software based upon the code review feedback in preparation for the final delivery/turn-over of the software to RU ITS. A final overall code review was conducted with RU ITS prior to accepting the code for production. RU ITS made minor enhancements to the code by adding LADP access code for user authentication which they wanted to keep internal (secure) to their department.

Incremental, Continuous Deployment – Enhancements to this application will follow the same Agile Development methodology that uses a DevOps approach. Small incremental improvements will be made to the software and then will be tested in the VM Cloud environment.

5. Conclusion

The Scholar application developed by the MS SEDT students was well received by RU ITS Department (Operations Support). The ease of acceptance resulted from a) adhering to the mentoring from the RU ITS Department, b) following the development standards and c) the use of the RU ITS development platform (e.g. Visual Studio C++, MS SQL Server and Gitlab). The code review provided by the RU ITS experts to the practicum students was immensely beneficial because it provided a real-life learning experience (feedback from experts) to the students. The use of the Agile Development Methodology using a DevOps approach was a successful endeavor because it included the direct involvement of the RU ITS (Operations Support), mentoring by the faculty and heavy stakeholder (RU CSRE staff) participation throughout the project.

References

Christensen, H. B. (2016). Teaching DevOps and Cloud Computing using a Cognitive Apprenticeship and Story-Telling Approach. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (pp. 174-179). ACM.

Fitzgerald, B., & Stol, K. J. (2014). Continuous software engineering and beyond: trends and challenges. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (pp. 1-9). ACM.

Humble, J., & Molesky, J. (2011). Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8), 6.

Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (2016). What is DevOps?: A Systematic Mapping Study on Definitions and Practices. In Proceedings of the Scientific Workshop Proceedings of XP2016 (p. 12). ACM.

Kim, G., Behr, K., & Spafford, G. (2014). The phoenix project: A novel about IT, DevOps, and helping your business win. IT Revolution.

Kim, G., Humble, J., Debois, P., Allspaw, J., & Willis, J. (2016). The DevOps handbook.

Sjodin, R., & Barnes, S. (2016). Teaching agile methodologies and DevOps/CI/CD in the classroom: concepts, techniques, modalities: panel discussion. Journal of Computing Sciences in Colleges, 32(2), 90-91.

Spencer, R. (2014). DevOps and ITIL: Continuous Delivery Doesn't Stop at Software. ChangeAndRelease.Com. Retrieved May 11, 2016, from https://changeandrelease.com/2014/04/05/devops-and-itil-continuous-delivery-doesnt-stop-at-software/.