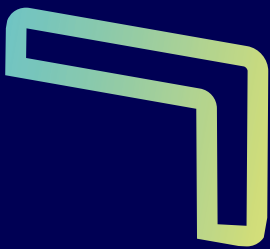




DOCUMENTATION LEVEL:
1 | PURPOSE
Why we do the things we do

Delivery and Operations Process

Book of Delivery



INDEX

THE DELIVERY AND OPERATIONS PROCESS	3
DELIVERY AND OPERATIONS	4
OVERVIEW	4
WHY AGILE DEVELOPMENT?	5
PRINCIPLES	7
SUBPROCESSES	14
SCALE DOWN	14

The Delivery and Operations Process

The Delivery and Operations Process defines the principles that should be followed by all software product teams covering the design, development and operations activities.

Our approach to software development encourages collaboration between all stakeholders. We believe that the best software products are those resulting from the combined and harmonic creativity, intelligence, and work of users, businesses, managers, designers, and engineers. This BizDevOps (also referred to only as DevOps or BizSecDevOps) approach, combined with agile software development practices, allows the organisation to develop software faster, be more responsive to user needs and maximise value.

Delivery and Operations

Overview

The concept of BizDevOps is founded on building a culture of collaboration between teams, removing the silo walls between the business, development and operations. The benefits include increased collaboration, trust, transparency and faster feedback, faster software releases, ability to solve critical issues quickly, and to manage unplanned work better. At Critical TechWorks, we use this collaboration mindset as the basis of the Agile Manifesto values.

AGILE MANIFESTO VALUES

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

We believe in these values. But more important than believing is living them and relying on them to drive our actions daily.

When we are at work, if we have doubts about the course of action, we should apply the values, which should be enough and ok.

A DevOps culture implements a set of practices that provides ongoing flow from business, development and operations with a feedback loop to continuously inspect and adapt what is being delivered to the users (Figure 1).

The process flow automation streamlines the movement of software change through the build, validate, deploy and delivery stages, while cross-functional teams have full ownership of software applications – from design to operations (production support). The transparency achieved in the entire process results in faster and more reliable increments in software design, build and release, thus maximising the value delivered to the users.

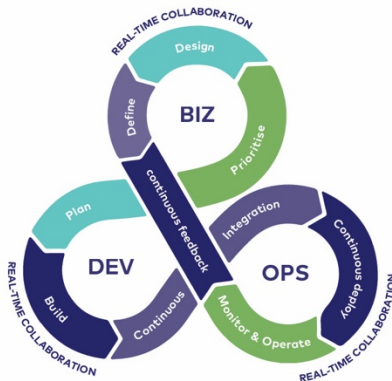


Figure 1 BizDevOps Approach.

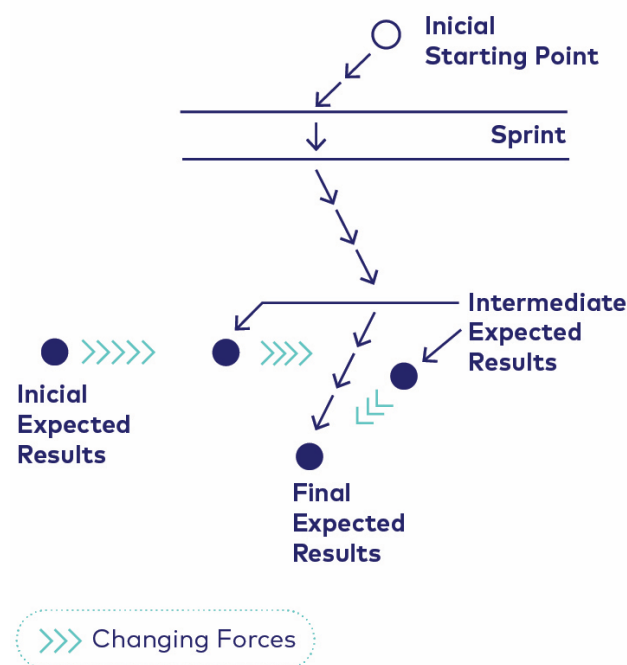
Why Agile Development?

The world today is increasingly marked by Volatility, Uncertainty, Complexity, and Ambiguity, or VUCA. We believe that by following the agile software development values and principles, we will be more prepared to deal with the VUCA context. We expect to produce the following main benefits, not limited to:

Responding to change

Change is inevitable. As product development time passes, the knowledge about the product increases and the need to change initial thoughts also increases.

Being able to accommodate change systematically is essential to build great products, and agile development allows teams to respond quickly to change. The flexibility of agile methods contributes to product quality and stability because changes in agile projects are predictable and manageable.



System and business thinking

Having system and business thinking is critical to be aware of how individual actions affect not only the team but also all the other groups involved in the product release process, including the business and users. Lack of systems thinking is often present in teams that work in silos.

A change in mindset to look at the development process holistically and break down the barrier between Biz, Dev and Ops is vital to guarantee the autonomy and responsibility of everyone involved.

Better manage unplanned work

Whether you realise it or not, unexpected work is a reality that every team faces because it is not possible to anticipate everything. This reality often impacts team productivity. With visibility, clear prioritisation and proactive retrospection, the BizDevOps teams can better manage and anticipate unplanned work while continuing to focus on prioritised work.

Increasing quality

A faster feedback loop allows the team to thrive with higher quality and stability. Short development cycles (sprints), automated release pipelines, full transparency and seamless communication enable teams to release faster and more frequently, minimise downtime and accelerate time to resolution.

Work smarter

Manual tasks, development and testing in sequence, development and operations performed by different teams, and poor incident response time kill the pace and team confidence.

Based on methods and processes that promote collaboration and flow, teams can increase productivity and release more frequently with fewer problems through automation and standardised tools.

Principles

“A principle is a kind of rule, belief, or idea that guides you.”

These guiding principles apply to all Critical TechWorks BizDevOps teams. We believe they contribute to better products, satisfied users, and happy and motivated teams. The principles are guidelines. They focus on capabilities and behaviours that BizDevOps teams are expected to follow but give adaptability not just at the instantiation moment of the project but continuously along with the product cycle.

The Chief Technical Titans, accountable for the teams' delivery and operations, are responsible for guaranteeing that these principles are applied by all teams and helping them by doing so whenever needed. They will also inspect, adapt and evolve the principles as needed in alignment with the organisation goals and the target of continuously improving them.

Always remember that the Delivery and Operations principles can be instantiated differently from product to product. Still, we should always look to implement them to the maximum extent to increase engineering excellence and achieve Product Excellence. This is the responsibility of all the roles involved in the product team, and each of them needs to understand how their own specialism contributes to each principle. This is even more important because these principles work together with the people behaviours stated in the People and Culture principles.

We had our 12 delivery and operations principles as a flat structure. We will keep these 12, while also believing it is better to group them into the areas of excellence. These areas help us understand our target focus with each principle. They are interconnected and are essential for creating an effective software engineering organisation, helping ensure balanced development practices that drive innovation, efficiency, and customer satisfaction. We achieve Principles through behaviours that we should demonstrate as individuals. That's why Principles are written in the first-person singular “I”. Below are the three areas of excellence used to organise the principles:

Collaboration, Learning, and Agility. We focus on fostering an environment where we can work together effectively, learn from each other, and quickly adapt to changing circumstances. We emphasise open communication, cross-functional collaboration, active knowledge sharing, and embracing a growth mindset. By prioritising these aspects, we can enhance our agility and ability to respond proactively to evolving requirements and challenges.

Quality, Reliability, and Resilience. Centring around ensuring the delivery of high-quality software that is reliable and resilient. We concentrate on practices such as continuous integration and deployment, automated testing, infrastructure as code, and designing for failure. By adhering to these principles, we create more robust systems that can withstand unexpected events and maintain consistent performance over time.

Visibility, Monitoring, and Continuous Improvement. Aiming to promote transparency and data-driven decision-making by emphasising the importance of visibility into system performance, user feedback, and process efficiency. We encourage active monitoring, regular reviews of system health and user experiences, and a culture of continuous improvement. By focusing on these aspects, we can proactively identify issues, optimise processes, and enhance overall product quality over time.

Collaboration, Learning, and Agility

Collaborative and Experimental Environment

I foster a collaborative and experimental environment where knowledge is shared across teams and domains, making decisions easier. This culture of continuous learning encourages experimentation, turning failures into opportunities for growth and improvement. By embracing risk-taking and high-trust, I develop more secure, reliable, and innovative systems that benefit everyone involved. I don't settle for conformity.

Organise regular cross-functional workshops, hackathons, or brown bag sessions to foster experimentation and knowledge exchange.

Implement Pair Programming and Mob Programming practises to promote collective code ownership and learning.

Analyse the result of work increments and derive actions to address problems or improve ways of working to achieve better delivery and operations

Establish a culture of blameless post-mortems after incidents or failures to extract learning and take actions to improve processes. Put the interests of the Product and team before individual interests.

Cross-Functional Teams

I advocate for balanced teams with diverse skill sets when influencing team composition. I actively contribute to creating fully responsible teams capable of handling the entire product lifecycle, fostering autonomous and self-sufficient collaboration among their members.

Create teams consisting of developers across the whole stack, designers and product visionaries.

Encourage team members to learn new skills and create a plan to become T-Shaped within your role and product to promote adaptability and versatility.

Establish a culture of collective ownership and responsibility for delivering software from inception to production.

User-Centric Approach

I prioritise a user-first approach, involving users actively in the development process through feedback, metric tracking, and collaborative decision-making.

Incorporate user feedback into backlog creation, prioritisation, reviews, and business goals achievement.

Give users a seat at the table for feature prioritisation and decision-making.

Develop strategies to compensate for limited or no user availability.

Self-Organised System

I define my own workflow and implementation approach based on product needs and the overall work that teams need to do. I use our company values of autonomy, self-organisation, and maximising value, and industry best practices to deliver high-quality solutions with minimal overhead. I don't ask for more autonomy than the accountability I am able to provide.

Proactively use your role's accountabilities to action the delivery and operations principles.

Define your team's workflow and implementation approach given our internal processes and BMW product context.

Consider every team member's opinion when making decisions. Sometimes, any decision is better than no decision at all.

Deliver high impact with minimal overhead, aligning with company values and industry best practices.

Quality, Reliability, and Resilience

Safety and Security First

I prioritise safety and security above all else. To achieve this, I deeply understand our users' needs and business goals through thorough knowledge acquisition. Every stage of software development, from design to deployment, is adequately documented and traceable for accountability and transparency. Regular audits, reviews, and clearly specified SLAs ensure ongoing protection and reliability throughout the system's lifecycle.

Understand and develop your security and safety product context.

Implement secure coding practices and enforce them through code reviews and automated tools.

Conduct regular security audits, penetration testing, and risk assessments.

Know and follow the defined incident response processes to minimise impact and recovery time.

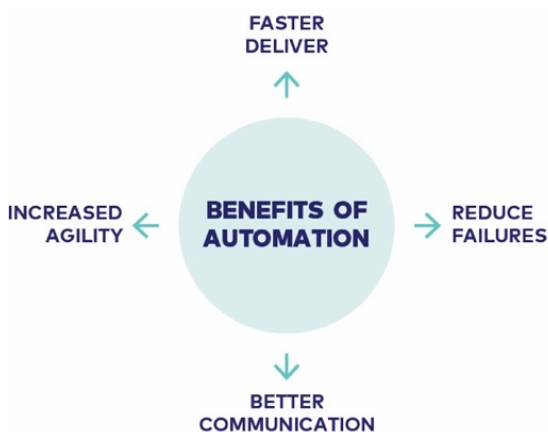
Automate Everything Possible

I automate everything possible through Continuous Integration and Deployment (CI/CD) practices, minimising manual effort, reducing errors, and accelerating innovation. This enables us to focus on creative problem-solving and innovation while streamlining repetitive tasks, fostering a culture of continuous improvement and faster delivery. I take decisive measures and inform outside the team if validation is being neglected.

Implement Infrastructure as Code (IaC) for managing environments consistently and efficiently.

Automate tests— unit, integration, end-to-end, and acceptance tests.

Use CI/CD pipelines to automate builds, tests, deployments, and rollbacks.



End-to-End Responsibility

I'm fully accountable for the entire product lifecycle, from development to end-of-life, and must focus on delivering high-quality results while understanding their role in the bigger picture. I ask for support when needed and take responsibility when things run well and when they don't. I am present when needed.

Take full responsibility for the quality and operations of products engineered while meeting business and customer expectations.

Prioritise delivering high-quality results while keeping track of progress and continuously improving processes to ensure timely completion.

Understand how individual roles contribute to the bigger picture, fostering collaboration and alignment within the organisation to achieve shared goals.

Sustainable Engineering

I must balance technical excellence with financial responsibility to deliver solutions that meet business requirements, are efficient, and sustainable.

Manage solution costs proactively, finding a balance between the best product and technical solutions and cost-effectiveness.

Deliver solutions on time and in line with business goals, identifying waste and streamlining operations to increase efficiency and sustainability.

Ensure that software solutions address business and all known non-functional requirements.

Take proactive steps to create cost-effective solutions and ensure the long-term sustainability of software solutions.

Visibility, Monitoring, and Continuous Improvement

Transparency Throughout the Entire Process

I need to provide insights of what I'm doing, to others, as to foster autonomy, trust, and responsibility through transparency and open communication. I don't ask for more transparency than I am able to provide.

Have complete knowledge of product team members' tasks and responsibilities.

Build information from all the available data to create transparency and support decision-making.

Information and decisions are shared transparently and as early as possible to build trust and understanding.

Take responsibility for your work, leverage available tools, and strive to maximise the product's potential.

Continuous Feedback and Improvement

I recognise the importance of personal growth through giving and receiving constructive feedback. As such, I strive to implement tailored feedback frameworks and strategies that cater to our product needs, using continuous improvement and lessons learned from failures as means to stimulate growth and minimise waste.

Implement a suitable feedback framework or strategy to create improvement actions (both team and product) that you can track and inspect their progress.

Use data as a feedback loop to continuously improve and learn from mistakes, minimising waste and optimising speed.

Combine agile methods and practices with direct feedback to support better product development and drive growth.

Monitor Everything Frequently

I prioritise implementing key performance indicators (KPIs) and other metrics that provide insights into efficiency, quality, and business outcomes. These indicators are essential for identifying areas requiring improvement, optimising processes, and maximising productivity early in the product life cycle. By regularly monitoring and validating these KPIs, I can ensure continuous improvement, prevent costly retrofits, and support future business decisions with data-driven evidence. Furthermore, I am accountable for monitoring all systems supporting operational activities to identify areas for improvement and optimise processes, ensuring a smoother operation and more informed decision-making about future improvements and expansions.

Implement key performance indicators (KPIs) and other metrics to monitor performance, efficiency, quality, and business outcomes.

Regularly monitor and validate these KPIs during the product life cycle to support continuous improvement and informed decision-making.

Be accountable for monitoring all systems supporting operational activities, identifying areas for improvement, and optimising processes for smoother operations and data-driven decision-making.



Respect Standards

I am mindful of the necessary regulatory standards that guide our development efforts, integrating these into our DevOps workflow seamlessly. By validating code against these at every stage, we maintain transparency and accountability without compromising agility or confidence in delivery.

Recognise that guidelines for adhering to relevant rules, frameworks, and best practices exist and integrate them into your daily work.

Implement automated tools and checks to perform a thorough validation in the software delivery lifecycle.

Conduct regular audits and assessments to identify and address any gaps or issues, reinforcing our commitment to be aligned to the defined guidelines.

Subprocesses

Scale down

This main process describes the principles, expected capabilities, and behaviours related to delivery and operations. The next detailed level, the Capabilities circle, details our core practices and sets in motion our way of collaborating.

The subprocesses indicate how we deliver value to the users by providing information about the delivery and operations working models, the flows we have in place, how we work together within the teams and how they approach their daily work.

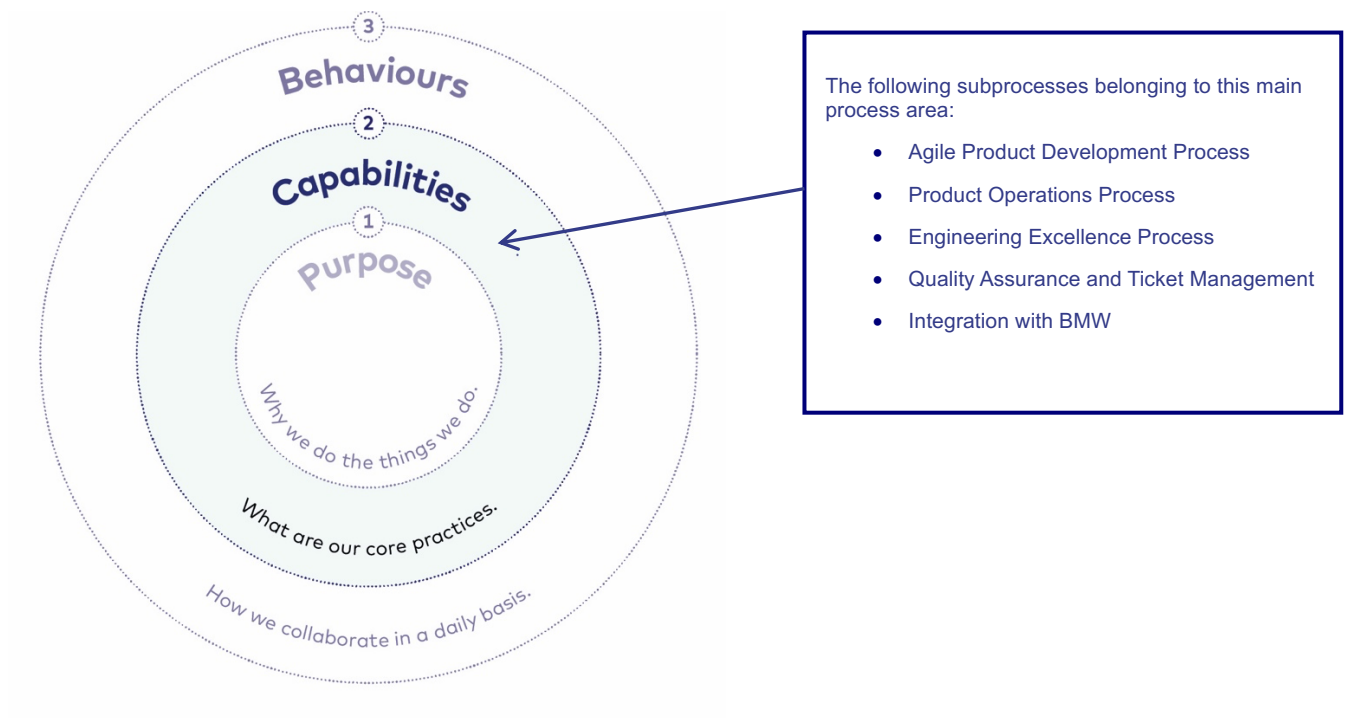


Figure 2 Golden Circle of Documentation.

Revision History			
Version	Date	Description	Author/Reviewer
1.0	2020 February	First approved version	Several contributors
1.1	2024 March	Template update. Process revision – no necessary changes.	Trends and Futuring Quality Guardian
2.0	2025 April	Delivery Principles re-thinked	João Pedro Pinto João Gonçalves Pedro Vieira da Silva

Information Classification Level and Access List	
Classification level	Internal
Disclaimer: The content of this document, regardless of its classification, are under copyright of Critical TechWorks, released on condition that it shall not be copied in whole, in part or otherwise reproduced (whether by photographic or any other method) and therefore shall not be divulged to any person other than the addressee (save to other authorised offices of his/her organisation in need of knowing such contents, for the purposes for which disclosure is made) without prior written consent of Critical TechWorks.	
Access list	
The access list attribute is considered complementary information and should be used to clearly indicate which recipients shall have access to the content of this document, following the principles of least privilege and need to know. If the access list is left blank, access is applicable to the entire scope covered by the classification level.	

**Get ready
for a journey
like no other!**



**Critical
Techworks**

CTW-2019-PCS-00043
