Contested Development: Rediscovering Pragmatism in Agile and DevOps

Agile and DevOps have emerged as dominant forces in software development, promising to deliver speed, flexibility, and quality. However, despite their widespread adoption, there is a growing sense of unease within the industry. Many practitioners are questioning whether Agile and DevOps have become overly dogmatic and prescriptive, leading to a loss of pragmatism in software development.

This article argues that Agile and DevOps are suffering from a crisis of pragmatism. We have become so focused on following the rules and rituals of these methodologies that we have forgotten the fundamental principles that underlie them. As a result, we are losing sight of the most important goal of software development: to deliver value to our customers.

In this article, we will explore the concept of 'Contested Development', which argues that Agile and DevOps have become too narrowly focused on a single way of working. We will examine the historical context of Agile and DevOps, critique their current state, and propose a more pragmatic approach that combines the best elements of both methodologies.

Contested Development: Finding Pragmatism in Agile & DevOps by Roy Furr

★★★★☆ 4.2 out of 5

Language	: English
File size	: 1716 KB
Text-to-Speech	: Enabled
Screen Reader	: Supported
Enhanced typesetting	: Enabled
Print length	: 79 pages

Agile and DevOps emerged in response to the challenges of developing software in the early 21st century. The waterfall model, which had been the dominant paradigm for software development for decades, was no longer able to keep pace with the rapidly changing needs of businesses.

Agile was developed as a way to address the shortcomings of the waterfall model. Agile emphasizes collaboration, flexibility, and iterative development. DevOps was developed as a way to bridge the gap between development and operations teams. DevOps emphasizes automation, continuous integration, and continuous delivery.

Both Agile and DevOps have been highly successful in their own right. Agile has helped to improve software quality and reduce development time. DevOps has helped to improve the reliability and security of software systems.

However, as Agile and DevOps have become more popular, they have also become more prescriptive. The Agile Manifesto, for example, outlines a set of 12 principles that all Agile teams must follow. The DevOps Handbook outlines a set of practices that all DevOps teams must follow.

This level of prescription has led to a loss of pragmatism in software development. Teams are so focused on following the rules that they are forgetting to think for themselves. They are blindly following the latest trends, even when those trends do not make sense for their particular context.

The current state of Agile and DevOps is characterized by a number of challenges:

- **Dogmatism:** Agile and DevOps have become too dogmatic. Teams are so focused on following the rules that they are forgetting to think for themselves. They are blindly following the latest trends, even when those trends do not make sense for their particular context.
- **Prescriptiveness:** Agile and DevOps have become too prescriptive. The Agile Manifesto and the DevOps Handbook outline a set of rules that all teams must follow. This level of prescription has stifled innovation and led to a loss of pragmatism.
- **Fragmentation:** Agile and DevOps have become fragmented. There are now dozens of different Agile and DevOps frameworks and tools available. This fragmentation has made it difficult for teams to find the right approach for their particular needs.

These challenges are leading to a crisis of pragmatism in software development. Teams are losing sight of the most important goal of software development: to deliver value to our customers.

We need a more pragmatic approach to software development. An approach that is based on the following principles:

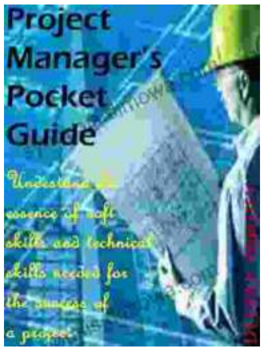
- **Contextualism:** There is no one-size-fits-all approach to software development. The best approach for a particular team will depend on their context.

- **Flexibility:** Software development is a complex and unpredictable process. We need to be flexible and adapt our approach as needed.
- **Collaboration:** Software development is a team sport. We need to work together to deliver the best possible results.

This pragmatic approach will allow us to rediscover the true spirit of Agile and DevOps. We will be able to tailor our approach to our own unique needs, and we will be able to respond to change as needed.

Agile and DevOps have been incredibly successful methodologies. However, they have also become too dogmatic, prescriptive, and fragmented. This has led to a crisis of pragmatism in software development.

We need a more pragmatic approach to software development. An approach that is based on the principles of contextualism, flexibility, and collaboration. This approach will allow us to rediscover the true spirit of Agile and DevOps, and it will help us to deliver better software, faster.


Project Manager's Pocket Guide
Understand the essence of soft skills and technical skills needed for the success of a project

Mastering Project Management: The Ultimate Guide to Success with Deepak Pandey's Project Manager Pocket Guide

In today's competitive business landscape, effective project management has become an indispensable skill for organizations striving for success. With the...

Let's Build Sue Fliess: Unleash the Polychrome Master Within

Chapter 1: The Art of Polychrome Sculpting In this introductory chapter, we delve into the captivating history of polychrome sculpture,...