Software Engineering Education in the DevOps Era: Experiences and Recommendations

Nicolás Paez[1,2], **Carlos Fontela**[2]

[1]Departamento de Ciencia y Tecnología - Universidad Nacional de Tres de Febrero
Saenz Peña, Argentina

[2]Facultad de Ingeniería - Universidad de Buenos Aires
Buenos Aires, Argentina

nicopaez@computer.org, cfontela@fi.uba.ar

Abstract. *DevOps has reached the mainstream, in part, because it proposed a way to solve the last mile of the software delivery process. It brought to the industry concrete solutions to some concerns that software engineering and academia have traditionally ignored. The popularity and success of DevOps got the attention of the academia and some universities have started to include DevOps related content in their programs. After doing our own experience, we found out that introducing DevOps is not as simple as adding a new course. In fact, it requires a major redesign of the Software Engineering curriculum. In this article we share some key ideas and recommendations to include DevOps in Software Engineering programs.*

1. Introduction

The Software Engineering discipline has traditionally ignored what we call the ”last mile” of the delivery process. This ”last mile” includes how to get the software solution into the production environment and, once in the production environment, how to operate the software solution. There is no mention to these delivery and operational concerns in the Guide to the Software Engineering Body of Knowledge (SWEBoK) [Bourque and Fairley 2014]. The same happens with classical Software Engineering textbooks. That was acceptable in the past. After all, going into production was a very special situation that happened once in a while. But at a certain point, the business requirements began to change more frequently, forcing deliveries to become more frequently. As a result, the ”last mile” of the process became more important. These business needs, along with the lack of formal knowledge and guidance, lead to the raise of the DevOps movement. DevOps brought concrete solutions to industry problems and it did not take long for it to go mainstream. DevOps grew in the industry and it gradually got the attention of the academia because it proved to be effective to solve industry problems. In the context of this article we will use the term ”DevOps” as a generic way to refer to all those concerns in the last mile of the delivery process which includes the ones related to the delivery itself and also to operations. All these DevOps topics are gradually being incorporated in the Software Engineering discipline. In fact, the version four of the Guide to the Software Engineering Body of Knowledge (which is still in beta) [IEEE Computer Society 2022], contains a new knowledge area called ”Software Engineering Operations”. At the same time, these topics are also getting interest from academic and scientific environments.

In our role of practitioners, we believe that these DevOps topics must be part of the Software Engineering body of knowledge. As a consequence, Software Engineering education needs to include these topics in their programs too. The challenge here, in our opinion, is that including those topics is not as simple as adding a new course. DevOps topics should be considered at the program level to be enable them to be to distributed in different courses.

In this article we aim to share some recommendations based on experiences in order to include DevOps concerns in Software Engineering education.

2. About DevOps

The term DevOps was coined back in 2009. There are several definitions of what DevOps is. Some of them are related to specific products or technologies. But one that is generally enough and is still compatible with most definitions is the one by Len Bass: *DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production while ensuring high quality.* [Bass et al. 2015]. Even when Bass' view of DevOps may sound exclusively focused on delivery, it also encompasses operational concerns because they are needed in order to get fast changes in production without compromising the stability of the system.

Approximately at the same time that DevOps started to get popular, there was another movement that got important visibility too: Site Reliability Engineering (SRE). This term was coined by Google long time ago but after the publication of the book "Site Reliability Engineering: How Google Runs Production Systems" [Beyer et al. 2016] it was widely spread in the industry. SRE incorporates Software Engineering practices to infrastructure and IT operation management. Some people consider SRE something different from DevOps or even an alternative, but Google proposes SRE as an specific implementation of DevOps [Fong-Jones and Vargo 2018].

3. Related work

Scientific publications about DevOps grow up every year. Some institutions have started to include DevOps in their programs. In some cases specific DevOps courses have been created [Demchenko et al. 2019, Hobeck et al. 2021, Jennings and Gannod 2019]. In some other cases DevOps topics have been added to existing Software Engineering courses [Hills 2020, Paez and de la Fuente 2022]. Hobeck et al. [Hobeck et al. 2021] delivered a DevOps course in two universities using a inverted classroom strategy. Alves and Rocha [Alves and Rocha 2021] designed a project-oriented DevOps course and conducted an empirical study to evaluate it. Their results evidence that first-hand experience paves the way for the comprehension of DevOps. Demchenko et al. [Demchenko et al. 2019] proposed a "DevOps Software Engineering Body of Knowledge" and described their implementation in a course at the University of Amsterdam. Chatley and Procaccini [Chatley and Procaccini 2020] describe how they introduced students to DevOps practices early in their degree program, and then wove their use through many different projects.

In the research field, Schilling [Schilling 2022] discusses the justification for including DevOps within a Software Engineering program and the associated challenges it places upon an instructor to teach. Fernandes et al. have studied DevOps challenges

and recommendations for teaching DevOps. They initially performed a literature review, and after that, they performed an interview study with 14 DevOps educators from different universities [Fernandes et al. 2022]. They identified 83 challenges and over 180 recommendations.

Our approach to teach DevOps topics, which is described in the rest of this article, is different from the mentioned works because: 1) we are doing a proposal at the program level and 2) at the same time we are giving concrete recommendations regarding the teaching strategy.

4. Our Current Program

What we describe in this section is related to the four-and-half year degree in System Analysis [1] offered by the Engineering School of the University of Buenos Aires. This degree had a major update in 2014 that, among other things, introduced several changes in the Software Engineering area. The body of knowledge that covers from business needs to the delivery, operation and support of the solution is organized in four core courses:

- Software Engineering Fundamentals 1 (95.20): this course is focused on the first part of the delivery process, which includes concerns like product discovery, requirements, domain modeling, but also covering some architecture, development and deployment concerns. In fact, as part of the course, the students have to develop a web application and deploy it to the cloud. More details about this course in Software Engineering and the program in general have been described by Villagra et al. [Villagra et al. 2020].
- Software Engineering Fundamentals 2 (95.21): this course is more focused on the second part of the delivery process which implies more technical concerns like design, configuration management, deployment strategies and several topics that are usually associated with DevOps. This course is described in detail in a recent publication by Paez and de la Fuente [Paez and de la Fuente 2022].
- Software Development Management (95.24): this course is guided by the project management knowledge areas as defined by the PMBOK (scope, time, costs, risks, quality, etc.), but applied to software development and with more than one approach. Therefore, students are confronted with management from both traditional and agile perspectives, following iterative-incremental and continuous flow life cycles, both in project and product mode. Special attention is given to team and people issues, including soft skills, such as communication, meetings, collaboration, collective learning, and working environment. More details about this course have been described by Fontela and Santos [Fontela and Santos 2022].
- Software Operations Management (95.59): this course is focused on design, operation and management of informatic services. This is far beyond the traditional Software Engineering scope, although is closely related to operational concerns which is under the scope of DevOps.

A matter worth mentioning is that the two courses on Software Engineering Fundamentals cover all the disciplines of the software delivery process but with different focus and level of detail. Figure 1 shows how these courses maps on to the software delivery stream.

[1] The name of the degree in Spanish is *Licenciatura en Análisis de Sistemas* and it could be considered a Master Degree in Software Engineering

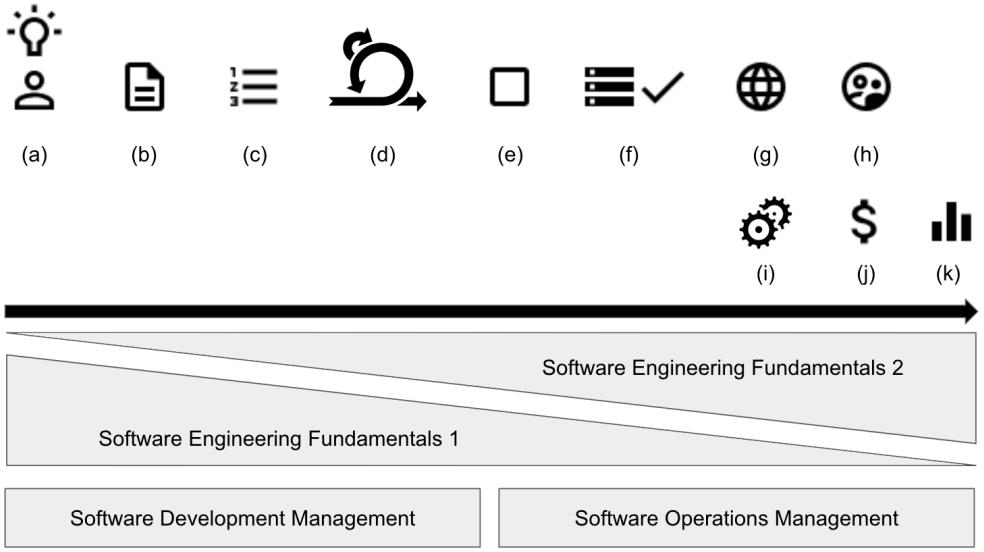

Figure 1. Software Delivery Stream and Software Engineering core Courses
It starts with a user need(a), that leads us to create a project vision (b), we create a product backlog (c) and start working in an iterative and incremental way (d). At the end of each iteration we have a product increment (e), we verify it (f) and pass it through different environments in its way to production (g). There, it finally meets the end user (h) and the software finally provides value (j). We operate it to ensure its availability (i) and get run-time metrics (k) to understand user behavior and it all starts over again

In addition to these four core courses, there are others that focus on specific aspects of Software Engineering, like Software Quality, Software Architecture and Object-oriented Programming and Design.

We know program updates tend to take too much time while knowledge, practices and tools in this area are in constant evolution. At the same time, professors in the software engineering area in our university, are usually practitioners. Therefore, professors try to keep their courses up-to-date moving faster than the program updates.

5. Recommendations

Recently, the School of Engineering decided to renew the programs of all degrees that it had been offering. In this context, we, as professors in the Software Engineering area, developed a set of recommendations to cover the Software Engineering body of knowledge in the new program of the System Analysis degree. Our recommendations are based on: a) the experiences we made in the Software Engineering courses during the last 10 semesters [Villagra et al. 2020, Paez and de la Fuente 2022, Fontela and Santos 2022] , 2) some recommendations made by other professors and researchers [Fernandes et al. 2022, Hobeck et al. 2021, Hills 2020] and 3) the guidance provided by the Guide to the SWEBoK version 4 [IEEE Computer Society 2022].

In the following subsections we describe each of our recommendations. They are listed in a sequential way but they are interrelated. Even when we are already applying some of these recommendations in our courses, we intend to make them an explicit part of the orientation of the upcoming program. It is worth mentioning that even when some of our recommendations are in some way generic and can be applied to many different topics, we consider them to be especially important when trying to teach DevOps.

5.1. DevOps Concerns

DevOps comprises a wide field of knowledge integrating organizational and technical aspects. It is too much content to be covered in one course. So our recommendation is to introduce DevOps topics in gradual way along several courses.

In our current program we are already covering some DevOps topics but not all what we would like to. Our idea for the new program is to add some topics to existing courses, and we plan to create new courses to cover some specific topics as well. In table 1 we present the list of DevOps topics we want to cover in our program along with our strategy to cover them. This list of topics was elaborated based mainly on the SWEBoK-V4[IEEE Computer Society 2022] and the DevOpsSEBoK [Demchenko et al. 2019].

Table 1. DevOps Topics Coverage

Topic	Course
DevOps mindset and Culture	C
Organisational impact of DevOps and Team structure	C
Continuous Integration and Delivery	C
Infrastructure as Code and Configuration Management	C, N
Cloud Computing Architectures, Service and Deployment models	C, N
Virtualization, Containerization and Container Orchestration	A, N
Networking	C, A
Monitoring, Telemetry and Observability	A, N
Resilience and Chaos Engineering	N
Operations, Service Levels, Outage and PostMortem Analysis	A
Security, Auditing and Compliance	A

C: already covered in existing courses; A: to be added to existing courses; N: new course

Some of the topics on the list are not new and have been in the Software Engineering body of knowledge for years. But for some reason, they might have been ignored and now, with the DevOps movement, they got more relevance.

5.2. Iterative and Incremental Delivery

Just as software is delivered in a iterative and incremental way, we believe that we should follow the same strategy to deliver knowledge. This implies that some topics should be studied in more than one course but, of course, with different level of detail, and possibly from different perspectives. This idea applies to traditional Software Engineering topics as well as DevOps topics.

At the same time, those practices learned in one course should be applied in all subsequent courses. This may sound obvious and it happens with some topics, but it does not always happen with Software Engineering ones. For example, a situation that we have detected in our institution is that students learn unit testing in a certain course, but they do not write unit tests in subsequent courses unless teachers explicitly ask them to do it.

We are applying this strategy in the Software Engineering core courses but now we are proposing this as an explicit guideline at the program level.

5.3. Transversal contents

There are some topics that are typically treated in Software Engineering courses but that we consider it could be much better to see them early in academic formation and along several courses. This idea applies to classical Software Engineering topics, like unit testing, but also to some DevOps related topics like Build Automation and Continuous Integration. This idea complements our iterative and incremental delivery approach. Let us consider the Continuous Integration (CI) practice. Early in the program the students could have a first contact with this practice by using a pre-configured CI pipeline. Later, in another course, the students could go deeper with the CI practice and design and configure themselves their CI pipeline.

We currently apply this strategy with several topics in our program. In particular, some Software Engineering topics like architecture and quality, are initially presented in the Software Engineering Fundamentals courses, and later the students take specific courses on those topics.

5.4. Hands-on Approach

Teaching effectively underlying lasting concepts, principles and practices of Software Engineering requires to put them into practice. At the same time, new generations of students want to understand what problems we need to solve before facing the theory that supports the solutions. Therefore, we think a hands-on approach is a good fit for this context. A hands-on approach allows the students to better understand the relation between tools, technologies and its underlying concepts, principles and practices.

When it comes to DevOps, tools and technologies may represent a major challenge for educators trying to deliver a hands-on approach because they are in constant evolution at a fast rate. This challenge compromises three dimensions: knowledge, guidance and support.

- **Knowledge**: in order to teach a topic educators must dedicate recurrent time and effort to stay up to date with tools and technologies.
- **Guidance**: educators must provide guidance and resources for students to learn and practice. This may require generating exercises, videos, guides, etc.
- **Support**: it is common, when dealing with tools and technologies for the first time, to face blocking issues, so educators must give support to the students to help them unblock.

This situation leads us, as educators, to reconsider how to plan our classes and here is where the flipped classroom may be an interesting option.

5.5. Flipped classroom

A flipped classroom approach proposes a reversal of the traditional class flow: the students "carry out" the lecture at home (for example, watching a video or reading an article) and then the "in-class" time is used for interactive activities. There is extensive bibliography regarding the usage of this technique to teach Software Engineering. Given the challenges brought by the hands-on approach, taking a flipped classroom approach sounds like a good match: deliver content in video to be watched at home and then use the class time to perform hands-on activities. These videos can be used to deliver different kinds of content like lectures, tools demonstrations and tools configuration/installation instructions.

Our Software Engineering Fundamentals courses have applied this strategy since their first edition in 2015 and several other courses have been adopting it recently, forced in part by COVID-19 pandemic.

5.6. Cloud Labs and virtual classes

In some institutions, the set of DevOps tools and technologies may represent a challenge for the laboratories. Frequent laboratory updates may be required. This may be partially mitigated by the use of cloud services, in fact some of the DevOps tools and technologies are cloud-based. Most of the cloud vendors have some kind of academic subscriptions that provide access to their services for free or at a very low cost. This may imply some initial bureaucracy for the institution to access to the cloud subscription while reducing the cost of laboratories.

At the same time, COVID-19 forced education to switch to virtual classes, at least for a while. In our own case, virtual classes result to be very convenient for some courses. In those cases where students have to work on development projects using a stack composed of several tools, the virtual classes allow the students to work on their own computers while the professors can give them remote-online support during the class time.

From a institution point of view, virtual classes require the institution to provide some service/subscription to allow professors to delivery online classes. At the same time, there may be some regulations limiting the virtual classes, but in our own experience, the combination of online and onsite classes allowed to have the best of both worlds.

6. Conclusions

DevOps came to stay and be part of the Software Engineering body of knowledge. It brings organizational and technical concerns, and teaching them face Software Engineering educators with major challenges. At the same time, DevOps is a wide topic and each institution may decide how deeply to cover it, depending on the program's profile.

In this article we shared how we started to include DevOps topics in some courses of our current program and some ideas we want to be implemented in the upcoming program. We also shared a set of recommendations to consider when including DevOps topics in Software Engineering education. While some recommendations may be applied to Software Engineering in general, we consider that using a flipped classroom strategy combined with a hands-on approach and virtual classes to be the best approach to teach the DevOps topics. A possible combination that proved to be effective in our case is: a) Deliver lectures in video, b) Use the class time to perform hands-on activities, c) Combine onsite and online classes, d) Use cloud-based services, e) Use online classes to give remote support to students working on their own computers and f) Provide tool guidance in video.

The surveys run at the end of each course have revealed that students are very pleased with the way we deliver the software engineering courses and the topics covered, in part because they see that are aligned with what they see in the daily work (most of our students are already working in the industry when they take the Software Engineering courses.)

References

Alves, I. and Rocha, C. (2021). Qualifying software engineers undergraduates in devops - challenges of introducing technical and non-technical concepts in a project-oriented course. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 144–153.

Bass, L., Weber, I., and Zhu, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 1st edition.

Beyer, B., Jones, C., Petoff, J., and Murphy, N. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Incorporated.

Bourque, P. and Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge, Version 3.0*. IEEE Computer Society.

Chatley, R. and Procaccini, I. (2020). Threading devops practices through a university software engineering programme. In *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*, pages 1–5.

Demchenko, Y., Zhao, Z., Surbiryala, J., Koulouzis, S., Shi, Z., Liao, X., and Gordiyenko, J. (2019). Teaching devops and cloud based software engineering in university curricula. In *2019 15th International Conference on eScience (eScience)*, pages 548–552.

Fernandes, M., Ferino, S., Fernandes, A., Kulesza, U., Aranha, E., and Treude, C. (2022). Devops education: An interview study of challenges and recommendations.

Fong-Jones, L. and Vargo, S. (2018). What's the difference between devops and sre? (class sre implements devops).

Fontela, C. and Santos, G. D. (2022). Learning communication skills in software development management courses: an active learning experience. In *2022 IEEE Biennial Congress of Argentina (ARGENCON)*, pages 1–8.

Hills, M. (2020). Introducing devops techniques in a software construction class. In *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*.

Hobeck, R., Weber, I., Bass, L., and Yasar, H. (2021). Teaching devops: A tale of two universities. In *Proceedings of the 2021 ACM SIGPLAN International Symposium on SPLASH-E*, SPLASH-E 2021, page 26–31, New York, NY, USA. ACM.

IEEE Computer Society (2022). Software engineering body of knowledge v4 public review.

Jennings, R. A. K. and Gannod, G. (2019). Devops - preparing students for professional practice. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–5.

Paez, N. and de la Fuente, H. (2022). Software engineering education meets devops: an experience report. In *2022 IEEE Biennial Congress of Argentina (ARGENCON)*.

Schilling, W. W. (2022). Wip: Integrating modern development practices into a software engineering curriculum.

Villagra, S., Benedetti, G. D., Bruno, T., Fernández, L., and Outeda, N. (2020). Teaching software engineering: an active learning experience. In *2020 IEEE Congreso Bienal de Argentina (ARGENCON)*, pages 1–6.