Faculty of Science, Technology and Communication

University of Luxembourg
2, avenue de l'Université,
L-4365 Esch-sur-Alzette
Luxembourg

DevOps and its Philosophy: Education Matters!

Evgeny Bobrov[1], Antonio Bucchiarone[3], Alfredo Capozucca[2], Nicolas Guelfi[2], Manuel Mazzara[1], Alexandr Naumchev[1], Larisa Safina[1]

[1] Innopolis University, Russian Federation
[2] University of Luxembourg
[3] Fondazione Bruno Kessler, Trento, Italy

TR_LASSY_19_02
Technical Report
April 2019

DevOps and its Philosophy : Education Matters!

Evgeny Bobrov, Antonio Bucchiarone, Alfredo Capozucca, Nicolas Guelfi,
Manuel Mazzara, Alexandr Naumchev, Larisa Safina

Abstract DevOps processes comply with principles and offer practices with main objective to support efficiently the evolution of IT systems. To be efficient a DevOps process relies on a set of integrated tools. DevOps is the first required competency together with Agile Method required by the industry. DevOps processes are sharing many aspects with microservices approaches especially the modularity and flexibility which enables continuous change and delivery. As a new approach it is necessary to developp and offer to the academy and to the industry training programs to prepare our engineers in the best possible way. In this chapter we present the main aspects of the educational effort made in the recent years to educate to the concepts and values of the DevOps philosophy. This includes principles, practices, tools and architectures, primarily the Microservice architectural style. Two experiences have been made, one at academic level as a master program course and the other, as an industrial training. Based on those two experiences, we provide a comparative analysis and some proposals in order to develop and improve DevOps education for the future.

1 Introduction

In a world of rapid technological development and full automation trend, *technological progress* is often identified in a new model of a digital device or a new release of a software package. In the full spectrum of technological progress, there are de-

Antonio Bucchiarone
Fondazione Bruno Kessler, Trento, Italy e-mail: bucchiarone@fbk.eu

Evgeny Bobrov, Manuel Mazzara, Alexandr Naumchev, Larisa Safina
Innopolis University, Russian Federation e-mail: {e.bobrov,m.mazzara,a.naumchev,l.safina}@innopolis.ru

Alfredo Capozucca, Nicolas Guelfi
University of Luxembourg e-mail: {alfredo.capozucca,nicolas.guelfi}@uni.lu

velopments that cannot be immediately perceived or purchased by the final user, among them *process innovation*. In a competitive world, and in order to offer better prices, companies have the need to optimize their operations [6]. Innovative business models appear everywhere from the game industry to the mobile application domain, and the distinction between what is Information Technology and what is not becomes less and less obvious. Is Uber a taxi or an IT company? Is Airbnb a realtor? Software development techniques and operations need also to catch up with this trend and with the new business context.

Until now, it was clear when the next release of Windows would have appeared, but what about a web service (e.g., Google, Yandex search)? Agile Methods deal with this problem only from the software development point of view focusing on customer value and managing volatile requirements. However, the current effective scenario practice in industry nowadays requires much more than that, and involves the entire life cycle of a software system, including its operation. Thus, it is not surprising that today the industry is desperate looking for qualified people with competences about DevOps[3].

DevOps is a natural evolution of the Agile approaches [5, 16] from the software itself to the overall infrastructure and operations. This evolution was made possible by the spread of cloud-based technologies and the everything-as-a-service approaches. Adopting DevOps is however more complex than adopting Agile [1] since changes at organisation level are required. Furthermore, a complete new skill set has to be developed in the teams [7]. The educational process is therefore of major importance for students, developers and managers. As long as DevOps became a widespread philosophy, the necessity of education in the field become more and more important, both from the technical and organisational point of view [7].

Chapter Outline and Contribution. This chapter describes parallel experiences of teaching DevOps both undergraduate and graduate students at the university, and junior professional developers with their management in industry. We proceed to a comparative analysis to identify similarities, differences and how these experiences can benefit from each other. After this introduction, Section 2 discusses the experience of teaching DevOps in a university context while Section 3 reports on the industrial sessions we have been delivering. Comparative analysis, conclusions and future work on education are summed up in Section 4.

2 Teaching in Academia

The success of software development project is very often (to no say always) aligned with the skills of the development team. This means that having a skillful team is not only a pre-requisite to have a chance to be successful in the software industry, but also to adopt new techniques aimed at simplifying the burden associated to the production of software in current times: i.e. remain competitive by continuously optimizing the production process. It is acknowledged that agile methods and DevOps principles and practices are nowadays among the most relevant new tech-

niques wished to be fully mastered by team members. Therefore, we, as part of the academia are responsible of forming students with a set of skills able to cope not only with today's needs, but also those of tomorrow.

For this purpose, we have rencently developped a new DevOps course offered at master level in our academic institution [8]. Despite of the fact that the course is part of an academic programme in computer sciences, it has been designed to make a pragmatic presentation of the addressed topics. This is achieved by applying the Problem-based learning (PBL) method as pedagogical approach. Thus, lectures are interleaved with hands-on, practical sessions and stand-up meetings where students (working in groups) along with guidance of the teaching staff, work out the solution to the assigned problem. This problem, common to every groups consists of the implementation of a deployment pipeline. The objective then it is not only engineer the deployment pipeline and demonstrate its functioning, but also justify the choices made in its design. Each group relies on a software product of its choice to demonstrate the functioning of the pipeline. Thus, it is the chosen product what makes each group's work unique.

2.1 Experience

Here we summarize the relevant information about the course (i.e. organisation, structure, execution, and assessment), along with the lessons learnt and some reflections to be considered into its following editions.

The PBL method allows students to focus on finding one (of the many possible) solutions for the given complex problem. The problem to be solved consists of implementing a deployment pipeline, which needs to satisfy certain functional and non-functional requirements. As students work in groups to find out the expected solution, they will experiment in first person the problems arisen in collaborative environments. The creation of such as environments is intentionally done to let students either acquire or improve (with the guidance of the teaching staff) the required soft-skills capacities need to deal with people and processes-related issues. Notice that it is acknowledged that DevOps culture is aimed at increasing inter and intra team's collaboration. Therefore, soft-skills capabilities are as important as operational tools meant for automation.

The knowledge is transferred to students through lectures, project follow-up sessions (kind of stand-up meetings) aimed at having a close monitoring of the work done for each group member and helping solve any encountered impediments, and assessment talks (where each group presents the advances regarding the projects objectives). This structure favours both the cohesion of groups and the exchange of ideas among every course participant. The topics presented during the lectures are those closely related to the project's goal. They are configuration, build, test, and deployment management. Such as topics are presented in the mentioned order, and after a brief general introduction to the DevOps movement. It is worth mentioning that the course opens with a high frequency of lecture sessions, but soon they leave

place to hands-on and stand-up meetings. Thus, a relevant time of the course is spent in learning practices and discussing the different alternatives to achieve the targeted solution. It is during this kind of sessions that groups soon realize the impact of the product[1] on the deployment pipeline aimed at supporting its development. It is worth remembering that one of the objectives in setting up a deployment pipeline (and of DevOps in general) is to reduce the time since a modification made by a developer is committed and pushed into the main branch of a repository until it appears in production ready to be used by the end-user, but without scarifying quality (the development team wants to have certain certainty that the modification would work as required without introducing flaws into the product). Therefore, for a product that belongs to those known as *monolithic* the deployment pipeline's throughput would be higher than for those architected according the microservices style. Notice that we (teaching staff) also advise in the selection of the product to be used during the execution of the course, despite of it is not part of the course's objectives to assess the quality of such product. The point in doing so it's twofold: first, to rise the concerns related to the constraints imposed by the product over its deployment pipeline; and second, to avoid groups struggle with technical issues out of the project's scope and which could lead to frustration, and eventually drop outs (although these risks are always present). Anyway, regardless the selected product, we drive students towards the implementation of the pipeline. This means that, eventually, they need to show us the functioning of such pipeline along with arguments that explain why such as pipeline is the most suitable for the product required to handle.

The experience until now has been very positive: students have provided good feedback about the course, no drop outs, and high quality of the project's outcomes. Feedback from students was gathered through a survey filled out anonymously once the course was over: 100% agreed on the statement *the course was well organized and ran smoothly*, 75% (25%) agreed (strongly agreed) on the statement *the technologies used in the course were interesting*, and 75% agreed with the statement *I am satisfied with the quality of the course*. Therefore, based on the obtained results, we can conclude that we have a good course baseline, which can be used to derive alternative variations of the course, depending on the context and attained learning outcomes. More about these alternatives, is explained in the following section.

2.2 Reflections

Definitively, the implementation of a deployment pipeline covers some of the DevOps aspects, but not all of them. However, we can argue that the backbone of DevOps is covered as the power of automation of testing, configuration, build and deployment tasks is clearly shown through the use of the pipeline as enabler to continuous product improvement. Thus, we are very happy with covering such DevOps aspects in a weekly course of 1.5 hs. lasting for 14 weeks. It is also important no to

[1] This product is chosen to demonstrate the functioning of the pipeline. Each group is requested to select an open source product for which there already exist implemented test cases.

forget that people-related aspects are also covered as, through the project, students need to perform in a collaborative manner developer and maintainer oriented tasks that have common concerns (e.g. develop provisioning scripts that need to be well structured and very configurable).

In case of more available time, then monitoring is a worth topic to be covered. This topic includes practices aimed at easing the detection of issues on the product once it has been released, but before them are noticed by end-users. Being able to incorporate such practices will let students understand how developers and maintainers can work together to define new requirements on the product meant to solve the issues detected through product monitoring.

Yet another alternative could be to move the focus on the product rather than the deployment pipeline when the attained objective is to make emphasis on the microservices style. In this case, both a deployment pipeline and a monolithic product are given at the beginning of the course, and then the project would be to refactor the product to adopt a microservices architecture. Performing such kind of project would make students aware of the important role played by the pipeline when doing refactoring. However, this idea has to be taken with caution as it addresses too many concerns at once. The most logical, it would be to make such kind of course as a continuation of the one described in the previous section.

3 Teaching in Industry

We have developed extensive experience in recent years in several domains of Software and Service Engineering, from service-related technologies and business processes [19, 25, 26] to workflows and their dynamic reconfiguration [17, 18] to formal methodologies for deriving specifications [20]. On top of this, we delivered training in corporate environments, both to a technical and a managerial audience, some time mixed. In particular, we had multiple interactions with east and west European companies operating in business domains going from banking to phone service provider and others [21]. In 2018 we have delivered more than 400 hours of training involving more than 500 employees in 4 international companies of mid to large size, employing more than 10k people.

The delivered sessions typically last one or two full days, that can be reiterated, at the premises of the customer and cover (as general topics):

- Agile methods and their application [1]
- DevOps philosophy, approach and tools [14]
- Microservices and applications [11, 12, 15, 23]

In order for the companies to effectively absorb the DevOps philosophy and its practice, the action has to focus not only on tools, but on people and processes too. The target group of the sessions is generally a team (or multiple teams combined) of developers and testers, often with the presence of mid-management. Before our

training we typically suggest customers to include also businesses and technical analysts, and when possible marketing and security departments representatives. These participants also benefit from participating to the training and from learnign the DevOps culture. The nature of the delivery depends on the target group: sessions for management focus more on effective team building and establishment of processes. When the audience is a technical team, the focus goes more on tools and effective collaboration within and across the teams.

For the purpose of this chapter we will summarize the experience with a particular company, an East-European phone service provider. Some details have to be omitted, but we describe the general structure of the training and some reflections. The detailed experience and some retrospective have been fully presented in [21].

3.1 Training Sessions

Here we describe our experience of training a team of developers of an East-European phone service provider which we have to keep anonymous. The training experience was structured in two sessions of two days each conducted in different weeks with a gap of about fifteen days. The first session was dedicated to the *Continuous Integration Delivery Pipeline*, and the second on*Agile methods*.

3.1.1 Session I: DevOps

The first session was conducted over two full days at the office of our customer. Our target group was a team of developers reporting to a line manager located in a different city (reachable only by flight). Due to circumstances related to company organization, previous direct communication with this specific team was not possible, and we could rely only on the information shared by the remote line manager. Therefore, the original agenda, communicated in advance to the team, had to be fine-tuned on site.

The training covered in detail the following four major topics. each representing a sub-session:

- Trends in IT and impact on software architectures and development
- Requirements volatility and Agile development
- Challenges of distributed development
- Microservices

This particular training emphasizing the difference between *hard technologies* and *soft technologies*. Hard technologies is the large-scale industrial production of commercial items of technological nature, while soft technologies is the continuous improvement and *agilization* of development process. Agile methods were discussed in terms of Requirements volatility. The final part covered distributed team development and *microservices*, which are considered the privileged architecture for

DevOps with their scalability features [12]. The key difference between monolithic service updates and microservice deployment was presented in order to motivate the need of migration to microservices.The audience therefore understood the vicinity between DevOps and microservices.

3.1.2 Session II: Agile

The second session was held for two full days at the same office than the first one. The objectives of the session according to plan were to cover *Agile software development*, in particular Scrum. On site the customer required to move the focus to Kanban, which appeared to be something that could be useful in the future. At some point it become obvious that the team itself did not have a clear mind on what actual process they intended to follow, therefore we started working on identifying a methodology that could work for their development teams. The framework described in *"Choose your weapon wisely"* [22] turned out to be useful. This document provides information on different popular development processes:

- Rational Unified Process (RUP) [4]
- Microsoft's Sync-and-Stabilize Process (MSS) [9]
- Team Software Process (TSP) [13]
- Extreme Programming (XP) [2]
- Scrum [24]

Following this approach the information about processes was delivered according to four blocks:

1. *Overview*: short description of the process
2. *Roles*: information about positions for the process
3. *Artifacts*: to be produced, including documentation
4. *Tool support*: tools available on the market for using the process

3.2 Lessons Learnt: Who Should Attend the Sessions

Here we will summarize some reflections that we derived from our professional experience. In retrospective, the most effective training were those in which the audience consisted of a mix of management and developers. The biggest challenges our customers encountered typically were not related to how to automatise the existing processes, but in fact how to set up from scratch the DevOps approach itself. Generally, technical people understand how to set up the automatization, but they may have only a partial understanding about the importance and the benefits for the whole company, for other departments, for the customer and ultimately for themselves. During training session it is therefore important to show the bigger picture and help them understanding how their work affects other groups, and how this in turn affects themselves in a feedback loop. The presence of management is very

useful in this cases, while the technical perspective can be often left for self-study or additional future sessions.

4 Comparative analysis, conclusions and future work

The last few years of experience on the field of DevOps education helped us in understanding the key aspects, and what are the differences between an academic and an industrial context. In this section we summarize our understanding of these two realities in order to help offering a better pedagogical programme in the future. Each of the two domains can indeed be cross-fertilised by the ideas taken by the other. The lessons that we have learnt in the Devops education can certainly be extended to other field, however we do not cover any generalization within the boundaries of this chapter.

The shared aspects between the two domains can be synthesized as follows:

- **Relevance**: the DevOps topics raises interests both in academia and industry. It is very actual and relevant.
- **Practice**: theory is always welcome, however students and developers mostly appreciate hands-on sessions that should not be forgotten in the educational process.
- **Dev vs. Ops**: the classic academic education and developers training typically dedicate more time and put more emphasis on development than operations. Sessions which present both as related and interdependent can strengthen the understanding of the whole matter and increase efficacy of the delivery.

Given the described common aspects, certain features of the education process can and should be kept on the same line (for example pragmatism and synergy Dev with ops). However, the difference between the two domains and their objectives requires some attention. The major differences we identified can be categorized as follows:

- **Entry Knowledge**: details on the academic curriculum and specific syllabus allow a university teacher to make assumptions on the entry knowledge of the students. In a corporate environment is very difficult to have this complete information in advance. In these cases the audience can be composed by people with different profiles and backgrounds on which you know very little.
- **Incentives**: for students the major incentive is the grade, which could be linked to a scholarship. This is a very short-run handle. Developers have different incentives and can look more at the mid-run in order to improve their working conditions, not only financially. Managers typically have incentives in terms of cost savings, and should be able to see things in the longer-run.
- **Delivery mode**: an academic course can last fifteen weeks and having project and assignments from one week to the other. In a corporate environment everything has to be compressed in a few days, and there is hardly time to do anything between one day and the other. This requires and adaptation of the delivery.

- **Assessment**: at the university there is the classic exam-based system. In a corporate environment the audience is not required to be assesses at the end of the sessions, instead the success of the delivery can only be observed in the long period when it is clear whether the adopted practices bring benefit to the company or not.
- **Expectation:** generally, a corporate audience is more demanding. This is due to the level of experience on one side, and on the direct costs on the other. While students see the teacher as an authority, the corporate audience does not. This has to be taken into account before and during the delivery.

In terms of pedagogical innovation, the authors of this paper have experimented for long with novel approaches under different forms [10]. However, DevOps represents a newer and significant challenge. Despite of the fact current educational approaches in academia and industry show some similarities, they are indeed significantly different in terms of attitude of the learners, their expectation, delivery pace and measure of success. Similarities lay more on the perceived hype of the topic, its typical pragmatic and applicative nature, and the minor relevance that education classically reserves to "Operations". While similarities can help in defining a common content for the courses, the differences clearly suggest a completely different nature of the modalities of delivery.

Our current experience suggests some changes to the approach. For what concerns the **University teaching**, the idea is to reduce the emphasis on final grade and to insist on the cultural aspect. Probably the relevance of practical assignments should be increased and that of final exam decreased. The understood importance of hand-on sessions should also suggest changes in the delivery. The ultimate plan is to to build a Software Engineering curricula fully based on the DevOps philosophy. In future **Corporate training** is important to avoid to base everything on a university-like frontal session. As seen in our experience, customer's request can change even during the session itself, and the agenda should be kept open and flexible.

References

1. Agile and devops: Friends or foes? https://www.atlassian.com/agile/devops. Accessed: 2018-07-01.
2. Extreme programming: A gentle introduction. http://www.extremeprogramming.org/. Accessed: 2018-07-01.
3. Preventing the AI crisis: the AISE Academy proposal for Luxembourg. http://www.itone.lu/pdf/AISE-academy.pdf. Accessed: 2019-04-03.
4. Rational unified process: Overview. http://sce.uhcl.edu/helm/rationalunifiedprocess/. Accessed: 2018-07-01.
5. Len Bass, Ingo Weber, and Liming Zhu. *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 1st edition, 2015.
6. Antonio Bucchiarone, Nicola Dragoni, Schahram Dustdar, Stephan Thordal Larsen, and Manuel Mazzara. From monolithic to microservices: An experience report from the banking domain. *IEEE Software*, 35(3):50–55, 2018.

7. Ineta Bucena and Marite Kirikova. Simplifying the devops adoption process. In *Joint Proceedings of the BIR 2017 pre-BIR Forum, Workshops and Doctoral Consortium co-located with 16th International Conference on Perspectives in Business Informatics Research (BIR 2017), Copenhagen, Denmark, August 28 - 30, 2017.*, 2017.
8. Alfredo Capozucca, Nicolas Guelfi, and Benoît Ries. Design of a (yet another?) devops course. In *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment - First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers*, pages 1–18, 2018.
9. Michael A. Cusumano and Richard W. Selby. How microsoft builds software. *Commun. ACM*, 40(6):53–61, June 1997.
10. Daniel de Carvalho, Rasheed Hussain, Adil Khan, Mansur Khazeev, JooYong Lee, Sergey Masiagin, Manuel Mazzara, Ruslan Mustafin, Alexandr Naumchev, and Victor Rivera. Teaching programming and design-by-contract. In *21th International Conference on Interactive Collaborative Learning - ICL 2018, Kos, Greece*.
11. N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. Microservices: yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering*. Springer, 2017.
12. Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, and Larisa Safina. Microservices: How to make your application scale. In *Perspectives of System Informatics - 11th International Andrei P. Ershov Informatics Conference, PSI 2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers*, pages 95–104, 2017.
13. Watts Humphrey and James Over. *Introduction to the Team Software Process(Sm)*. Addison-Wesley Professional, first edition, 1999.
14. Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. What is devops?: A systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, XP '16 Workshops, pages 12:1–12:11, New York, NY, USA, 2016. ACM.
15. K. Khanda, D. Salikhov, K. Gusmanov, M. Mazzara, and N. Mavridis. Microservice-based iot for smart buildings. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 302–308, March 2017.
16. Gene Kim, Patrick Debois, John Willis, and Jez Humble. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016.
17. Dragoni Nicola Zhou Mu. Mazzara, Manuel. Dependable workflow reconfiguration in WS-BPEL. In *Proceedings of the 5th Nordic Workshop on Dependability and Security*, 2011.
18. M. Mazzara, F. Abouzaid, N. Dragoni, and A. Bhattacharyya. Toward design, modelling and analysis of dynamic workflow reconfigurations - A process algebra perspective. In *Web Services and Formal Methods - 8th International Workshop, WS-FM*, pages 64–78, 2011.
19. Manuel Mazzara. *Towards Abstractions for Web Services Composition*. PhD thesis, University of Bologna, 2006.
20. Manuel Mazzara. Deriving specifications of dependable systems: toward a method. *CoRR*, abs/1009.3911, 2010.
21. Manuel Mazzara, Alexandr Naumchev, Larisa Safina, Alberto Sillitti, and Konstantin Urysov. Teaching devops in corporate environments - an experience report. In *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment - First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers*, pages 100–111, 2018.
22. Justin Rockwood. Choose your weapon wisely. Carnegie Mellon University.
23. D. Salikhov, K. Khanda, K. Gusmanov, M. Mazzara, and N. Mavridis. Jolie good buildings: Internet of things for smart building infrastructure supporting concurrent apps utilizing distributed microservices. In *Proceedings of the 1st International conference on Convergent Cognitive Information Technologies*, pages 48–53, 2016.
24. Ken Schwaber and Jeff Sutherland. The Scrum guide, 2001.

25. Zhixian Yan, Emilia Cimpian, Michal Zaremba, and Manuel Mazzara. BPMO: semantic business process modeling and WSMO extension. In *2007 IEEE International Conference on Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA*, pages 1185–1186, 2007.
26. Zhixian Yan, Manuel Mazzara, Emilia Cimpian, and Alexander Urbanec. Business process modeling: Classifications and perspectives. In *Business Process and Services Computing: 1st International Working Conference on Business Process and Services Computing, BPSC 2007, September 25-26, 2007, Leipzig, Germany.*, page 222, 2007.