**Organization of cross-functional (DevOps·PM) teams in e-commerce: processes for teamwork**

**Abstract**. The article investigates how to organize cross-functional e-commerce teams by integrating DevOps and traditional project management (PM) processes with systems engineering and quality assurance practices. Drawing on empirical data, classic Waterfall, Agile, and DevOps models, and a review of advanced techniques, it proposes a unified team structure comprising a Product Owner, Business Analyst, UX/UI Designer, Tech Lead, Developers, QA Specialists, and Systems Analysts. It demonstrates how Agile workflows and CI/CD pipelines align with PMBOK-based PM processes, how incremental system architecture and requirements traceability ensure precise tracking from conception to deployment, and how automated testing and DevSecOps practices balance release speed with platform reliability. Tables detail role distribution and a DevOps–PM alignment matrix, clearly illustrating which DevOps activities correspond to each PMBOK process group. These findings offer a methodological foundation for building high-performance e-commerce teams that accelerate time-to-market without sacrificing quality or scalability, and will be of interest to researchers of organizational behavior and sociotechnical systems, as well as to DevOps and project managers focused on optimizing continuous integration, delivery, and product management.

**Keywords:** e-commerce, cross-functional team, DevOps, Project Management, system engineering, CI/CD, software quality.

**Introduction**

In the current environment, shaped by the rapid growth of electronic commerce (e-commerce), the ability to swiftly deploy new features while maintaining platform stability becomes a critical factor for sustaining competitive advantage. Traditional lifecycle models (Waterfall) often lead to delays during design and testing phases, making it difficult to respond promptly to changing market and user demands [2]. In contrast, Agile and DevOps methodologies are oriented toward continuous delivery of working code and close collaboration between development and operations teams, reducing time-to-market and improving product quality [1].

This article **aims** to analyze the characteristic features of the process of organizing cross-functional teams in e-commerce to ensure coordinated operation.

The **scientific novelty** lies in the formalization of communication, planning, and change management procedures.

The **author’s hypothesis** is based on the premise that DevOps·PM teams formed on the principles of self-organization and cross-functionality—equipped with unified communication protocols and collaborative tools—can reduce the "idea-to-production" cycle and improve release stability.

The **methodology** is based on a review of academic publications addressing the characteristics of organizing and managing cross-functional DevOps·PM teams in e-commerce, as well as articles focusing on communication, planning, and operational practices within such teams.

**Materials and methods**

In recent years, researchers in the academic community have increasingly emphasized the need for deeper integration of systems engineering into Agile and DevOps programs. They argue that without a holistic systems approach, organizations risk fragmenting architecture and processes, thereby reducing product delivery efficiency. A. Hira [1] notes that "traditional" DevOps practices often focus solely on CI/CD automation while overlooking system requirements and architectural decisions, which leads to increased technical debt and hinders scalability. A.A. Popov [7] emphasizes that the success of corporate strategies directly depends on the synergy between educational, scientific, and industrial engineering practices. It is the interdisciplinary approach that ensures the sustainability and adaptability of digital projects.

Existing automation practices in cloud environments are discussed in a number of empirical studies. For instance, R. Kyadasu et al. [2] describe in detail the migration to AWS and Azure using "infrastructure as code" and continuous delivery pipelines, demonstrating that standardizing Terraform and Ansible templates can reduce deployment time by 30%. In turn, Rajkumar Kyadasu, Sandhyarani Ganipaneni, Sivaprasad Nadukuru, Om Goel, Niharika Singh [3] demonstrate how Kubernetes, when combined with data stream automation tools (Spark, Kafka), enables horizontal scaling of analytical tasks without sacrificing fault tolerance. I. Karamitsos, S. Albarhami, and C. Apostolopoulos [4] further develop the topic of seamless ML pipeline integration by introducing "infrastructure as code" practices for training and deploying models, which simplifies experiment reproducibility and accelerates the feedback loop between developers and data scientists.

The transition to intelligent automation is examined by Ng K. K. H. et al. [5], who in their systematic review identify three levels of intelligence—from rule-based automation to self-learning agents—pointing to the need to align theoretical models with practical implementations in large corporations. A. K. Tyagi et al. [6] emphasize that the core of Industry 4.0 lies in the integration of IoT devices, robotics, and intelligent control systems, with the DevOps approach enabling rapid software updates in distributed production networks without shutting down production lines.

The organizational and methodological aspects of cross-functional Dev·Ops–PM teams are reflected in comparisons between Agile and Waterfall models in the following works: A. Sinha and P. Das [8] show that in the traditional SDLC, QA departments often act as "bottlenecks," whereas Agile practices allow testing to be incorporated early in the process, improving release quality and speeding up feedback cycles. D. Russo [9] develops the theme of large-scale transformations by proposing a mixed Agile Success Model, where not only processes but also change culture, leadership, and continuous learning across all organizational levels play key roles. A. Hemon-Hildgen, F. Rowe, and L. Monnier-Senicourt [10], studying the case of large European enterprises, find that job satisfaction and risk perception are directly tied to the transparency of communication, knowledge sharing, and workload management within DevOps teams.

Thus, current research demonstrates a rich diversity of methodological and technical approaches to the organization of DevOps practices and their integration into business architecture. At the same time, there is a noticeable contradiction in the literature: on the one hand, the emphasis is placed on technical automation [2, 3]; on the other hand, on cultural and organizational factors [9, 10]. However, practical aspects of collaboration between DevOps engineers and project managers in the context of e-commerce remain underexplored. There is a lack of empirical case studies on e-commerce startups, where iteration speed and distributed teams are particularly critical. Mechanisms for measuring team efficiency and coordination tools (e.g., specialized PM platforms tailored to DevOps processes), as well as the influence of industry regulations (such as payment security and data protection) on the organization of cross-functional teams, are insufficiently addressed. The framework description in this work was based exclusively on the analyzed findings of other studies.

Turning now to the theoretical foundations, it should be noted that e-commerce projects require the integration of DevOps and project management (PM) practices into a single cross-functional team capable of rapidly and reliably launching new features and promotional campaigns into production [1]. In the traditional approach, the DevOps function is responsible for continuous integration and delivery (CI/CD), while PM ensures planning, deadline control, and budget management. From a socio-technical perspective, the organizational structure must support both technical and human dimensions of software development and operation. A role matrix formalizes each position (e.g., Site Reliability Engineer, Project Manager, Solutions Architect, CI/CD Specialist, Product Owner) and maps them to DevOps processes (continuous integration, monitoring, automated deployment) and PM processes (initiation, planning, execution, monitoring and control, closure). This mapping reveals overlaps, gaps, or conflicts in responsibilities—critical in multidisciplinary teams where RACI principles (Responsible, Accountable, Consulted, Informed) ensure coordinated action.

Traditional PM processes follow phased milestones under PMBOK or PRINCE2, whereas DevOps relies on a continuous Plan–Do–Check–Act cycle for dynamic delivery. Therefore, the matrix must indicate not only who does what but also how DevOps artifacts (automation scripts, infrastructure-as-code configurations, containerization test results) integrate into PM documents (project charter, change management plan, risk register, schedule). A doctoral-level analysis goes beyond formal roles to consider how a DevOps engineer’s technical autonomy intersects with the Project Manager’s decision-making—such as reallocating resources during unexpected incidents.

Technically, the matrix is built as a multi-layered network model: strategic roles at level one (Director of Development, Director of Operations, Senior Project Manager); tactical roles at level two (Scrum Master, Release Manager, SRE, Requirements Analyst, Business Analyst); and operational roles at level three (Developer, Test Engineer, Infrastructure Automation Engineer, Support Specialist). For each level, defined activities range from architectural design to monitoring and deployment analysis. By applying Value Stream Mapping, processes are decomposed from requirement intake to production release. Each matrix cell specifies inputs, outputs, exit criteria, quality metrics (lead time, deployment frequency, change failure rate), and interdependencies—thus formalizing PM artifacts (Gantt charts, charters, risk reports) while embedding DevOps artifacts (CI/CD pipelines, canary deployments, log and metrics monitoring).

The synergy of these roles helps eliminate silos and accelerate response to market changes [3, 4]. Figure 1 below provides a visual overview of the Dev·Ops·PM team composition.

Description of the Dev·Ops·PM Team

DevOps Engineer

Project Manager (PM)

Data Analyst Quality Assurance

Quality Assurance

Marketing Liaison (Business Leader)

Fig.1. Description of the DevOps·PM team (compiled by the author based on [1, 7, 8]).

Effective organization of a cross-functional Dev·Ops·PM team is impossible without well-established communication, planning, and decision-making processes. In e-commerce environments, where promotional campaigns can generate thousands of transactions per hour, timely information exchange and flexible planning become critically important. The fundamental principles for building an effective team include:

1. Self-organization: the team independently distributes tasks and assesses risks, accelerating the decision-making process.
2. Cross-functionality: each team member has a basic understanding of adjacent roles (Dev, Ops, PM, QA, Data), which minimizes the "handoff gap" between workflow stages [10].
3. Shared goals and metrics: such as release time, fault tolerance, and ROI of promo campaigns, which align the team under a unified direction [2, 5].

In cross-functional teams, the responsibilities of roles often overlap, reducing the risks associated with manual task transitions. For instance, a business analyst works with the product owner to clarify requirements, then collaborates with the UX designer to build a prototype. The tech leadsupports testers and developers in embedding non-functional requirements (e.g., load, security), while the system analyst ensures that new functionality does not violate overarching business rules.

To illustrate this, Table 1 provides a structured summary of key roles, their responsibilities, skills, and contributions to DevOps and PM processes.

*Table 1. Roles and areas of responsibility in the cross-functional e-commerce team [1, 2, 5]*

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Key Responsibilities | Key skills | Participation in DevOps/PM Processes |
| Product Owner | Backlog prioritization, roadmap planning, stakeholder communication | User story mapping, negotiation skills | Release planning, dependency management  |
| Business Analyst | Detailed requirement gathering/documentation, writing acceptance criteria | BPMN, process modeling | Requirement clarification during sprint review |
| UX/UI designer | Prototyping, usability testing, conversion optimization | Prototyping (Figma, Sketch), usability methods | Design integration in CI, design system validation |
| Tech Lead | System architecture, code review, mentoring | Architectural patterns, leadership skills | CI/CD pipeline setup, tool selection  |
| Developers | Feature implementation, unit/integration test coverage | Programming languages, TDD | GitOps workflows, participation in release processes  |
| QA Engineers | Test case design, functional and load testing automation | Selenium, JMeter, TestRail | Building and maintaining CI test pipelines |
| System Analysts | End-to-end business process modeling, integration validation, requirement traceability | UML/BPMN, data analysis | Architecture validation, continuous integration control  |

The key points of intersection and mechanisms of interaction between DevOps and project management are outlined below. Agile ceremonies (sprint planning, daily stand-ups, reviews, retrospectives) not only provide team transparency but also align with PMBOK processes (PMI, 2017) in the following ways:

* Sprint Planning ↔ Schedule Management: detailed task estimation, creation of a comprehensive micro-schedule [1].
* Daily Stand-Ups ↔ Communications Management: prompt identification and resolution of blockers, change coordination [9].
* Sprint Review ↔ Quality Management: evaluation of completed work, demonstration of functional features, stakeholder feedback gathering [8].
* Retrospectives ↔ Risk Management: root cause analysis of issues, determination of corrective actions for the next cycle [10].

The DevOps pipeline (Continuous Integration / Continuous Delivery) automates build, testing, and deployment processes, accelerating time-to-market and reducing human error. From the PMBOK perspective, this aligns with:

* Integration Management, which ensures compatibility between system components and overall architecture [1].
* Scope & Schedule Management, where automated synchronization with the release calendar minimizes potential downtime [5].

Combining DevOps metrics (DORA metrics: Lead Time, Deployment Frequency, Change Failure Rate, Mean Time to Recovery) with PM KPIs (Budget Variance, Schedule Variance, Earned Value) enables comprehensive project control [1, 7]. In e-commerce teams, regular data exchange is practiced between DevOps tools (Jenkins, GitLab CI/CD) and PM systems (Jira, MS Project) to maintain a unified management dashboard.

Below is a summary table 2, demonstrating how key DevOps processes relate to PMBOK processes and areas, as well as emerging conclusions.

*Table 2. Compliance of DevOps practices with PMBOK processes [1].*

|  |  |  |  |
| --- | --- | --- | --- |
| DevOps Practice | Description | PMBOK Area  | Key Output  |
| Sprint Planning | Decomposition of features into user stories and tasks | Schedule Management | Sprint Backlog |
| Daily Stand-up | Status synchronization, identification of blockers | Communications Management | Issue Log, Daily Status Report |
| Sprint Review | Demonstration of completed increment, feedback collection | Quality Management | Product Increment, Customer Feedback Report |
| Sprint Retrospective | Analysis of problems and improvement proposals | Risk Management | Action Plan (Sprint Improvement Backlog) |
| Continuous Integration | Automatic build and unit tests triggered by every commit | Integration Management | Build Artifacts, CI Logs |
| Continuous Delivery | Releases are always deployment-ready | Scope & Schedule Management | Release Candidate, Deployment Pipeline Definition |
| Infrastructure as Code | Declarative description of server and network infrastructure | Resource Management | IaC Scripts, Environment Manifests |
| Automated Testing (E2E) | End-to-end testing of functional and non-functional requirements | Quality Management | Test Reports, Test Coverage Metrics |
| Monitoring & Logging | Collection of performance metrics and production errors | Risk & Communications Management | Monitoring Dashboards, Incident Reports |

This coordination between DevOps and PM processes calls for a DevOps Coordinator or Scrum Master, who monitors CI/CD pipeline synchronization with the release plan and manages data integration between version control systems, CI servers, and PM tools. This approach ensures that every step of the process is both measurable and manageable while retaining the flexibility inherent in DevOps culture.

**Results and Discussion**

As previously noted, forming an interdisciplinary team implies the inclusion of the following roles: Data Engineer, responsible for designing ETL processes and maintaining the data warehouse; Data Scientist/ML Engineer, engaged in the development and support of predictive models; DevOps Engineer, handling CI/CD setup, Infrastructure as Code implementation, and infrastructure automation; SRE/Platform Engineer, responsible for monitoring, automatic scaling, and service self-healing; PM/Scrum Master, coordinating tasks, conducting CAB sessions, and managing the release cycle; and Product Owner/Analyst, collecting business requirements and defining success metrics [2]. For comprehensive promo campaign analysis, data from marketing, sales, supply chain, and IT must be unified in a common format. Integrating systems engineering (SE) practices into DevOps workflows, along with rigorous quality assurance methods, helps strike a balance between incremental delivery and high reliability levels [1].

Unlike the classical Waterfall model, in which most architectural work is completed at the outset, the Agile/DevOps approach distributes system design efforts more evenly throughout the project lifecycle. This is achieved through:

* Architecture Runway – maintaining a "track" for future changes so that new features can be smoothly integrated into the existing platform.
* Incremental Architectural Reviews – regular lightweight evaluations of each microservice or module design before integration into the overall system.

The Agile methodology implies continuous refinement of requirements: user stories are clarified and expanded in each sprint, with business analysts and system analysts jointly ensuring traceability from high-level business requirements to specific code. The following practices are widely used:

* Definition of Ready / Done – clear criteria for readiness to develop and for acceptance after implementation;
* Traceability Matrix – an automated matrix linking requirements, test cases, and test results [1, 5];
* Continuous Integration – each commit triggers unit and integration tests, enabling early detection of integration issues.

The DevSecOps approach extends traditional DevOps by incorporating security practices "to the left" of the CI/CD pipeline (shift-left security). Key measures include:

* Automated Testing (unit, integration, E2E) with code coverage threshold ≥ 80% [8];
* Performance & Load Testing – regular stress testing (JMeter, Gatling) to ensure SLA compliance in response time and throughput;
* Threat Modeling & SAST/DAST – architectural threat analysis (STRIDE) and static/dynamic code analysis (OWASP ZAP, SonarQube) aligned with NIST SP 800-160 recommendations;
* Continuous Monitoring – collection of performance and incident metrics (Prometheus, Grafana) for rapid response to system degradation.

To better illustrate these concepts, Table 3 below summarizes the core practices applied in systems engineering and quality assurance.

*Table 3. Main practices of system engineering and quality assurance [1, 7, 8]*

|  |  |  |  |
| --- | --- | --- | --- |
| Practice | Description | Tools/ Methods | Expected Outcome |
| Incremental Architecture Review | Lightweight architectural assessments before integrating new modules | Architecture Runway, C4 Model | Design integrity retention |
| Definition of Ready/Done | Clear criteria for user story readiness and acceptance | Confluence, Jira | Reduced number of incomplete tasks |
| Requirement Traceability | Automated linking of requirements to tests and code | Traceability Matrix, Jira | Full coverage overview |
| Automated Testing | Unit, integration, E2E testing at each build stage | Jenkins, Selenium, Cypress | Early defect detection |
| Load Testing | Performance validation against non-functional requirements | JMeter, Gatling | SLA compliance assurance |
| Threat Modeling & SAST/DAST | Architecture threat analysis and vulnerability scanning | OWASP ZAP, SonarQube, STRIDE | Reduced security risks |
| Continuous Monitoring & Incident Management | Metric collection and visualization, automated alerts on anomalies | Prometheus, Grafana, Alertmanager | Fast system degradation response |

To ensure transparency in team coordination, it is advisable to conduct daily stand-ups, use shared boards (e.g., Jira or Azure DevOps) to track progress, hold weekly CAB sessions with pre-announced agendas, and conduct adaptive retrospectives after each release to continuously improve practices [1, 9]. Figure 1 below depicts the DevOps–PM integration lifecycle.



Fig.2. DevOps–PM integration lifecycle (compiled by the author by authors)

It is also recommended to adopt a unified tool ecosystem, which includes: GitLab or a combination of GitHub and Jenkins for source control and CI/CD management; Terraform and Ansible for infrastructure as code (IaC); Airflow or Luigi for orchestration of ETL processes; cloud data warehouses such as Snowflake or BigQuery; monitoring and alerting systems like Prometheus and Grafana; feature flag platforms and A/B testing tools (e.g., LaunchDarkly).

The following key performance indicators must be continuously tracked: MTTR (Mean Time to Recovery); Lead time from commit to production; Precision and recall of predictive models; Percentage of successful releases without rollback; Response time to critical alerts.

A culture of continuous learning and knowledge sharing is supported through: regular incident post-mortems, internal technical talks, pilot proof-of-concept (PoC) projects for evaluating new tools, and mentorship programs to accelerate onboarding and integration of new specialists.

The application of these recommendations, based on proven practices, enables a balance between release velocity and service reliability, contributing to the sustainable development of Dev·Ops–PM teams.

**Conclusion**

In the highly competitive e-commerce market, companies are under increasing pressure to accelerate the delivery of new features without compromising platform reliability and security. This paper substantiates the necessity and demonstrates practical mechanisms for forming cross-functional teams that integrate DevOps and PM approaches with systems engineering and quality assurance.

The integration of DevOps ceremonies (Sprint Planning, Daily Stand-Up, CI/CD, IaC) with PMBOK processes, reinforced by a compliance matrix, results in a synergistic effect: accelerated development cycles while maintaining control over timelines, resources, and risks. Special attention is given to the collection and alignment of DevOps metrics (DORA) and PM KPIs, as well as to the role of the DevOps Coordinator in maintaining a unified dashboard.

Overall, the integrated approach to team organization—combining the flexibility of DevOps and the structured nature of project management with incremental systems engineering and advanced quality assurance practices—enables e-commerce organizations to significantly reduce time-to-market without compromising product reliability, scalability, or security.

COMPETING INTERESTS DISCLAIMER:

Authors have declared that they have no known competing financial interests OR non-financial interests OR personal relationships that could have appeared to influence the work reported in this paper.

Disclaimer (Artificial intelligence)

Option 1:

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc.) and text-to-image generators have been used during the writing or editing of this manuscript.

Option 2:

Author(s) hereby declare that generative AI technologies such as Large Language Models, etc. have been used during the writing or editing of manuscripts. This explanation will include the name, version, model, and source of the generative AI technology and as well as all input prompts provided to the generative AI technology

Details of the AI usage are given below:

1.

2.

3.

**References**

1. Hira A. Are Agile/DevOps Programs Doing Enough Systems Engineering? //Journal of Cost Analysis and Parametrics: Volume. – 2022. – Vol. 10. – No. 3. – pp. 44-63.
2. Kyadasu R. et al. DevOps Practices for Automating Cloud Migration: A Case Study on AWS and Azure Integration //International Journal of Applied Mathematics & Statistical Sciences (IJAMSS), 2020, vol. 9, No. 4, pp. 155-188.
3. Rajkumar Kyadasu, Sandhyarani Ganipaneni, Sivaprasad Nadukuru, Om Goel, Niharika Singh Leveraging Kubernetes for Scalable Data Processing and Automation in Cloud DevOps // Iconic Research and Engineering Journals. – 2023. – Vol.7 (3). – pp. 546-571.
4. Karamitsos I., Albarhami S., Apostolopoulos C. Applying DevOps practices of continuous automation for machine learning //Information. – 2020. – Vol. 11. – No. 7. – p. 363.
5. Ng K. K. H. et al. A systematic literature review on intelligent automation: Aligning concepts from theory, practice, and future perspectives //Advanced Engineering Informatics. – 2021. – Vol. 47. – pp. 1-9.
6. A. K. Tyagi K. et al. Intelligent automation systems at the core of industry 4.0 //International conference on intelligent systems design and applications. – Cham : Springer International Publishing, 2020. – Pp. 1-18.
7. A.A. Popov А. Engineering as the key to successful business strategies: the synergy of education, science and business // Current research. – 2023. – No. 18 (148) [Electronic resource] Access mode:<https://apni.ru/article/6116-inzhiniring-kak-klyuch-k-uspeshnym-biznes-strategiyam-sinergiya-obrazovaniya-nauki-i-biznesa> (date of request: 04/25/2025)
8. Sinha A., Das P. Agile methodology vs. traditional waterfall SDLC: A case study on quality assurance process in software industry //2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech). – IEEE, 2021. – Pp. 1-4.
9. Russo D. The agile success model: a mixed-methods study of a large-scale agile transformation //ACM Transactions on Software Engineering and Methodology (TOSEM). – 2021. – Vol. 30. – No. 4. – Pp. 1-46.
10. Hemon-Hildgen A., Rowe F., Monnier-Senicourt L. Orchestrating automation and sharing in DevOps teams: a revelatory case of job satisfaction factors, risk and work conditions //European Journal of information systems, 2020, vol. 29, No. 5, pp. 474-499.