# Object-Oriented Programming with DevOps

Sam Chung
Southern Illinois University
1365 Douglas Dr. Mailcode 6614
Carbondale, IL 62901
1-618-453-7279
samchung@siu.edu

DevOps is an emerging culture that emphasizes continuous collaboration between software developers and IT operators through continuous standard process with automated tools for continuous delivery. DevOps participants take diverse roles to support its values - continuous collaboration, continuous process, and continuous delivery. A development team needs to be familiar with user cases, Object-Oriented Analysis (OOA), Object-Oriented Design (OOD), Object-Oriented Programming (OOP), and software testing. A quality assurance team must know use cases, abuse cases, software testing, and penetration testing. An operation team requires understanding deployment of Application Programming Interface (API) documents and executable components, and monitoring them and sharing their monitoring outcomes with both development and quality assurance teams.

In this paper, we challenge how we can infuse DevOps into a beginning-level programming course. Instead of teaching OOP, can we teach OOP with DevOps? If so, what concepts and practices need to be included in the programming course without losing what the course originally covers? What new subjects and practices do we need in order to design an introductory OOP with DevOps?

For this purpose, we first visit the concept of DevOps and their values in terms of 'Continuous *' in which * denotes Collaboration, Process, and Delivery. In addition, we discuss Evidence-Based Practice (EBP) to identify DevOps for OOP. EBP is an approach seeking best practices that have underpinning research evidences

Second, we explore the emergence of DevOps and their related research. However, we cannot find pilot or case studies with programming that demonstrate how computing major students studying beginning-level programming can experience DevOps through reengineering a non-DevOps-aware software application to a DevOps-aware application.

Third, we seek best practices in DevOps that can fulfill the values of DevOps – Continuous *. One of the major concerns of software industry is the lack of collaboration and communication ability for teamwork between existing and newly hired software developers immediately out of college. To enhance collaboration, DevOp participants need to understand their concerns and define the concerns through architecture. We also need a standard process to manage the application project during software development life cycle – software process. Then, for continuous deployment, we can see many research outcomes in software testing, software documentation, software security, and software deployment.

Fourth, we demonstrate how we can reengineer a given legacy application to a DevOps-aware target application. We choose a simple legacy application from an introductory programming course. We apply the identified evidence-based practices to the application, 'Payroll.java' to demonstrate the OOP with DevOps.

Fifth, we collect and analyze what concepts and best practices of DevOps that we employed during the reengineering process. Then, we argue that we need to infuse those identified concepts and practices into an introductory programming course towards programming with DevOps.

The results of this research clearly show that we can teach DevOps in the introductory programming course without losing what the course covers. The curriculum of the course covers important Java practices useful for continuous collaboration such as method definition and invocation, class definition and instantiation, control statement and continuous deployment for JavaDoc comment, exception, try statement, and catch statement. We also need to expand the current curriculum for continuous collaboration by including the IPO model and the MVC architecture. The students can learn separation of concerns and software architecture through functional and object-oriented decomposition. We need to emphasize the effective use of Java package in the programming practice.

We need to introduce an agile software process for the students to learn iterative and incremental approach. If the students will use this process continuously for their programming assignments, they will learn one of the values of DevOPs – continuous process.

We need to cover more for continuous deployment by practicing software documentation, software security, and software testing. Although the introductory course covers JavaDoc comment, we need to show how we can use the JavaDoc to generate API documentation for public classes and methods. To learn how we can make an application secure, we need to show how we can define a user-defined Exception class for a Model class and can call it in a pair of Try and Catch statements within a View class. We also include new subject areas for software testing with JUnit, input data validation with pattern matching using regular expression, and software deployment with a JAR file, not class files.


*SIGITE'17, October 4–7, 2017, Rochester, NY, USA*

ACM ISBN 978-1-4503-5100-3/17/10.
https://doi.org/10.1145/3125659.3125670