**Agile vs. DevOps: A Comparison of Two Software Development Methodologies**

A Report by

Mark Kardash

SDEV360: Secure Software Engineering

Professor David Gabello

25 October 2023

University of Maryland Global Campus

Software Development is a field encompassing many different tasks, which require many different approaches to be completed. These approaches are known as methodologies, and are divided into four categories: Traditional, Agile, DevOps, and Secure Development Lifecycle (SDL). While all of these have their advantages and disadvantages, this report will compare the Agile and DevOps methodologies, using reliable sources to highlight their similarities, differences, and main components.

To avoid confusion, the analysis will begin with the Agile software development methodology, later comparing it to DevOps. Laoyan (2022) defines Agile as “a project management framework that breaks projects down into several dynamic phases, commonly known as sprints.” (Summary). This methodology makes developers’ work easier, by allowing them to complete projects in parts, rather than work on everything at once. Organizing meetings after each sprint also provides opportunities to address any difficulties, mistakes, and need for improvement (Laoyan, 2022). The basis of Agile methodology is the idea that developing quality, functional software matters more than extensive documentation on it (Tripathi & Mishra, 2017). To uphold this standard, Agile users follow four core principles, known as the four pillars of Agile:

1. Individuals over processes and tools: Human collaboration has more value than any machine.
2. Working software over comprehensive documentation: It is more important to create a system that functions properly than to document every little detail.
3. Customer collaboration over contract negotiation: Once again, because of the great value of human collaboration in Agile, the customer always plays an active role in software development, advising the team on their next steps. The formality of contracts is not as important.
4. Responding to change over following a plan: Since each Agile project is broken into phases, developers using it are not as strict about following a solid plan, having the ability to switch between approaches and strategies.

(Laoyan, 2022).

Because the Agile methodology focuses greatly on customer satisfaction, companies using it ensure that both customers and developers communicate with each other effectively, and that all efforts are directed towards providing a quality product. To achieve this, Agile provides several models to follow when completing a project. These include Extreme Programming (XP), Agile Modeling (AM), SCRUM, Crystal methodologies, Feature-Driven Development, and Adaptive Software Development (Tripathi & Mishra, 2017). For the comparison with DevOps, the first two will be analyzed in detail below.

The Extreme Programming (XP) model consists of several basic principles for effective software development (Tripathi & Mishra, 2017). The first is to write simple, but highly functional code, to reduce cost of any future maintenance. The second is to conduct rigorous testing at various points of development, to eliminate as many issues as possible before the final release. Yet another recommendation is to welcome sudden changes to the project, rather than be intimidated by them. In this way, the project can be kept “alive”, allowing customers to provide new ideas for the still developing product. Finally, the overall guiding principle of the model is to develop a product of high quality (Tripathi & Mishra, 2017).

The above principles can be applied to a project across all its stages, including initial planning, design, coding, and testing of an app. Although the principles themselves have existed for a long time, they were only recently combined in the XP model (Tripathi & Mishra, 2017). To accommodate them, developers have integrated several common practices when using XP. A few of these practices are:

* Sitting close to each other: Because of the importance of face-to-face communication in Agile, team members are often put close together, without physical barriers, such as a wall.
* Pair Programming: Having two programmers behind one machine. Increasing the workforce will increase both speed and efficiency.
* Cycles: Cycles are meetings taking place after critical stages of a project. There are two types of cycles: The Weekly Cycle occurs after each iteration, while the Quarterly Cycle occurs after release. The goal of these meetings is for developers to share progress, ideas, and any concerns about the project.
* Continuous Integration: This involves testing the code each time that new changes are added to it. In this way, any existing errors will be found and corrected early in the design process, ensuring a smooth release.
* Incremental Design: This essentially means preparing beforehand. If a developer is at least somewhat familiar with the system before beginning, it will be easier for them to develop and test specific components.

(Agile Alliance, 2023).

The Extreme Programming model provides several major benefits. The first of them is that developers are always made aware of what the customer wants, constantly communicating with each other. This allows customers to provide feedback to the developers based on the working models they are presented with, enhancing the project while still in the works (Tripathi & Mishra, 2017). Another benefit is the reduction of both time and costs involved in a project due to design simplicity. Finally, the XP model benefits development with its great adaptability to changes, and reception of requirements through the feedback cycle (Tripathi & Mishra, 2017).

Despite the above advantages, there are several shortcomings to Extreme Programming. They are mostly related to the model’s emphasis on collaborative work. In companies with too many employees to cooperate (over 12 people), or where cooperation is not possible due to physical distance, the XP model can do more harm than good, and a different approach is recommended. Finally, it is best not to use Extreme Programming when the product being developed does not require extensive testing (Tripathi & Mishra, 2017). Therefore, for the XP model to serve its full potential, certain preconditions must be met.

The second model to be discussed here, appropriately called Agile Modeling, has both similarities and differences to XP. Agile Modeling also places emphasis on communication and collaboration between developers, through which product feedback is provided, as well as on the development of simple but functional code. Users of Agile Modeling are expected to welcome changes on the same level as XP users. However, two additional things valued in Agile Modeling are courage and humility. This essentially means being prepared to make difficult decisions regarding the product being designed, while respecting the input and ideas of others, treating everyone as equals, and admitting personal mistakes (Tripathi & Mishra, 2017).

Thanks to its many commonalities with Extreme Programming, Agile Modeling can be used in combination with both that and other models. However, it also has its own practices and principles. To use it, developers are required to fully understand the purpose of modeling, focus on the content, rather than representation, of their work, and working with a multitude of models. Agile Modeling is often used in Agile Model Driven Development (AMDD), where it is combined with other approaches to model the entire system in early development stages (Tripathi & Mishra, 2017).

The advantages of Agile Modeling are also very similar to XP, such as team-customer communication, reduced project costs and time, adaptability to changes, timely feedback, and fast software production (Simplilearn.com, 2021). However, the model carries with it some disadvantages as well, such as the need for experienced and knowledgeable developers to handle the process, the potential confusion of teams due to lack of extensive documentation, the uncertainty about the amount of work to be done, and the excessive dependence of the model on stakeholder input (Simplilearn.com, 2021). Therefore, before using Agile Modeling, both developers and customers must make sure they have a clear plan for creating the future product.

Overall, Agile methodology is a useful approach to software development, as long as it is used on the right projects. The advantages of and disadvantages of Agile are mostly shown through those of its models. Because of the principles it follows, this methodology is best suited for companies where employee collaboration and communication are highest, while the number of employees themselves is small. In addition to this, since Agile focuses on quality software, the success of the results mostly depends on the skills of the team members. Because some software components are created by a single developer, such practice is a risky bet for companies. Lastly, Agile is unsuitable for things like software maintenance, due to its extensive focus on specific software components (Tripathi & Mishra, 2017). Based on all these facts, to use the Agile methodology to its full potential, one must know not only how to apply it, but in what circumstances to apply it as well.

Now that the Agile software development methodology has been sufficiently explained, it can be successfully compared to the second methodology explored here, DevOps. DevOps (from “Development” and “Operations”) is defined by Amazon Web Services as “the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity” (para.1). Although software quality still matters in DevOps, this methodology places more emphasis on the speed of delivery for final products. While Agile often siloes its teams, having individual developers work on specific components of a project, DevOps occasionally merges the Development and Operations teams into one, thus getting its name. This practice allows all developers to work together through every stage of the project, learning diverse skills that span across many of its aspects (Amazon Web Services, n.d.). Quality assurance and security are important among the values of this methodology and are thus often integrated into DevOps. DevSecOps is the practice of developing software with a focus on security (Amazon Web Services, n.d.). The emphasis on quality assurance in DevOps echoes, in some ways, the Extreme Programming model in Agile, where extensive testing is required before a product can be released.

Just like Agile, DevOps follows its own set of principles, some of which intersect with those of the former. According to Corrales (2023), the 7 principles of DevOps are:

1. Collaboration: As in Agile, collaboration and cooperation between developers is highly valued in DevOps, as it can bring the best results for the final product.
2. Data-Based Decisions: DevOps users follow the principle that to make an effective project-related decision, data about each option must be collected and compared.
3. Customer-oriented development: DevOps users believe it is important to focus their work on the customer’s needs, and develop products based on feedback from them, once again echoing Agile.
4. Automating: While automation in Agile is given attention only through Agile Modeling, DevOps places much more value on it. This is because DevOps focuses a lot on speed. By using automation whenever possible, teams can expedite tasks that would take too long to complete manually.
5. Continuous improvement: Constantly improving software allows developers to learn from mistakes, and meet customer needs more adequately. It is a similar practice to the incremental releases used by Agile.
6. End-to-End Responsibility: As the software passes between phases of a project, the developers handling each phase is responsible for fixing the issues left after the previous phase.
7. Embracing Failure: This is similar to the principle of Courage in Agile Modeling, as it encourages teams to take risks, and not be afraid to make mistakes, using them as a learning opportunity.

Rather than being split into several different models, the DevOps methodology is a model in itself. To be an effective tool for software development projects, it follows a list of practices that can be adapted according to a particular situation. Amazon Web Services lists these practices as follows:

* Continuous Integration: In this process, tests are performed after all project code has been merged into a single central repository. This allows for fast, more efficient detection of bugs, and improvement of overall software quality.
* Continuous Delivery: This practice uses automation for the entire software development process, often deploying any changes in the code to the testing environment, to integrate them into the final product. This reduces the amount of developers’ work, giving them an almost finished product.
* Microservices: In this practice, the software is not developed as a single batch of code, but rather as a series of smaller services, each capable of running separately from the others. Despite this autonomy, they are connected to each other through pathways like an application programming interface (API). Microservices make the DevOps model ideal for business, as each has a specific task it can complete.
* Infrastructure as Code: DevOps engineers may use code to manage infrastructure, using API to handle infrastructure at the level of programming, reducing the amount of manual tasks to be performed.
* Monitoring and Logging: Unlike Agile, which prioritizes quality code over documentation, DevOps encourages teams to document and observe as many details as possible about the software. The collected log data is then sorted and analyzed, providing developers with an understanding of the impact that their product updates have on the users, while also informing them of any issues that occur after release.
* Communication and collaboration: Constant cooperation between team members is just as important in DevOps as it is in Agile, with the only difference being that in DevOps, the customer is not as directly involved in the process. These practices are upheld by tools like chats, as well as by the proximity of all development processes.

Using the DevOps software development methodology can speed up the completion of tasks, therefore increasing customer satisfaction, keeping up with the market, and increasing positive results (Amazon Web Services, n.d.). It can also give a company an advantage over its competitors by allowing a faster resolution of issues. Techniques employed by DevOps, such as automation, can improve cooperation and communication between team members, while ensuring that the final product is secure against potential threats (Amazon Web Services, n.d.). On the other hand, the methodology has some significant disadvantages, such as the small number of available DevOps specialists, high costs of infrastructure, and unmet need for professional knowledge to perform processes like continuous integration or automation (Great Learning, n.d.).

The Agile and DevOps software development methodologies are both similar and different, in many ways. Some aspects, such as automation and focus on customers, are present in both approaches, but given higher priority in one than the other. Even some advantages, such as speed, are common between them. Developers must look at the intricacies of each individual project, such as its nature, costs, duration, and customer feedback, to decide which methodology should be followed in completing it.

|  |  |
| --- | --- |
| Agile: | DevOps: |
| Focus on functional product over excessive documentation. | Focus on speed of delivery. Values logging and monitoring data for better customer service. |
| Teams are siloed so that individual developers or groups work on specific project components. | Development and Operations teams merged, enabling them to work through the entire project, and learn together. |
| Requires rigorous testing of software before release. | Values attentive quality assurance (testing) and security checks of software before release. |
| Prioritizes collaboration and communication. | Prioritizes collaboration and communication. |
| Makes decisions based on customer needs and feedback. | Makes decisions based on data, customer needs, and feedback. |
| Uses automation as an option in Agile Modeling. | Uses automation across multiple project phases. |
| Encourages developers to take risks and learn from mistakes. | Encourages developers to take risks and learn from mistakes. |

**Table 1:** Comparing the Agile and DevOps Software Development Methodologies.

References:

Corrales, E. (2023, August 23). *What are DevOps principles?* TechRepublic. <https://www.techrepublic.com/article/devops-principles/>

*DevOps*. (n.d.). Great Learning. <https://www.mygreatlearning.com/devops/tutorials/devops-advantages-and-disadvantages>

Laoyan, S. (2022, October 15). *What is Agile methodology? (A beginner’s guide)*. Asana. <https://asana.com/resources/agile-methodology>

Simplilearn. (2021, May 21). *Agile modeling: Definition, core principles and advantages*. Simplilearn.com. <https://www.simplilearn.com/agile-modelling-article#:~:text=Agile%20Modeling%20Best%20Practices%201%20Active%20Stakeholder%20Participation.,Models.%20...%208%20Prioritized%20Requirements.%20...%20More%20items>

Tripathi, V. Y., & Mishra, A. (2017). Case Study of Agile Methodologies in Field of Software Development. *International Journal of Engineering Research & Technology (IJERT)*, *6*(11), 175-180.

*What is DevOps? - DevOps models explained - Amazon web services (AWS)*. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/devops/what-is-devops/>

*What is extreme programming (XP)?* (2023, October 18). Agile Alliance |. <https://www.agilealliance.org/glossary/xp/>