International Journal of Learning, Teaching and Educational Research
Vol. 18, No. 7, pp. 128-144, July 2019
https://doi.org/10.26803/ijlter.18.7.9

An Agile-DevOps Reference Architecture for Teaching Enterprise Agile

Georges Bou Ghantous and Asif Qumer Gill
University of Technology Sydney
Sydney, Australia

Abstract. DevOps emerged as an important extension to support the Agile development for frequent and continuous software delivery. The adoption of Agile-DevOps for large scale enterprise agility depends on the most important human capability such as people competency and experience. Hence, academic education and professional training is key to the successful adoption of Agile-DevOps approach. Thus, education and training providers need to teach Agile-DevOps. However, the challenge is: how to establish and simulate an effective Agile-DevOps technology environment for teaching Enterprise Agile? This paper introduces the integrated Adaptive Enterprise Project Management (AEPM) and DevOps Reference Architecture (DRA) approach for adopting and teaching the Agile-DevOps with the help of a teaching case study from the University of Technology - Sydney (UTS), Australia. These learnings can be utilised by educators to develop and teach practice-oriented Agile-DevOps for software engineering courses. Furthermore, the experience and observations can be employed by researchers and practitioners aiming to integrate Agile-DevOps at the large enterprise scale.

Keywords: Agile framework; Agile adoption model; Agile and DevOps; DevOps reference architecture.

1. Introduction

Agile methodology focuses on solving computing and software engineering problems both from human and technology perspectives (Alzoubi et al. 2015; Gill 2015; Neve et al. 2017). Recent advancements in software engineering practices and methods facilitated the adoption of improved Agile ways of working to deal with the complex nature of software problems (Alzoubi et al 2018; Qumer et al. 2007). Human factor, in particular, has direct influence on the quality of a software whether it is at the academic, research or industry level (Mason et al. 2017; Gill et al. 2018;). It has been observed that communication and collaboration present a major challenge to development teams in the IT industry (Alzoubi and Gill 2014), in particular, if a team is geographically distributed. In

order to address the challenges of team collaboration, Agile requires a project management system that enables constant communication among people in software development (Bai et al. 2018). The modern Agile approaches require also fast product delivery and deployment. The Agile product development may be achieved through incremental software releases and iterations; accordingly, an overall model for Agile project management system is preferred if it enables automation and continuous integration. These two concepts (Bou Ghantous and Gill 2017) are the corner stone of DevOps approach. By relying on automation and continuous integration, Agile frameworks or models may vertically augment the speed and quality of software (Gill 2014; Perera et al. 2017).

The contemporary adaptive enterprise project management (Gill 2015b) would require DevOps practices and tools to guide the enterprise scale Agile-DevOps transformation for fast software delivery (Snyder et al. 2017). Software productivity and frequency would require essential DevOps continuous integration to enhance traditional Agile scrums. Relationship between DevOps and Agile Project Management (Lwakatare et al. 2016) would positively improve Quality of Software by enabling real-time monitoring and automated testing which consequently facilitate feedback loop mechanism for continuous improvement. Further, Quality of Service can be achieved by increasing the deployment speed and frequency through continuous deployment (Colavita 2016). It is anticipated that typical model driven continuous deployment quality can be ensured by using DevOps practices and tools (Artac et al. 2016). Agile team can get clear insights into the project lifecycle status using DevOps monitoring and reporting tools (Gill et al. 2017). To reach vertical Agile maturity level (Qumer et al. 2007), development relies on continuous delivery assessment (Bai et al. 2018) through real-time feedback from DevOps monitoring system (Bou Ghantous and Gill 2018). This also enables collaboration and minimizes the constraints of distributed teams (Wang and Liu 2018). By using DevOps (Mohamed 2016), organizations, practitioners and researchers would be able to adopt and scale Agile at the large enterprise level.

DevOps seems to be an interesting approach. However, the challenge is how to effectively teach and learn DevOps to students so that they can effectively adopt it to their software development environments? The purpose of this paper is to provide such approach that can facilitate the teaching and learning of DevOps. This paper uses the collaborative learning theory (Garfield 1993) to inform the method of teaching of DevOps and Agile for large enterprise scale, which is not an easy task. Thus, we also used a DevOps Reference Architecture (DRA) models (Bou Ghantous and Gill 2018) within an enterprise scale adaptive enterprise project management (AEPM) capability reference model (The Gill Framework) (Gill 2015b) to provide students with large scale Agile-DevOps software engineering experience in such a way that fits into a single semester. This paper reports our learnings from a teaching case at the University of Technology Sydney (UTS) and addresses the following main research question and sub-questions:

RQ: How to effectively teach Agile-DevOps for the delivery of enterprise scale portfolio of software projects?

This research question has following 3 sub-questions:

- RQ1: How to establish and simulate an effective Agile-DevOps technology environment for teaching?
- RQ2: What are the reference models available for establishing such environment for teaching?
- RQ3: How to teach using the established Agile-DevOps environment?

This paper aims to address the above mentioned research questions and is organized as follows. Firstly, it presents the teaching case study context. Secondly, it discusses the AEPM capability reference model for teaching the Agile for the large scale portfolio of projects. Thirdly, it explains the DRA models for teaching the DevOps within the AEPM. Fourthly, it highlights the vital factors and elements required for a successful integration of the AEPM and DRA for establishing and using the Agile-DevOps environment for teaching. Finally, it concludes with possible options for further research and improvements.

2. Case Study Context

UTS offers an undergraduate subject (Software Engineering Practice) in spring sessions for (approx. 200 students). SEP duration is 12 weeks through face-to-face workshops (three hours length each workshop, 6 credit points). The course runs every spring session. The data collected for this paper are extracted from spring 2016 and spring 2017 semesters when we started integrating the Agile and DevOps approach for teaching SEP subject.

SEP starts in week 1 with initial introductory induction lecture or session which focuses on the recent software industry trends and also introduces fundamental Agile concepts, principles, practices and methods for large scale software engineering. From week 2 until week 12, students are distributed in collaborative learning workshop (approx. 40-50 students in each workshop) based on the collaborative learning theory (Garfield 1993). The workshop managers then setup the students in each workshop in groups. Each group has 5 or 6 students. Students are required to apply the modern Agile practices and develop an industry level project over a period of 12 weeks. The software (each group choses their own idea) is expected to developed using Agile methodology and various technologies, programming languages and tools including Agile and DevOps. The project delivery is divided into two releases (R0 and R1) with an optional third release (R2). Each release is composed of 4 iterations (I0-I3). Groups may choose to continue developing their project idea after graduating from the subject. This collaborative and flexible approach encourages students to take their project idea to the industry or add it to their professional resume. Groups developing R0 and R1 are expected to apply:

- Agile requirements analysis and planning.

- Agile architecture and design.
- Agile implementation and testing by means of DevOps approach using DRA as template.

The teaching materials are based on the overall AEPM capability reference model from The Gill Framework (Gill 2015b). Workshop manager (tutor) manages multiple groups developing various projects as a portfolio/ program of projects. DRA (Bou Ghantous and Gill 2018) is utilized and taught to students within the AEPM to construct automated software development and deployment pipelines using a range of DevOps practices and toolchain. Integrating DRA in the course allows to educate students how to apply DevOps practices in the Agile development process for fast software release management. Hence, in this paper we present our observations and experience as managers or teachers of multiple teaching workshops (spring 2016 and spring 2017 semester). Thus, in order to address the research questions in-hand, this paper demonstrates how can we teach enterprise scale Agile-DevOps using the AEPM (Gill 2015b) and DRA (Bou Ghantous and Gill 2018) .

3. The AEPM Reference Model

The Gill Framework® (Gill 2015b) is meta-framework for defining, operating, managing, supporting and adapting capabilities (see Fig. 1). Thus, it is not another Agile method. This adaptive or Agile framework offers several reference models such as adaptive enterprise architecture management (AEAM) (see Fig.2) and AEPM reference models (see Fig.3). The focus of the paper is AEPM, and thus it is briefly explained in this section. The AEPM can be tailored and used for a situation at the portfolio, program, project, release and iteration levels for scaling agility at different levels. The tailored capability is operated for developing and managing project(s) in small releases and iterations. Thus, AEPM describes Agile or adaptive planning, analysis, architecture, design, implementation; testing and deployment services at different levels, from portfolio to iteration level (see Fig. 3).

Figure 1: The Gill Framework® overview (with permission from A.Q. Gill)

Portfolio management:

The teaching was managed using the portfolio management approach using the AEPM capability reference model (see Fig. 3). The teaching subject has several workshops and each workshop has a number of projects and all the projects are

managed as a portfolio of student projects. Thus, the subject coordinator performed the role of the portfolio manager to manage the overall teaching and provided the required guidance to understand the agility at the large portfolio scale. The subject runs for 12 weeks in spring semester. The subject coordinator assigns tutors (managers) to each workshop.

Program management:
Tutors of each workshop are the program managers of the students' projects in their individual workshop or tutorial. Students are organised into a group of 5 or 6 with the help of the program manager. Groups are given the choice of project idea for their project work. Further, each group needs to form a virtual start-up company to deliver the software project, this is an important consideration to bring entrepreneurship thinking and practices into academic teaching.

Project Management:
All the student projects are independent and can be developed using a selection of software technologies (Web app, IoT app, Mobile app, Desktop app using Java, JSP, Python, HTML, JS, Node.JS, Angular.JS, REACT, RUBY, C#, ASP.NET, etc.). Student group needs to self-organise as a virtual start-up company and appoint a student project manager to coordinate the group project activities using the Agile practices (e.g. Scrum).

Release Management:
Groups are expected to deliver the software using the Agile release management practice. Students were required to only deliver release 0 (R0) and release 1 (R1). R0 deliverable is a documented software prototype based on the project proposal. R0 prototype is a good way to learn how to identify the project related risks and technical dependencies earlier in the project before committing too many resources upfront. R1 deliverable is full-working software. Student group needs to appoint a release manager or scrum master (one of the students to take on this role) to manage the delivery of releases.

Iteration Management:
Iterations are core to Agile-DevOps process. Each release has 4 iterations (I0-I3). I0 is about setting up the environment to initiate the release in hand. Iteration may span over a week or two. Students are expected to apply Agile practices for analysis, planning, architecture and design at the iteration level. DevOps automation and continuous integration (DRA) concepts vertically augment the Agile team collaboration, testing, deployment and delivery (see highlighted DevOps part in AEPM for adaptive iteration implementation in Fig. 3). DRA was taught to students which enabled them to apply automated deployment, continuous integration, automated testing (acceptance and unit) and real-time monitoring practices to the delivery or release of a software product. For instance, R1 software application is expected to be hosted on a server or cloud. The project database is also expected to be hosted on a server or cloud (SQL, or NoSQL). DRA model is used as guidance for students during R1. Students were guided to apply DevOps practices within the DevOps part of the AEPM capability reference model (see Fig. 3). This indicates the vertical integration and

relationships between the AEPM (Agile) and DRA (DevOps) in iterations within in the release.

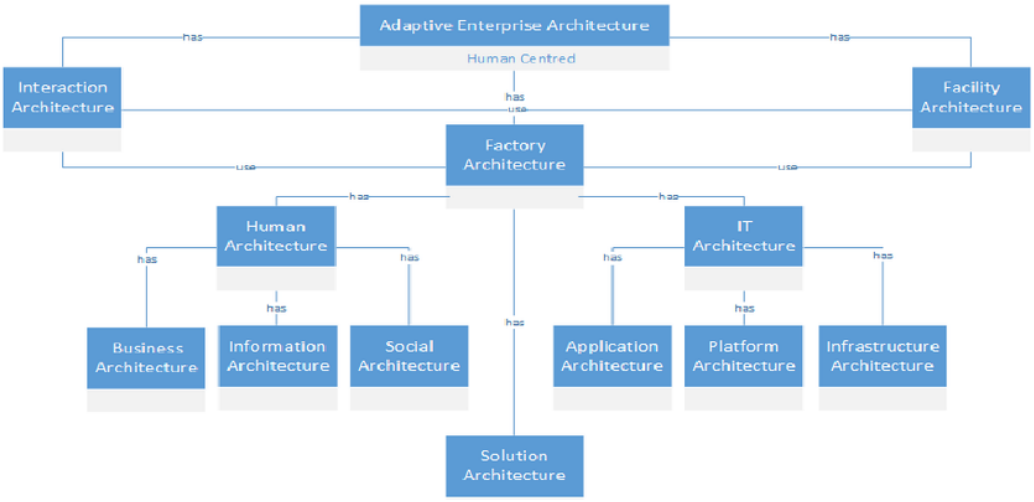

Figure 2: The Gill Framework® – Architecture Domains (with permission from A.Q. Gill)

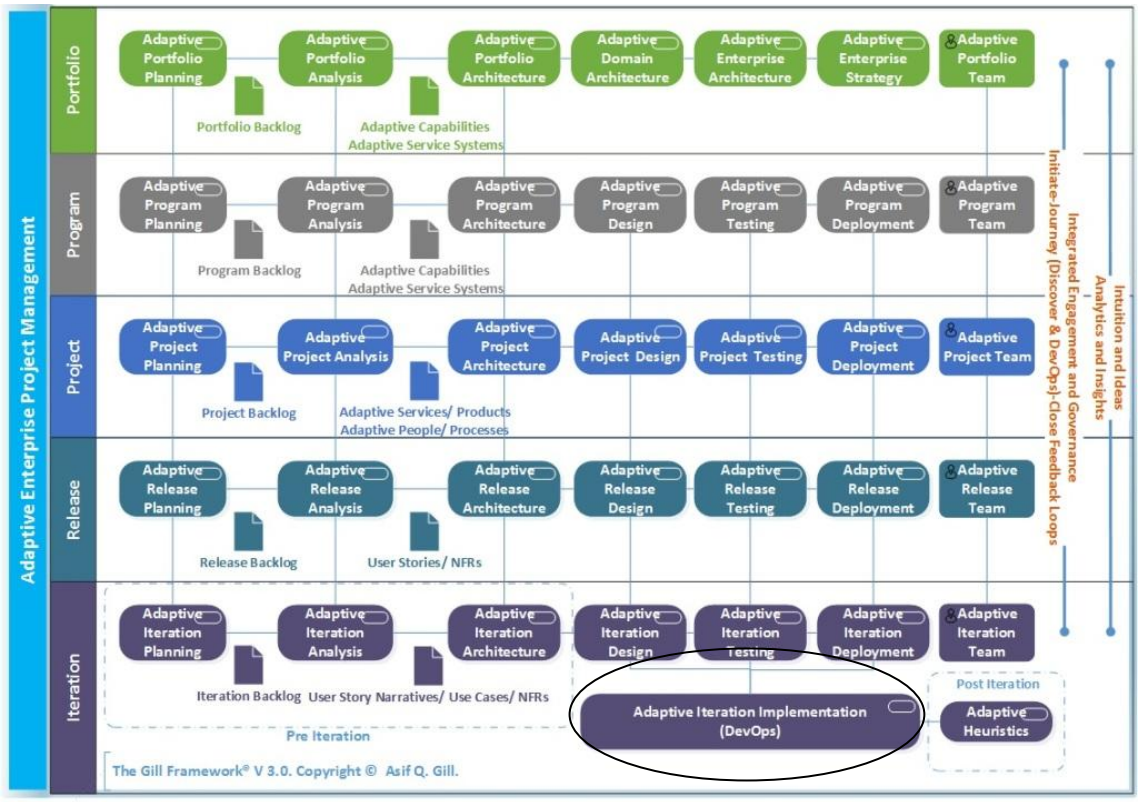

Figure 3: The Gill Framework® - AEPM Reference Model (with permission from A.Q. Gill)

4. The DRA Model

The DRA (see Fig. 4) has three key components: DevOps, Multi-Cloud and IoT. The concept of DRA design is to create a model that enables automation, continuous integration, and real-time monitoring for software application deployment in cloud including IoT applications. The DRA was used to teach DevOps concepts, practices and tools to students for developing and deploying their projects. The DRA context supports IoT applications, however the DRA architecture is programming language independent and may deploy scale and deliver any type of software application. The DRA context is expanded to a conceptual model (Fig. 5) that supports human architecture and IT architecture (Fig. 2). The combination of both architectures leads to architecture solution (Fig. 2) represented by DRA operation model instance (Fig. 7).

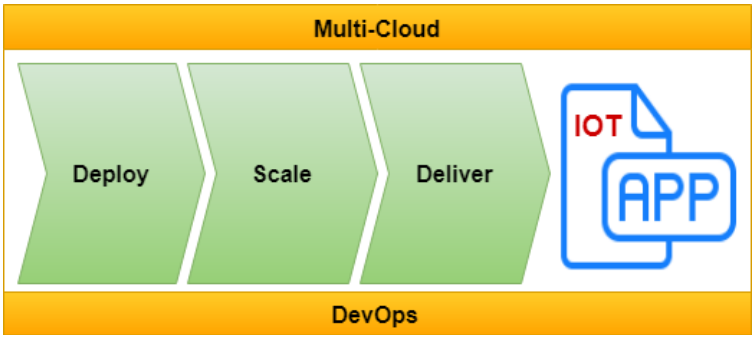

Figure 4: The DRA – Contextual Model

DRA Conceptual Model Architecture:

The Conceptual Model (see Fig. 5) describes the DRA components details at conceptual level. It presents the It highlights key DevOps practices for students such as integration, automation, collaboration etc. The model also highlights the cloud and multi-cloud and its linking to DevOps. For instance, it highlights the need for a continuous integration broker tools (CI-Broker) to distribute the software application to multi-cloud environments after it had been automatically tested. The benefits of this are that the deployment step was shifted outside the multi-cloud and solely managed by CI-Broker. Also it means that multi-clouds are only used for PaaS (platform as a service) to execute and scale the software application. DRA conceptual model laid the foundation for a Logical Architecture composed of 5 models (see Fig. 6).

DRA Logical Model Architecture:

DRA Logical Architecture is composed of five models (see Fig. 6). The models are integrated at later stages with AEPM to function as a deployment code engine. The models assist Agile software development with critical requirements such as: automated synchronization of software code, automated testing (unit and acceptance tests), automated build for project container, automated distribution of software application to multi-cloud, automated deployment on multiple clouds, automated log capturing, and automated reporting to Agile-DevOps team.

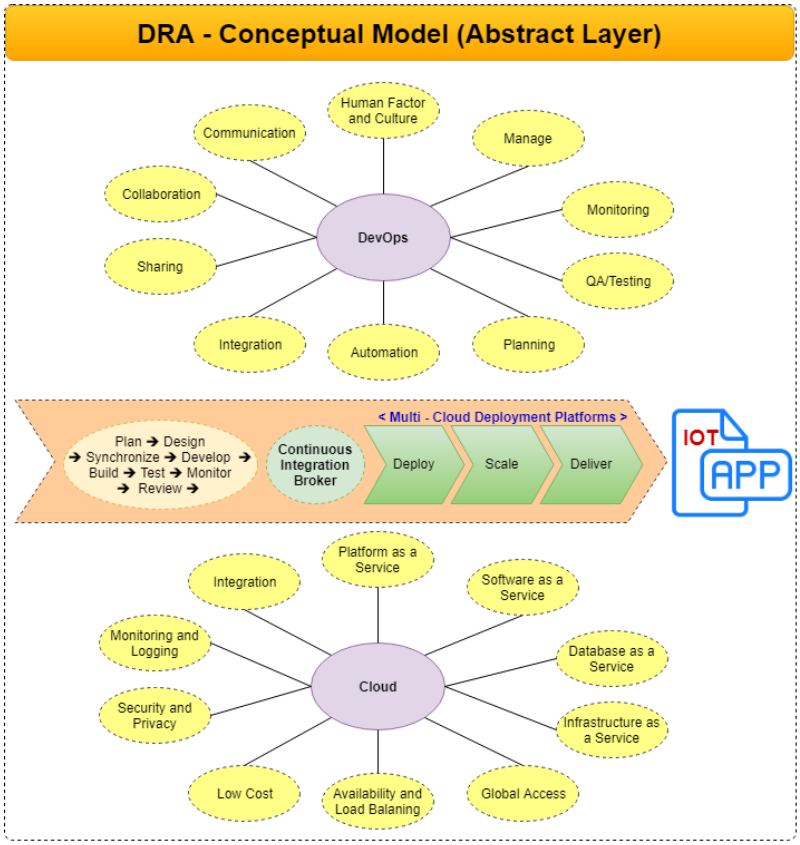

Figure 5: The DRA – Conceptual Model

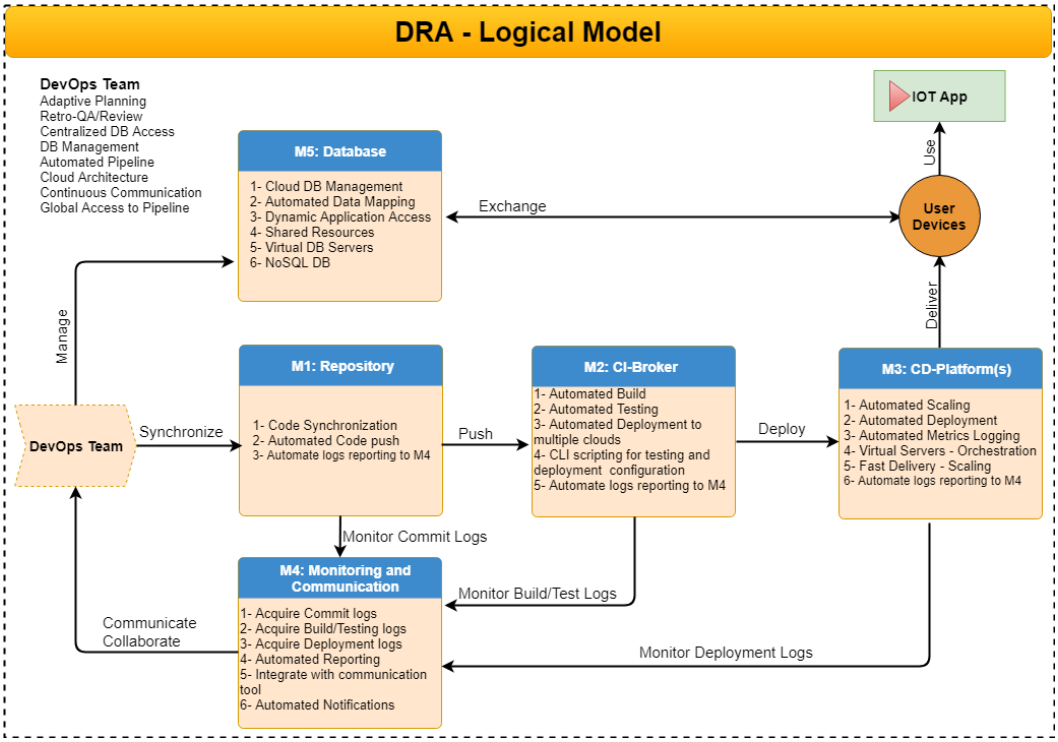

Figure 6: The DRA –Logical Model

DRA Logical model has 5 models, which are discussed in Table 1. Each model has certain features and supporting tools. DRA Logical Model features are derived from DevOps practices and realized using DevOps tools. Students participating in the SEP subject learn how to implement the features mapped in Table 1 and apply DevOps practices by integrating DevOps tools to create an operational pipeline (Fig 7).

Table 1: The DRA Logical Models

Model	DRA Models	
	Features	***Tools***
M1	1- Code Synchronization 2- Automated Code push 3- Automate logs reporting to M4	1- Github 2- BitBucket
M2	1- Automated Build 2- Automated Testing 3- Automated Deployment Multi-Cloud 4- CLI scripting for Testing/Deployment 5- Automate logs reporting to M4	1- Codeship 2- Travis-CI 3- Jenkins
M3	1- Automated Scaling 2- Automated Deployment 3- Virtual Servers - Orchestration 4- Fast Delivery – Staging 5- Automate logs reporting to M4	1- Heroku 2- Google App Engine 3- AWS CodeDeploy
M4	1- Acquire Commit logs 2- Acquire Build/Testing logs 3- Acquire Deployment logs 4- Automated Reporting 5- Integrate with communication tool 6- Automated Notifications	1- Papertrail 2- Nagios 3- New Relic 4- HipChat 5- Slack
M5	1- Cloud DB Management 2- Automated Data Mapping 3- Dynamic Application Access 4- Shared Resources 5- Virtual DB Servers 6- NoSQL DB	1- MongoDB (mLab) 2- DBMaestro 3- Firebase

DRA Pipeline Instance (for IoT):

The DRA Logical model is the basis to create pipelines for software application deployment. (see Table 1, Fig. 6). DRA Logical model highlights a range of DevOps tools to support the software project teams. DRA can be re-configured to deploy different types of software applications (including IoT). DRA pipeline (based on model M5) provides an external cloud database (MongoDB in this example). Also DRA pipeline (based on model M2) provide external CI-Broker (in this example Codeship). The mentioned elements in the pipeline are effective key-factors that enable developers and practitioners to avoid multi-clouds vendor lock-in. For the purpose of a practical demonstration of the DRA to

students and also for the proof of concept, we provided a video (YouTube) showing the deployment of an IoT-app in DRA instance pipeline (Fig. 7). DRA demo: https://youtu.be/JN38xS27ek0.

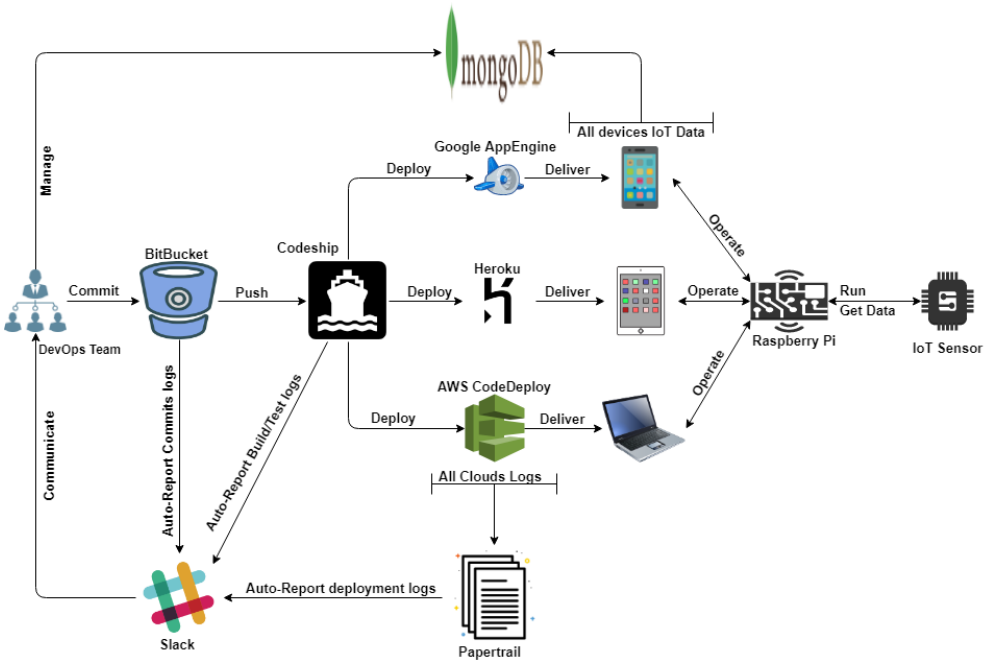

Figure 7: The DRA - Pipeline Instance for IoT Application

5. Dual Integrated Agile-DevOps Architecture

Agile enterprise project management (e.g. AEPM) requires automation and frequent fast software application release. The AEPM provides an overall guidance to management the large scale enterprise software delivery at the portfolio, program, project, release and iteration levels (See section 2 and 3). The DRA offers the DevOps practices and tools to support the DevOps concept of the AEPM at the iteration level. This integration between the DRA and AEPM is shown in Fig 8. The DRA as an extension and integration to AEPM is aimed to provide concrete directions to development and operations teams to effectively integrate and apply DevOps practices and tools as appropriate to their context. The key-integration factors are mapped into Table 2. The cross-architecture integration matrix shows the key elements of the AEPM reference model elements and the support features of the DRA models (see Table 2). The integrated AEPM and DRA elements were used as teaching instruments in the software engineering teaching workshops to enable students to clearly understand and apply the integrated Agile-DevOps approach to their team projects. The AEPM and DRA made it simple and clearer for students on how to effectively use DevOps for enterprise scale Agile software engineering.

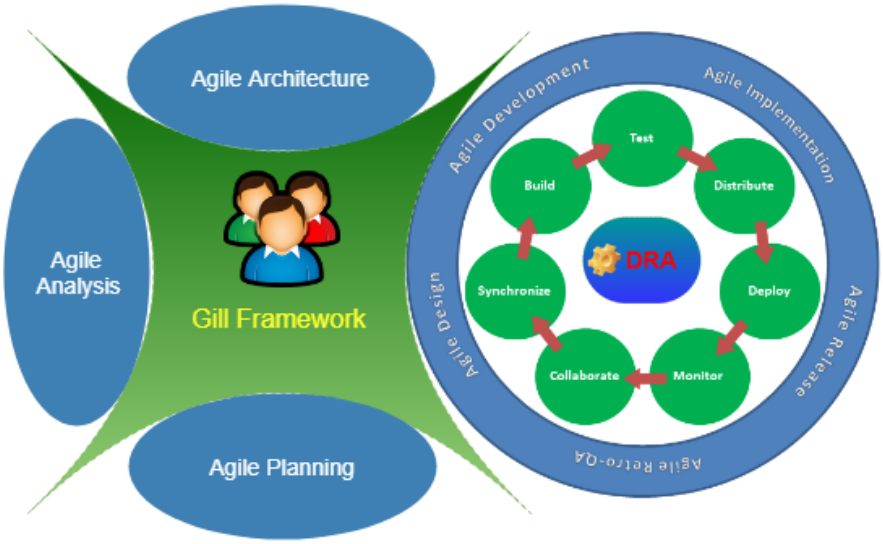

Figure 8: The Gill Framework AEPM (Agile) and DRA (DevOps) Integration

Table 2: The AEPM and DRA Integration

The AEPM Reference Model	The DRA Support Features	DRA model
Agile Analysis	1- Communication 2- Logging 3- Reporting 4- Collaboration	M1 and M4
Agile Planning	1- Communication 2- Logging 3- Reporting 4- Collaboration	M1 and M4
Agile Architecture	1- Communication 2- Logging 3- Reporting 4- Collaboration	M1 and M4
Agile Design	1- Communication 2- Logging 3- Reporting 4- Collaboration	M1 and M4
Agile Implementation	1- Communication 2- Collaboration 3- Automated Build 4- Automated Monitoring 5- Automated Deployment 6- Cloud database 7- Real-time Monitoring 8- Automated logging	M1, M2, M3, M4, and M5
Agile Testing	1- Communication 2- Collaboration 3- Automated Testing 4- Automated Monitoring 5- Cloud database 6- Real-time Monitoring 7- Automated logging	M1, M2, M4, M5

The AEPM Reference Model	The DRA Support Features	DRA model
Agile Deployment	1- Communication 2- Collaboration 3- Automated Build 4- Automated Monitoring 5- Automated Deployment 6- Cloud database 7- Real-time Monitoring 8- Automated logging	M1, M3, M4, M5
Agile Product Release	1- Communication 2- Collaboration 3- Auto-scaling 4- Automated Monitoring 5- Automated Deployment 6- Continuous Distribution 7- Real-time Monitoring 8- Automated logging 9- Fast Delivery	M1, M3, M4
Agile Quality Assurance	1- Communication 2- Collaboration 3- Automated Testing 4- Automated Monitoring 5- Real-time Monitoring 6- Automated logging	M1, M2, M4
Agile Team Collaboration	1- Communication 2- Collaboration 3- Automated Monitoring 4- Real-time Monitoring	M1, M4
Agile Retrospective	1- Communication 2- Collaboration 3- Automated Monitoring 4- Real-time Monitoring 5- Automated Testing	M1, M2, M4

6. The Agile-DRA for Teaching – A Case Study

In Section 3 of this paper, we discussed the program and project management. The release management in earlier section indicates that the SEP subject project has 2 releases (R0 and R1) and each release has 4 iterations (I0 to I3). R0 deliverable [week 1 to week 6] consists of Agile development process report, a software prototype, and initial setup of the DRA operational model instance (DRA pipeline Fig. 7). The initial DRA setup includes: 1) Source management and repository (GitHub or BitBucket); 2) Collaboration and communication tool (Slack); integration of (1) and (2). R1 deliverable [week 7 to week 12] consists of: 1) revised (updated report) Agile development process (user-stories, data dictionary, architecture, design); 2) full working software that satisfies the Agile requirements; and 3) DRAv2.0 pipeline for multi-cloud. The groups' cohorts configure their software delivery pipelines based on the logical model features explained in earlier section (Fig. 6 and Table 1). Table 2 shows DRA logical (Table 1) features that support Agile development process (planning, analysis, architecture, design, implementation, testing, deployment and delivery); and

enable the automation of development steps and faster application delivery to multiple clouds. Fig. 8 is an abstract illustration that expresses how DRA core concepts integrate with the Agile (The Gill Framework) process. Students participating in the course benefit from learning the principles of Agile development process and learn how to adopt integrated Agile-DevOps approach in a software development project.

7. The Teaching Case Study Evaluation

The case study discussed in this paper was evaluated using a student feedback survey (SFS) conducted online between 09/10/2017 and 12/11/2017. The SFS is an anonymous survey (survey No. 200063) that allows students (total 203 enrolled in spring 2017) to input ratings based on a scale composed of 5 possible entries. The SFS scale is: SD (strongly disagree); D (disagree); N (neither agree or disagree); A (agree); SA (strongly agree). The scale answers are used for the SFS questionnaires (10 questions) (see example of some questions and student feedback in Fig 9 and Fig 10). The survey also offered open questions section to students (Fig 10). The open question section allows students to freely express their opinion and give feedback about the quality of the subject materials and staff. The responses and feedback to open question section indicate that the students are overall satisfied with the subject materials, management and teaching staff.

The students' responses are computed to create Mean values for each question (a decimal between 0 and 5). Table 3 shows the mean values distribution for each question and calculates the average Mean value of the survey. The average Mean value (AMV) of the survey is AMV = 3.82 out of 5 (76.38%). The AMV indicates the subject has an acceptable level of success (above 75%) overall. This also shows that the integration of DRA (DevOps based framework) and The Gill Framework was a successful attempt to teach integrated Agile-DevOps development in academic settings.

Table 3: SFS Mean Distribution

Question	Mean	Percentage
Q1	3.85	77.00%
Q2	4.00	80.00%
Q3	3.66	73.20%
Q4	3.93	78.60%
Q5	3.84	76.80%
Q6	3.87	77.40%
Q7	3.80	76.00%
Q8	3.69	73.80%
Q9	3.67	73.40%
Q10	3.88	77.60%
AMV =	**3.82**	**76.38%**

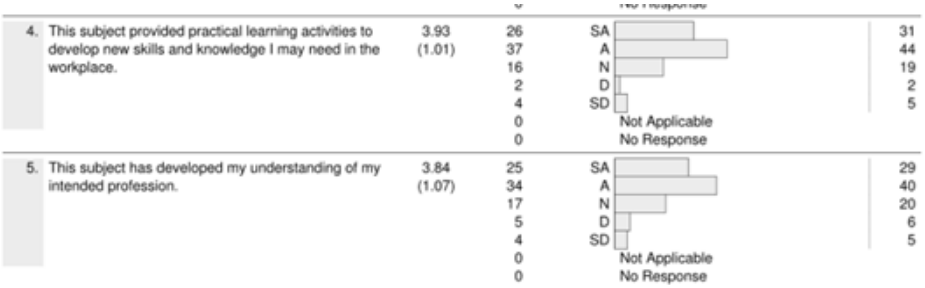

Figure 9: SFS Rating Example

Figure 10: The SFS Feedback Example

8. Discussion

This section discusses the key learnings from the teaching of integrated AEPM and DRA to software engineering undergraduate students.

- Firstly, it is important to train the trainer or use experienced tutors before actually teaching the complex concepts and integration of Agile-DevOps to students.

- Secondly, we used an iterative and collaborative teaching approach, which required us to explain the theory in the beginning of the semester (first 3 workshop sessions). Remaining 9 sessions were focused on actually using the concepts and tools of Agile-DevOps and applying it to student projects within the overall context of agile portfolio and program management.

- Thirdly, the role of the tutor was much more than just a teacher. They played the role of a program manager along with the subject coordinator (portfolio manager) who managed the overall learning instead of just teaching. Thus, the focused shifted from traditional teaching to more engaged practice oriented learning.

- Fourthly, students were required to self-organise, form a virtual start-up company and nominate the project manager and scrum master. The role of the project manager was to coordinate the student group project activities or interactions (Solheim 2019) and serve as a single point of contact for the program manager such as the tutor. Further, to internally manage the technical deliverables of the software they were required to appoint the scrum master or technical lead. This helped the students to get near real industry experience.

- Fifthly, in order to facilitate the outside class learning, pre-workshop and post-workshop material including videos and quizzes were posted to enhance learning experience beyond classroom. This provided the students with the flexibility to learn in their own time and prepare before coming into the schedule collaborative learning workshops.

- Sixthly, the students' feedback results for spring 2016 and spring 2017 semesters was positive and optimistic. Students conveyed that Agile-DevOps integration enabled optimal and efficient workshop environment which reflected industry experience.

In summary, both the AEPM and DRA reference models guided the teachers and students to effectively setup the Agile-DevOps environment, and then plan, deliver and manage the learning of complex Agile-DevOps concepts, principles, practices and tools during a single semester. It is also important to note here that several well-known research databases were used (e.g. IEEE, ACM, AIS (eLibrary) and Google Scholar) for finding the relevant literature for this paper.

9. Conclusion

This paper demonstrates how to teach complex Agile and DevOps concepts, principles, practice and tools to students in a University environment using the reference models such as the AEPM and DRA as a guide. This has been clearly demonstrated in this paper using a teaching case study example. The teaching case study provided several insights, in particular, the use of the reference models, role of the subject coordinator, tutors and students in the overall collaborative learning process. Subject coordinator used the reference models and played the role of the industry level portfolio manager. Tutors played the role of a program manager for the student projects in their workshops. Students played the roles of project manager, scrum master, business analyst, developer, tester and DevOps engineer. This helped the students to understand the industry roles and required skills. This is an attempt to provide industry like experience and orientation to students to make them ready for the job. The learnings from this paper can be used by practitioners and academics to effectively develop and improve human capability in the area of Agile and DevOps. Hence, this paper is expected to help researchers, academics and practitioners seeking to further research the enterprise scale Agile-DevOps software engineering.

References

Artac, M., Borovšak, T., Di Nitto, E., Guerriero, M. and Tamburri, D.A. (2016). Model-Driven Continuous Deployment for Quality DevOps. Proceedings of the 2nd International Workshop on Quality-Aware DevOps. doi:10.1145/2945408.2945417.

Alzoubi, Y. I. and Gill, A.Q. (2014). Agile Global Software Development Communication Challenges: A Systematic Review. PACIS 2014.

Alzoubi, Y.I, Gill, A. Q. Al-Ani, A. (2015). DistributedAgile Development Communication: An Agile Architecture Driven Framework. *Journal of Software*. 681-694. doi:10.17706/jsw.10.6.

Alzoubi, Y.I., Gill, A. Q. and Moulton, B.(2018). A measurement model to analyze the effect of Agile enterprise architecture on geographically distributed Agile development. Journal of software engineering research and development, Springer. doi:10.1186/s40411-018-0048-2.

Bai, X. Li, M., Pei, D., Li, S. and Ye, D. (2018). Continuous Delivery of Personalized Assessment and Feedback in Agile Software Engineering Projects. 2018 ACM/IEEE 40th International Conference on Software Engineering: Software Engineering Education and Training.doi: 10.1145/3183377.3183387.

Bou Ghantous, G. and Gill, A. (2017). DevOps: Concepts, Practices, Tools, Benefits and Challenges. 21st PACIS 2017. http://aisel.aisnet.org/pacis2017/96.

Bou Ghantous, G. and Gill, A. (2018). DevOps Reference Architecture for Multi-Cloud IOT Applications. 20th IEEE International Conference on Business Informatics CBI2018 Vienna Austria.doi:10.1109/CBI.2018.00026.

Colavita, F. (2016). DevOps Movement of Enterprise Agile Breakdown Silos, Create Collaboration, Increase Quality, and Application Speed. Proceedings of 4th International Conferencein Software Engineering for Defence Applications, Advances in Intelligent Systems and Computing 422. doi:10.1007/978-3-319-27896-4_17.

Garfield, J. (1993) Teaching Statistics Using Small-Group Cooperative Learning, J*ournal of Statistics Education*. doi:10.1080/10691898.1993.11910455.

Gill, A. Q. (2014). Applying agility and living service systems thinking to enterprise architecture. *International Journal of Intelligent Information Technologies (IJIIT)*, 10(1), 1-15.doi:10.4018/ijiit.2014010101.

Gill, A. Q. (2015). Learning Enterprise Agile Software Engineering. IEEE ASWEC 2015. doi:10.1109/.26.

Gill, A.Q. (2015b). Adaptive Cloud Enterprise Architecture, World Scientific.

Gill, A. Q., Henderson-Sellers, B., & Niazi, M. (2018). Scaling for agility: A reference model for hybrid traditional-agile software development methodologies. *Information Systems Frontiers*, 20(2), 315-341.doi: 10.1007/s10796-016-9672-8.

Gill, A.Q., Loumish, A., Riyat, I. and Han, S. (2017). DevOps for information management systems. *VINE Journal of Information and Knowledge Management Systems*, 48 (1), 122-139. doi:10.1108/VJIKMS-02-2017-0007.

Lwakatare, L. E., Kuvaja, P., and Oivo, M. (2016). Relationship of DevOps to Agile, Lean and Continuous Deployment A Multivocal Literature Review Study. PROFES 2016, Springer LNCS 10027, 399–415.doi:10.1007/978-3-319-49094-6_27.

Mason, R.T., Masters, W. and Stark, A. (2017). Teaching Agile Development with DevOps in a Software Engineering and Database Technologies Practicum. 3rd International Conference on Higher Education Advances, HEAd'17. doi:10.4995/HEAd17.2017.5607,2017.

Mohamed, S.I. (2016). DevOps Maturity Calculator DOMC - Value oriented approach. *International Journal of Engineering Research & Science* (IJOER) ISSN: [2395-6992], 2(2).

Neve, J. R., Godbole, K., Neve, R. (2017). Productivity And Process Improvement Using 'Scaled Agile' Approaches: An Emphasized Analysis. Proceedings of the International Conference on Inventive Computing and Informatics (ICICI 2017). doi:10.1109/ICICI.2017.8365245.

Perera, P., Silva, R., and Perera, I. (2017). Improve Software Quality through Practicing DevOps. IEEE 2017 International Conference on Advances in ICT for Emerging Regions (ICTer). doi:10.1109/ICTER.2017.8257807.

Qumer, A., and Henderson-Sellers, B. McBride, T. (2007). Agile adoption and improvement model. Proceedings European and Mediterranean Conference on Information Systems 2007 (EMCIS2007) June 24-26 2007, Polytechnic University of Valencia, Spain.

Snyder, B., Mae, F. and Curtis, B. (2017). Using Analytics to Guide Improvement during an Agile–DevOps Transformation. IEEE Software, IEEE Computer Society.doi:10.1109/MS.2017.4541032.

Solheim, K. (2019). Teachers' Aspirations to Improve their Classroom Interaction. International Journal of Learning, Teaching and Educational Research. 18(6), 147-169. doi:10.26803/ijlter.18.6.9.

Wang, C. and Liu, C. (2018). Adopting DevOps in Agile: Challenges and Solutions. Independent thesis Advanced leve. Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden.