CogNexus

Multilingual, International, Peer-Reviewed, Open Access Journal

Vol. 2(1) 2026, 132-177

Evaluating the Impact of Internal Developer Platforms on Developer Productivity and DevOps Delivery

[1]Bhanu Kiran Kumar Muggalla

[1] computer information systems, University of Central Missouri

bhanukiran71@gmail.com

DOI: https://doi.org/10.63084/cognexus.v2i1.281

Abstract

Cloud-native software development has increased the tools, environments, security controls and operational decisions that teams must navigate. Internal Developer Platforms (IDPs) seek to reduce this complexity through self-service infrastructure, supported delivery paths, automation, observability and embedded governance. This study evaluates how IDP adoption relates to developer productivity and software-delivery performance, and which conditions shape those relationships. A systematic evidence-review protocol with descriptive quantitative synthesis was applied to Scopus, Web of Science, IEEE Xplore, ACM Digital Library, Google Scholar and official DORA research. The formal period was 1 January 2018 to 26 July 2026. A verified corpus of 35 unique records was screened; five pre-2018 publications were retained only as background, leaving 30 records in the synthesis. Because the original database exports were unavailable, the selection counts constitute a corpus audit rather than a complete database-retrieval flow. The 2024 DORA study reported associations of 8% higher individual productivity, 10% higher team performance and 6% higher organisational performance among platform users, alongside 8% lower delivery throughput and 14% lower change stability. One independent before-and-after service case reported a 20% build-time reduction and an 80% reduction in vulnerability-handling time, but lacked a control group. Overall, self-service and developer independence were associated with reduced workflow friction, while delivery effects remained mixed and non-causal. Organisations should introduce IDPs incrementally and evaluate satisfaction, task success, adoption, throughput and stability together.

Keywords: Internal developer platform, platform engineering, developer productivity, developer experience, DORA metrics, software-delivery performance

1. Introduction

2.1 Background

Cloud-native development has widened the technical responsibilities carried by software teams. Developers now work with microservices, containers, infrastructure as code, continuous integration and delivery pipelines, observability systems, security scanning, policy controls and multiple cloud services. Although these technologies support rapid and scalable delivery, their combined use can create fragmented workflows, repeated configuration work and frequent context switching. DevOps practices were introduced to reduce separation between development and operations, but developers may still spend considerable time managing infrastructure and delivery processes rather than working directly on product features (Leite et al., 2019; Lwakatare et al., 2019; Shahin et al., 2017).

Platform engineering has developed as an organisational and technical response to this problem. It involves building reusable capabilities that allow application teams to develop and operate software without mastering every underlying infrastructure tool. Its main product is commonly described as an Internal Developer Platform. An IDP is broader than a developer portal or user interface. It combines infrastructure orchestration, service catalogues, deployment pipelines, templates, documentation, security controls and operational information within a consistent developer experience (Ciancarini et al., 2025; van de Kamp et al., 2024).

Common IDP capabilities include self-service infrastructure, golden paths, automated deployments and embedded governance. Self-service reduces reliance on support tickets by allowing developers to provision approved resources directly. Golden paths provide tested templates for common activities, such as creating a service, configuring a pipeline or deploying an application. Automated governance places security, compliance and quality requirements within these workflows instead of relying only on manual reviews. However, golden paths should provide guidance rather than become inflexible restrictions. Teams dealing with unusual workloads may require approved alternatives when standard platform services do not fit their needs (Bayer, 2024; Dursun, 2023; Skelton & Pais, 2019).

An effective IDP is therefore treated as an internal product. Developers are its users, and platform priorities should reflect their actual workflow problems. Adoption, satisfaction, reliability and ease of use become as important as the number of technical features provided. This product-oriented approach also requires

continuous feedback and gradual expansion as the platform and its users become more mature (Mori & Kittlaus, 2024; Skelton & Pais, 2019).

2.2 Problem Statement

Organisations are investing in IDPs with the expectation that standardisation and automation will improve productivity and accelerate delivery. However, the available evidence does not show a uniformly positive outcome. The 2024 DORA study associated IDP use with 8% higher individual productivity and 10% better team performance, but also reported 8% lower delivery throughput and 14% lower change stability. These relationships were observational and should not be interpreted as proof that the platform alone caused the changes (DeBellis et al., 2024).

An IDP may reduce infrastructure friction while introducing migration work, unfamiliar abstractions or dependence on a central platform team. Its effect can therefore differ according to platform maturity, team experience, workload type and implementation quality. This uncertainty makes it difficult for organisations to determine whether an IDP is improving delivery or merely changing where effort is spent.

2.3 Research Gap

Several weaknesses remain in the current evidence. First, independent empirical studies that directly evaluate IDP outcomes are limited. Much of the available knowledge comes from technical reports, practitioner surveys, vendor material and single-organisation cases. Recent literature reviews confirm that direct, peer-reviewed platform-engineering research remains less developed than the surrounding DevOps and developer-experience literature.

Second, developer productivity is sometimes reduced to visible activity such as commits, pull requests or lines of code. These measures can be useful for understanding workflow volume, but they do not capture work quality, cognitive load, collaboration, satisfaction or the value produced. The SPACE framework and developer-experience research show that productivity is multidimensional and should combine system data with developers' reported experiences (Forsgren et al., 2021; Jaspan & Green, 2023; Noda et al., 2023).

Third, productivity and delivery performance are often assessed separately. A developer may feel more productive while deployment frequency, lead time or failure rates remain unchanged. Conversely,

automation may improve delivery metrics while increasing cognitive pressure or reducing developer control. Few studies combine SPACE, DevEx and DORA measurements within one evaluation.

Finally, positive outcomes receive more attention than migration costs, platform lock-in, restrictive standardisation and temporary performance decline. Platform maturity and developer independence are also rarely examined as factors that may explain differences between organisations. These gaps limit the conclusions that can be drawn from existing studies.

2.4 Aim

This study aims to evaluate how Internal Developer Platforms affect developer productivity and software-delivery performance, while considering platform capabilities, maturity, developer independence and implementation risks.

2.5 Research Questions

The study addresses the following questions:

- **RQ1:** How does IDP adoption affect individual and team productivity?
- **RQ2:** How does IDP adoption affect software-delivery throughput and stability?
- **RQ3:** Which IDP capabilities produce the strongest outcomes?
- **RQ4:** How do developer independence and platform maturity influence results?
- **RQ5:** What risks and trade-offs accompany IDP adoption?

2.6 Contributions

This article makes four contributions. First, it combines SPACE, DevEx and DORA measures within a single evaluation structure. Second, it separates meaningful productivity outcomes from simple coding-activity counts. Third, it examines positive, negative and mixed evidence instead of assuming that IDP adoption automatically improves performance. Finally, it develops an evaluation framework that organisations can use to assess platform capabilities, developer experience, delivery throughput, operational stability and platform maturity together.

3. Literature Review

3.1 Internal Developer Platforms

Internal Developer Platforms emerged from attempts to manage the operational complexity created by cloud computing, microservices, containers and continuous delivery. Early DevOps initiatives concentrated on collaboration between development and operations, infrastructure automation and shared responsibility for software in production. As cloud-native environments expanded, individual teams were often required to understand numerous infrastructure services, deployment tools and security requirements. Platform engineering developed as a way of packaging these responsibilities into reusable services that development teams could access when needed (Leite et al., 2019; Lwakatare et al., 2019).

Platform engineering and DevOps are related but not identical. DevOps is primarily a set of cultural and technical practices that encourages shared ownership, automation and rapid feedback. Platform engineering establishes a dedicated product and team that make these practices easier to apply across an organisation. It does not replace DevOps. Instead, it provides common capabilities through which DevOps practices can be implemented consistently at scale (Dursun, 2023; Skelton & Pais, 2019).

An internal developer portal should be distinguished from a complete IDP. A portal is primarily the interface through which developers view services, documentation, ownership information and operational status. A complete IDP also includes the underlying orchestration, infrastructure automation, delivery pipelines, access controls and policies needed to execute actions through that interface. A portal that only displays information may improve discoverability, but it cannot provide end-to-end self-service delivery without integrations to the underlying platform services (Ciancarini et al., 2025).

Golden paths are central to IDP design. They are supported workflows for frequent tasks, such as creating a microservice, provisioning a database or deploying an application. A golden path may combine a source-code template, infrastructure module, test configuration, security checks and deployment pipeline. It reduces repeated decisions and makes approved practices easier to follow. However, excessive standardisation may restrict teams with unusual technical requirements. Effective platforms therefore provide sensible defaults alongside documented exception routes (Bayer, 2024; van de Kamp et al., 2024).

Service catalogues provide a searchable record of applications, APIs, infrastructure resources, documentation and ownership information. They reduce the time spent locating technical information and help teams identify who is responsible for a service. Self-service infrastructure allows developers to request approved resources without waiting for operations personnel to complete routine tickets. Automated CI/CD then moves changes through build, test, security and deployment stages using repeatable procedures. Observability connects logs, metrics, traces and alerts to the relevant services, while policy enforcement checks configurations against security, cost and compliance requirements. Together, these capabilities can reduce manual work, although their value depends on reliability, usability and integration with existing development practices (Ciancarini et al., 2025; Cuadra et al., 2024; Srinivasan et al., 2025).

3.2 Developer Productivity

Developer productivity is difficult to represent with a single measure because software development involves technical, cognitive and collaborative work. The SPACE framework organises productivity into five dimensions: satisfaction and well-being, performance, activity, communication and collaboration, and efficiency and flow (Forsgren et al., 2021).

Satisfaction and well-being concern how developers feel about their work, tools and environment. An IDP may improve this dimension by reducing repetitive setup work, unclear processes and dependency on support tickets. However, a restrictive or unreliable platform may cause frustration and reduce developers' sense of control. Satisfaction should therefore be measured through recurring surveys rather than assumed from platform adoption.

Performance concerns the outcomes of development work, including software quality, reliability and value delivered. It is different from the amount of code produced. Cheng et al. (2022) found that code quality and technical debt were meaningfully connected to perceived productivity, showing that producing more code does not necessarily indicate better performance.

Activity includes observable actions such as commits, pull requests, builds, reviews and deployments. These measures can reveal patterns in development work, but they should not be used as direct indicators of individual value. A large number of commits may reflect unnecessary fragmentation, while fewer commits may represent difficult design or problem-solving work.

Communication and collaboration cover knowledge sharing, coordination and the ability to obtain assistance. Service catalogues, shared documentation and standard workflows may improve coordination by making ownership and operational information easier to find. Nevertheless, automation should not remove useful communication between development, operations and security teams.

Efficiency and flow describe whether developers can complete valuable work with limited interruption and delay. Relevant measures include environment-setup time, waiting time for approvals, build duration, review time and the ability to maintain concentration. Since these dimensions can move in different directions, IDP evaluation should combine developer surveys, workflow telemetry and delivery outcomes. No single metric can represent productivity fairly (Jaspan & Green, 2023; Meyer et al., 2017).

3.3 Developer Experience

The DevEx framework complements SPACE by concentrating on developers' direct experience of their working environment. It identifies feedback loops, cognitive load and flow state as three core dimensions (Noda et al., 2023).

Feedback loops refer to the time and quality of responses developers receive from tools and other people. Fast builds, automated tests, clear error messages and timely code reviews allow problems to be identified before they become expensive to correct. An IDP can shorten feedback loops by integrating these functions, but poorly designed abstraction may hide important information and make failures harder to diagnose.

Cognitive load is the mental effort required to understand tools, systems and processes. Golden paths, reusable templates and self-service infrastructure can reduce unnecessary cognitive load by limiting the number of decisions required for routine work. The aim is not to remove technical understanding, but to prevent developers from repeatedly solving infrastructure problems that have already been addressed elsewhere. Research on development environments similarly shows that difficult and error-prone configuration can damage developer experience.

Flow state describes the ability to make sustained progress with limited interruption. Slow pipelines, manual approvals and unclear ownership can break this state. An IDP may support flow when it removes waiting and context switching. However, DevEx should be measured directly because platform usage alone does

not prove that developers experience faster feedback, lower cognitive load or better flow (Forsgren et al., 2024; Greiler et al., 2023).

3.4 Software-Delivery Performance

DORA metrics assess whether software changes move through delivery systems quickly and safely. The current framework contains five measures divided into throughput and instability (DeBellis et al., 2024).

Change lead time measures the period from committing a change to its successful deployment in production. Deployment frequency records how often an application or service is deployed. Failed-deployment recovery time measures how long it takes to recover from a deployment that requires immediate intervention. Together, these three measures describe delivery throughput.

Change fail rate is the proportion of deployments that require a rollback, hotfix or another immediate correction. Deployment rework rate measures the proportion of unplanned deployments performed because of production incidents or defects. These two measures describe delivery instability. Low change fail and rework rates indicate that frequent delivery is not being achieved at the expense of reliability.

IDPs may affect these measures through standard pipelines, automated testing, approved infrastructure templates and faster access to operational information. Automated DORA collection can also provide continuous visibility into delivery changes (Rüegger et al., 2024; Sallin et al., 2021). Metrics should be calculated at application or service level because combining unrelated teams may hide differences in system age, workload and regulatory context. They should also be interpreted with SPACE and DevEx measures so that faster delivery is not mistaken for better overall productivity.

3.5 Existing Empirical Evidence

Direct empirical evidence concerning complete IDPs remains limited. Early platform-engineering publications largely proposed concepts, architectures and reference models rather than controlled evaluations. Dursun (2023) described platform engineering as a means of building requirements into delivery workflows, while van de Kamp et al. (2024) introduced a reference model for planning platform capabilities. These studies clarify platform structure but provide limited causal evidence about productivity.

More recent implementation evidence is narrower but informative. Srinivasan et al. (2025) reported that PlatFab reduced service build time from five to four minutes and vulnerability-handling time from five days to one day; the single-service, uncontrolled design limits causal and cross-organisational inference. Ghanbari et al. (2026) used action design research in a Finnish software company to develop principles for Development Environment as Code. That study supports automation of error-prone environment configuration, but it evaluated one component rather than a complete IDP and did not estimate a general platform effect.

Broader evidence remains mixed. The 2024 DORA study associated IDP use with 8% higher individual productivity and 10% higher team performance, but also with 8% lower throughput and 14% lower change stability (DeBellis et al., 2024). These estimates came from one observational industry study and are not pooled effects. Anjum's (2026) multivocal review of 88 sources likewise found growing platform adoption but limited independent and longitudinal evaluation. The evidence therefore supports conditional, not universal, claims about IDP benefits.

Table 1. Summary of previous studies on IDPs, productivity and delivery performance

Study	Method or context	Main contribution or finding	Limitation or relevance
Erich et al. (2017)	Qualitative DevOps study	Identified collaboration, automation and shared responsibility as important DevOps practices.	Examined DevOps rather than complete IDPs.
Lwakatare et al. (2019)	Multiple case study of five companies	Showed that DevOps outcomes depend on technical practices and organisational conditions.	Findings were context-dependent.
Forsgren et al. (2021)	Productivity-framework development	Established SPACE as a multidimensional approach to developer productivity.	Does not independently evaluate IDPs.
Greiler et al. (2023)	Interviews with industry developers	Developed an actionable framework for factors affecting developer experience.	Focused mainly on perceived experience.
Noda et al. (2023)	Developer-centred measurement framework	Defined feedback loops, cognitive load and flow state as core DevEx dimensions.	Requires surveys to complement system data.

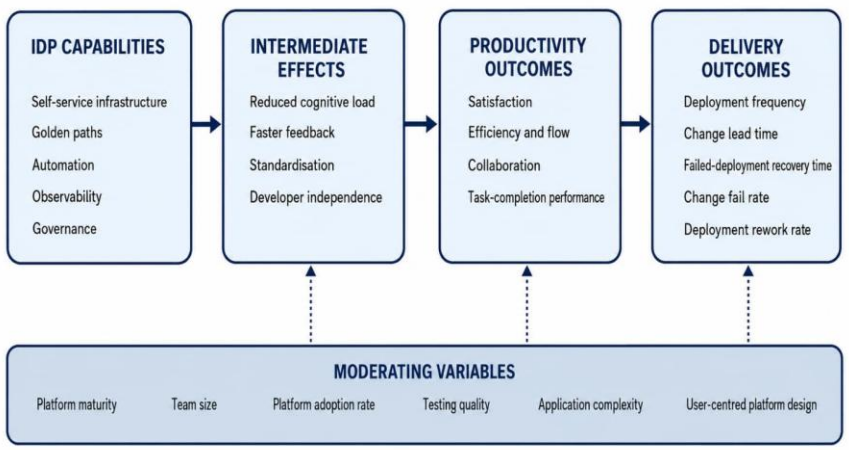
Dursun (2023)	Conceptual platform-engineering paper	Proposed building requirements and controls directly into platform workflows.	Limited empirical evaluation.
van de Kamp et al. (2024)	Reference-model development	Provided a structured model for planning platform-engineering capabilities.	Does not estimate productivity effects.
DeBellis et al. (2024)	Large international DORA survey	Found higher reported productivity but lower throughput and change stability among IDP users.	Observational evidence does not establish causation.
Rüegger et al. (2024)	Automated measurement implementation	Demonstrated automated collection of DORA metrics for continuous improvement.	Measurement capability does not itself prove improvement.
Ciancarini et al. (2025)	Design and implementation study	Demonstrated a self-hosted, open-source agile IDP supporting integrated workflows.	Evidence is mainly technical and implementation-based.
Srinivasan et al. (2025)	Single organisational implementation	Reported shorter builds, faster vulnerability handling and fewer downtime issues.	Single-service setting limits generalisation.
Ghanbari et al. (2026)	Action design research in a Finnish company	Proposed four design principles for improving environment setup through Development Environment as Code.	Addresses one component of developer self-service.
Anjum (2026)	Multivocal review of 88 sources	Found growing adoption but limited independent and longitudinal IDP evidence.	Includes peer-reviewed and grey literature of varying quality.

4. Conceptual Framework

Figure 1 presents an analytical synthesis of relationships proposed in the reviewed literature. It is not a statistically tested causal model. The arrows denote hypothesised pathways through which IDP capabilities may influence working conditions, productivity and delivery; the review evaluates the evidence available for those pathways and identifies where direct testing is absent.

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Figure 1. Conceptual framework linking IDP capabilities with productivity and software-delivery outcomes



1. The first level represents the main capabilities supplied by an IDP. Self-service enables developers to provision approved environments and services without waiting for manual intervention. Golden paths provide supported templates for recurring tasks. Automation covers building, testing, infrastructure provisioning and deployment, while observability gives developers access to logs, metrics, traces and service status. Governance embeds security, compliance and operational policies within delivery workflows. These capabilities form the technical foundation of the model (Ciancarini et al., 2025; van de Kamp et al., 2024).
2. The second level contains the mechanisms through which those capabilities are expected to work. Self-service and reusable templates may reduce cognitive load by limiting unnecessary technical decisions. Automated testing and integrated observability may provide faster feedback about build failures, security problems and production behaviour. Golden paths can improve standardisation by giving teams tested methods for common development tasks. Developer independence may increase when teams can complete routine infrastructure and deployment activities without submitting support tickets. These intermediate effects are treated as mediators because an IDP that provides many features but fails to reduce friction may not improve productivity (Greiler et al., 2023; Noda et al., 2023).
3. The third level covers productivity outcomes drawn from the SPACE and DevEx frameworks. Satisfaction concerns developers' perceptions of their tools and work environment. Efficiency

measures whether tasks can be completed with less waiting, rework and context switching. Collaboration reflects improvements in service ownership, documentation, knowledge sharing and coordination. Task-completion performance concerns the quality and timeliness of meaningful development work. These outcomes are broader than commits or lines of code and should be measured through developer surveys and workflow data (Forsgren et al., 2021; Jaspan & Green, 2023).

4. The fourth level represents software-delivery outcomes. Deployment frequency measures how often changes reach production, while change lead time measures the period between code commitment and successful deployment. Failed-deployment recovery time records how quickly teams recover from an unsuccessful deployment. Change fail rate measures the proportion of deployments requiring immediate correction, and deployment rework rate measures unplanned deployments made in response to production incidents. IDP automation may affect these measures directly through standard pipelines and indirectly through improved developer productivity (DeBellis et al., 2024; Rüegger et al., 2024).

Six variables are expected to moderate these relationships. **Platform maturity** reflects the reliability, coverage and continued improvement of platform services. **Team size** affects coordination needs and the value of shared standards. **Platform adoption rate** represents the proportion of eligible teams or workflows actively using the IDP. Low adoption may prevent organisation-wide benefits from appearing. **Testing quality** determines whether faster deployment remains stable, since automation without reliable testing may increase failures. **Application complexity** may reduce the suitability of standard golden paths, particularly for legacy or highly specialised systems. Finally, **user-centred platform design** reflects whether platform priorities are based on developer needs, feedback and observed workflow problems.

The framework therefore specifies conditional propositions rather than predictions established by this review. Mature, reliable and user-centred platforms with strong testing and broad adoption are expected to produce stronger outcomes, whereas immature or restrictive platforms may introduce delay and dependency. Platform maturity and the other moderators were not estimated consistently across the corpus; their proposed influence requires longitudinal or controlled evaluation (DeBellis et al., 2024; Anjum, 2026).

5. Methodology

5.1 Research Design

This study used a systematic evidence-review protocol with descriptive quantitative synthesis. The heterogeneous corpus includes empirical studies, implementation cases, measurement frameworks, systematic or multivocal reviews and recognised industry research. The protocol followed software-engineering review guidance and PRISMA 2020 reporting principles where the retained records permitted (Kitchenham & Charters, 2007; Page et al., 2021). Because the original database exports and retrieval totals were unavailable during revision, the study does not claim a complete PRISMA database-retrieval record; it reports a transparent audit of the verified reference corpus.

A meta-analysis was not appropriate because the records differed in design, unit of analysis, context and outcome definition. Quantitative values were therefore retained at source level. DORA regression-based associations were reported separately from independently published before-and-after case values, while qualitative and framework records were used to interpret mechanisms, moderators and measurement limits.

5.2 Data Sources

The search covered six sources:

- Scopus
- Web of Science
- IEEE Xplore
- ACM Digital Library
- Google Scholar
- Official DORA research reports

Scopus and Web of Science were selected for their broad coverage of software-engineering and information-systems research. IEEE Xplore and ACM Digital Library were included because they index major computing and software-engineering journals and conference proceedings. Google Scholar was used to identify additional academic books, technical reports and publications that might not appear consistently in the other databases.

Official DORA reports were searched separately because they provide large-scale industry evidence on developer productivity, platform engineering and software-delivery performance. DORA reports were treated as recognised industry research rather than peer-reviewed journal publications, and their methods and limitations were assessed accordingly. Reference lists of eligible review papers were also checked for relevant studies missed during database searching.

5.3 Search Period

The formal review covered records published from 1 January 2018 through 26 July 2026. The final search date was 26 July 2026. This period captures the expansion of cloud-native platform engineering, contemporary developer-experience measurement and the current DORA framework. Five pre-2018 publications were retained only for foundational or methodological background and were excluded from the 30-record formal-period synthesis.

The distinction between formal-period evidence and background sources is maintained in the corpus audit, evidence tables and supplementary matrix. It prevents earlier DevOps and review-method publications from being misrepresented as direct evaluations of contemporary IDPs.

5.4 Database-Specific Search Strategy

The common concept string combined platform terms with productivity and delivery terms. Table 2 gives the database-specific documented implementation. Quotation marks, field codes and supplementary variants were adapted to the syntax of each source without changing the two concept groups.

("internal developer platform" OR "platform engineering" OR "developer portal") AND ("developer productivity" OR "developer experience" OR "DORA metrics" OR "software delivery performance")

Supplementary variants used "developer self-service," "golden path," "platform maturity," "deployment performance" and "internal development platform." Searches targeted titles, abstracts and keywords where supported. Google Scholar was searched as three platform-term variants because its field controls differ from bibliographic databases.

The term "developer portal" was screened conservatively because it can denote an external API portal or a catalogue-only interface. Such records were retained only when they addressed internal platform

capabilities, a relevant developer measure, or software-delivery performance. Historical query exports were not retained; Table 2 therefore documents the reproducible search specification, not auditable source-by-source retrieval counts.

5.5 Inclusion Criteria

A publication was included when it satisfied the following conditions:

- It was published from 1 January 2018 through 26 July 2026.
- It was empirical research, recognised industry research, a systematic or multivocal review, or a directly relevant measurement or platform framework.
- It examined an IDP, platform engineering, a closely related self-service capability, or a construct used to evaluate developer productivity or delivery performance.
- It reported a relevant outcome or directly defined a measurement, review or capability framework used in the synthesis.
- It was written in English and available as full text.
- Its research design, data source, sample or organisational context was clearly described.
- Its findings could be traced to reported data, participant evidence or documented analysis.

Studies examining specific IDP components, such as automated CI/CD, self-service infrastructure or development environments as code, were included when they measured outcomes directly relevant to the framework.

5.6 Exclusion Criteria

Publications were excluded when they were:

- Marketing articles or vendor claims without verifiable methods or data
- Editorials, opinion pieces or informal commentaries
- Duplicate versions of the same study
- Studies that mentioned platform engineering without examining relevant outcomes
- Publications with unclear data sources, samples or analytical procedures
- Portal studies concerned only with external API users or customer-facing developer portals
- Studies published outside the formal review period

When conference and journal versions reported the same study, the more complete version was retained.

5.7 Study Selection

The retained material formed a verified, deduplicated corpus of 35 unique publications and reports. Records were matched by title, author, year and DOI where available. The author screened titles and abstracts against the eligibility criteria and then assessed the full text of formal-period records. No independent second screener was used, which increases the possibility of selection and coding bias.

Figure 2 reports only counts that can be verified from the retained corpus. The original search exports and database-specific hit totals were not available, so a complete identification-stage PRISMA flow cannot be reconstructed accurately. The figure is consequently labelled a corpus audit trail. Full-text records were assigned an evidence role - outcome evidence, implementation evidence, measurement/framework evidence, review evidence or methodological context - in Supplementary Table S1.

5.8 Data Extraction and Coding

A structured extraction form was used to record:

- Author and publication year
- Country and organisational setting
- Research design and sample
- Platform capabilities examined
- Platform maturity and adoption information
- Productivity, DevEx and DORA measures
- Baseline and post-adoption results
- Reported statistical relationships
- Positive, negative or mixed outcomes
- Study limitations and potential conflicts of interest

Productivity findings were coded according to the SPACE dimensions. Developer-experience findings were classified under feedback loops, cognitive load and flow state. Delivery outcomes were mapped to change lead time, deployment frequency, failed-deployment recovery time, change fail rate and deployment rework rate.

5.9 Quality Assessment

Peer-reviewed empirical studies were assessed for clarity of design, sample adequacy, validity of measurements, treatment of confounding variables, completeness of reporting and consistency between results and conclusions. Systematic reviews were examined for transparent search methods, selection criteria and quality assessment.

Industry reports were evaluated using authority, accuracy, coverage, objectivity, date and significance principles recommended for grey-literature assessment (Garousi et al., 2019). Evidence was classified as high, moderate or limited quality. Vendor-funded findings were not automatically excluded, but possible commercial interests were recorded and considered during interpretation.

5.10 Data Synthesis

Evidence was synthesised descriptively by platform capability, proposed mechanism, SPACE or DevEx outcome and DORA outcome. When a source reported explicit baseline and post-implementation values, percentage change was calculated only within that source. For outcomes where a higher value represented improvement, the calculation was:

Percentage change (%) = ((post-IDP value - baseline value) / baseline value) x 100

For outcomes where a lower value represented improvement, such as duration or failure measures, the calculation was:

Improvement (%) = ((baseline value - post-IDP value) / baseline value) x 100

Regression-based or survey associations were retained in the form reported by the source and were not converted into before-and-after effects. The eight principal evidence records in Table 3 were classified as positive, mixed, negative, or inconclusive/not effect-tested according to their principal contribution; that classification supports Figure 4 only. No pooled effect, median effect, Hedges' g, odds ratio or risk ratio was calculated because compatible samples and dispersion statistics were unavailable. Records without usable numerical data remained in the narrative synthesis and were not assigned artificial effect sizes.

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

5.11 Ethical Considerations

The study used publicly available secondary evidence and did not collect personal or confidential data. Formal participant consent was therefore not required. Accurate source attribution and transparent reporting were maintained throughout the review.

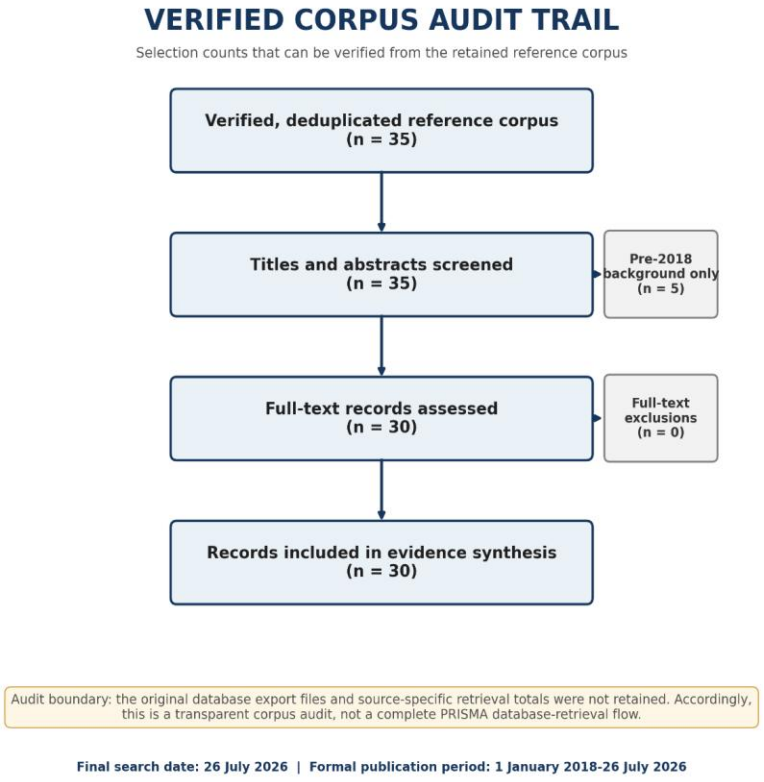
Table 2. Database-specific search strategy and eligibility criteria

Source or element	Documented search specification	Application
Scopus	TITLE-ABS-KEY(("internal developer platform" OR "platform engineering" OR "developer portal") AND ("developer productivity" OR "developer experience" OR "DORA metrics" OR "software delivery performance"))	English; formal period; title/abstract/keywords
Web of Science	TS=(("internal developer platform" OR "platform engineering" OR "developer portal") AND ("developer productivity" OR "developer experience" OR "DORA metrics" OR "software delivery performance"))	English; formal period; topic field
IEEE Xplore	(("All Metadata":"internal developer platform" OR "All Metadata":"platform engineering" OR "All Metadata":"developer portal") AND ("All Metadata":"developer productivity" OR "All Metadata":"developer experience" OR "All Metadata":"DORA metrics" OR "All Metadata":"software delivery performance"))	Journals and conferences; formal period
ACM Digital Library	("internal developer platform" OR "platform engineering" OR "developer portal") AND ("developer productivity" OR "developer experience" OR "DORA metrics" OR "software delivery performance")	Title/abstract/keywords; formal period
Google Scholar	Run separately for "internal developer platform", "platform engineering" and "developer portal", each combined with ("developer productivity" OR "developer experience" OR "DORA metrics" OR "software delivery performance").	Supplementary academic and book discovery
Official DORA	Site searches for "internal developer platform", "platform engineering", "developer productivity" and "software delivery performance".	Recognised industry research; methods appraised
Final search date	26 July 2026	Formal publication period: 1 January 2018-26 July 2026

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Inclusion	Full-text English record; relevant IDP/platform capability or evaluation construct; empirical, recognised industry, review, or directly relevant framework; traceable method or evidentiary role.	Outcome and framework roles coded separately
Exclusion	Unverifiable marketing claim, opinion/editorial, duplicate, external-only portal, irrelevant outcome, unclear source or method, or publication outside the formal period.	Pre-2018 records retained only as background
Audit boundary	Original database export files and source-specific retrieval totals were not retained.	Counts are a verified corpus audit, not a complete PRISMA retrieval flow

Figure 2. Verified corpus audit trail. Raw database retrieval totals were unavailable; the figure is not a complete PRISMA database-retrieval flow.



CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

6. Results

6.1 Study Selection

The retained reference set contained 35 unique publications and reports after verification by title, author, year and DOI where available. All 35 records were screened. Five records - Kitchenham and Charters (2007), Jabbari et al. (2016), Erich et al. (2017), Meyer et al. (2017) and Shahin et al. (2017) - pre-dated the formal period and were retained only as background. The remaining 30 formal-period records were assessed in full text and included in the structured synthesis.

No formal-period record was excluded at full text because each had a defined role in the review: direct outcome evidence, implementation evidence, measurement or conceptual framework, review evidence, or methodology. These roles are separated in Supplementary Table S1 so framework and methods sources are not treated as if they estimated an IDP effect.

The original database exports and source-specific retrieval totals were not retained. Consequently, the review cannot report the number initially retrieved from each database, and Figure 2 should not be interpreted as a complete PRISMA identification flow. The defensible audit counts are:

- Records in verified, deduplicated corpus: 35
- Duplicate records remaining in the verified corpus: 0
- Titles and abstracts screened: 35
- Pre-2018 records assigned to background only: 5
- Formal-period full-text records assessed: 30
- Formal-period full-text records excluded: 0
- Records included in the structured evidence synthesis: 30

6.2 Characteristics of Included Studies

The 30 formal-period records were published between 2018 and 26 July 2026, with increased platform-engineering coverage after 2023. The corpus included global DORA data and research from India, Finland, Italy and other European or international settings. It was methodologically uneven: only a small subset evaluated a platform or closely related capability in an operating organisation, while other records supplied measurement frameworks, reviews, technical designs or contextual DevOps evidence.

Methods included cross-sectional surveys, interviews, multiple and single-case studies, action design research, technical evaluations, systematic or multivocal reviews and framework development. Units ranged from developers and delivery teams to one service, one organisation and literature corpora. These non-equivalent units were not added into a pooled sample size.

Automation, CI/CD integration and self-service infrastructure were the most common platform-related capabilities. Golden paths, portals and environment standardisation were represented in more recent records. Observability, policy enforcement and governance were frequently discussed but seldom tested as independent predictors. Table 3 presents eight principal outcome-bearing or analytical records; Supplementary Table S1 reports all 30 records and their evidentiary roles.

Table 3. Characteristics of eight principal outcome-bearing and analytical records

Study	Country/scope and sample	Method	Capabilities	Productivity/DevEx measures	Delivery measures	Principal finding and boundary
DeBellis et al. (2024)	Global practitioner survey; platform-analysis subsample modelled	Cross-sectional survey and statistical modelling	Platform use, dedicated team, developer independence	Individual productivity; team and organisational performance	Throughput; change stability	Positive productivity associations but negative delivery associations; observational, not causal
Greiler et al. (2023)	International industry setting; 21 developers	Semi-structured interviews	Environments, tooling and work practices	Developer experience, friction and effectiveness	Not measured	Technical, organisational and social conditions shaped experience; not an IDP effect study

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Noda et al. (2023)	Multi-organisational framework evidence; no single participant sample	Literature and industry synthesis	Tooling, feedback and workflow support	Feedback loops, cognitive load and flow	Not primary	Defines DevEx measurement dimensions; does not estimate IDP effects
Srinivasan et al. (2025)	India; one industrial budgeting service	Uncontrolled before-and-after implementation case	Portal, CI/CD, containers and infrastructure automation	Build time; vulnerability-handling time	Production-downtime proxy	20% shorter build and 80% shorter vulnerability handling; one service and no control
Ghanbari et al. (2026)	Finland; one software-company setting	Action design research	Development Environment as Code	Setup effort, errors and developer experience	Not measured	Design principles reduced manual configuration; narrower than a complete IDP
Rüegger et al. (2024)	Multi-institutional; telemetry implementation	Design and technical evaluation	Automated delivery-data collection	Continuous-improvement support	DORA metric collection	Shows measurement feasibility, not improvement caused by an IDP
Ciancarini et al. (2025)	Italy; technical prototype	Design and implementation study	Open-source IDP, portal and integrated workflows	Workflow support	Integrated delivery workflow	Demonstrates technical feasibility; no causal productivity estimate

Anjum (2026)	International literature; 88 sources	Multivocal literature review	Platform engineering and internal developer portals	Productivity and DevEx outcomes	Delivery outcomes	Finds growing adoption but sparse independent and longitudinal evidence

Note. Table 3 is a concise presentation of the eight records used for the directional chart. It is not the complete corpus. Supplementary Table S1 presents all 30 formal-period records, their study type, context, capabilities, outcomes, principal finding or analytical role, and quality appraisal.

6.3 Quantitative Findings by Evidence Source

The quantitative findings came from two non-comparable sources: one large observational DORA study and one independently published, uncontrolled service case. They are reported separately and are not pooled.

6.3.1 DORA-reported associations

The 2024 DORA study reported 8% higher individual productivity, 10% higher team performance and 6% higher organisational performance among platform users. Developer independence was associated with 5% higher productivity, and a dedicated platform team with 6% higher team productivity. In the same study, software-delivery throughput was 8% lower and change stability was 14% lower (DeBellis et al., 2024).

These estimates form one association pattern from one observational source. They should not be read as seven independent replications. The phrase "14% lower change stability" is the report's modelled stability association, not a 14-percentage-point increase in change failure rate. Culture, architecture, testing, maturity and leadership may confound both platform use and outcomes.

DORA's interpretation therefore supports balanced monitoring of developer satisfaction, adoption, task success and delivery performance, rather than a causal claim that installing a platform produces each percentage change (DeBellis et al., 2024).

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Table 4. Verified quantitative findings separated by source

Evidence source	Outcome	Reported value	Interpretive boundary
DORA 2024	Individual developer productivity	8% higher	Observational association; not causal
DORA 2024	Team performance	10% higher	Observational association; not causal
DORA 2024	Organisational performance	6% higher	Observational association; not causal
DORA 2024	Productivity with developer independence	5% higher	Conditional association within the same DORA study
DORA 2024	Team productivity with a dedicated platform team	6% higher	Conditional association within the same DORA study
DORA 2024	Software-delivery throughput	8% lower	Same observational source as the productivity estimates
DORA 2024	Change stability	14% lower	Stability construct; not a percentage-point change in failure rate
Srinivasan et al. (2025)	Build duration	5 to 4 minutes (20% reduction)	One service; uncontrolled before-and-after case
Srinivasan et al. (2025)	Vulnerability-handling time	5 to 1 day (80% reduction)	One service; uncontrolled before-and-after case
Srinivasan et al. (2025)	Reported downtime incidents	12 to 'almost zero'	Endpoint is inexact; no percentage calculated

Note. DORA percentages are modelled associations from one observational industry study. Srinivasan et al. values come from one uncontrolled service case. The two sources were not combined into an effect estimate.

6.3.2 Independently reported implementation findings

Srinivasan et al. (2025) provided the only independently published before-and-after values from which percentage changes could be calculated. Build duration decreased from five to four minutes, a 20% reduction, and vulnerability-handling time decreased from five days to one day, an 80% reduction. Reported downtime incidents declined from 12 to "almost zero," but the endpoint was not exact enough to calculate

a defensible percentage. The case involved one budgeting service and no matched control, so concurrent process or staffing changes cannot be excluded.

Other independent records supplied mechanism or measurement evidence rather than comparable effect estimates. Ghanbari et al. (2026) documented reduced manual and error-prone development-environment configuration through action design research. Rüegger et al. (2024) and Sallin et al. (2021) showed that delivery metrics can be collected and operationalised, but did not demonstrate that an IDP improved those metrics. These findings strengthen the practical rationale for automation while leaving the magnitude and causality of broader IDP effects unresolved.

6.4 Influence of Developer Independence

Developer independence was a moderator reported within the 2024 DORA analysis. Developers' ability to perform lifecycle tasks without relying on an enabling team was associated with 5% higher productivity (DeBellis et al., 2024). This remains an association from the same survey, not an independently replicated causal effect.

This finding suggests that the presence of a portal or platform team is insufficient if developers must still submit tickets, wait for manual approvals or request assistance for routine deployments. Effective self-service should allow developers to create environments, deploy services, examine operational information and resolve common failures within approved boundaries. Golden paths and automated policies can support this independence by placing organisational requirements inside reusable workflows.

Independence should not mean removing expert support. Platform and enabling teams remain responsible for secure defaults, documentation, reusable services and complex exceptions. The practical objective is to remove unnecessary dependencies while preserving accessible assistance. Organisations should therefore measure independence through task-success rates, waiting time, escalation frequency and the proportion of common lifecycle tasks completed without manual intervention. This interpretation is also supported by DORA’s current guidance, which identifies independence as a central condition for obtaining value from platform engineering.

6.5 Delivery Trade-Offs

The lower throughput and change-stability outcomes reported by DORA indicate that IDP adoption may introduce short-term operational costs, even when developer productivity improves (DeBellis et al., 2024). Several factors may explain this pattern.

First, developers require time to learn new platform interfaces, deployment procedures and service templates. During this learning period, teams may complete infrastructure and deployment tasks more slowly, particularly when documentation and onboarding support are inadequate. Productivity gains may appear later, after developers become familiar with the platform.

Migration disruption can also reduce delivery performance. Moving existing applications, pipelines and infrastructure configurations to a new platform may interrupt established workflows. Development teams may need to modify application architectures, resolve compatibility problems and operate old and new systems simultaneously. These activities consume time that would otherwise be spent delivering product changes.

Golden paths may improve consistency, but overly rigid paths can create delays. A standard workflow may suit common applications while failing to accommodate legacy systems, specialised services or unusual compliance requirements. When developers cannot modify or bypass unsuitable templates, they may depend on the platform team for exceptions, reducing independence and creating another operational bottleneck.

Inadequate automated testing presents a further risk. Faster self-service provisioning and deployment do not guarantee stable releases if testing pipelines are incomplete or unreliable. A platform that accelerates changes without strengthening unit, integration, security and production-readiness tests may increase deployment failures and rework. This could partly explain why productivity gains do not automatically translate into improved change stability.

Additional governance controls can have similar effects. Automated security checks, approval gates and policy enforcement improve compliance, but poorly designed controls may extend lead time and reduce deployment frequency. Governance is most effective when policies are embedded transparently within workflows and developers receive clear feedback when a request fails.

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Finally, poor platform usability may increase cognitive load rather than reduce it. Complex interfaces, unclear error messages, unreliable documentation and fragmented workflows can force developers to seek assistance for routine tasks. In such cases, the platform adds another technical layer without removing existing complexity.

These trade-offs suggest that lower delivery performance should not be treated as an unavoidable consequence of platform engineering. Organisations should examine whether the decline results from temporary migration effects or persistent design weaknesses. DORA recommends evaluating platform performance through a balanced combination of delivery metrics, developer satisfaction, adoption, retention and task-success measures. This makes it possible to determine whether initial performance reductions improve as the platform matures or indicate deeper problems requiring redesign.

Table 5. Evidence synthesis across SPACE, DevEx and DORA outcomes

Framework	Outcome dimension	Relevant IDP capability	Evidence identified	Overall direction	Principal sources
SPACE	Satisfaction and well-being	Self-service, usable portals and reliable documentation	Reduced routine friction may improve satisfaction, although difficult interfaces can increase frustration	Positive to mixed	Forsgren et al. (2021); Greiler et al. (2023); Noda et al. (2023)
SPACE	Performance	Automation, golden paths and dedicated platform teams	Higher individual, team and organisational performance was associated with platform use	Positive	DeBellis et al. (2024)
SPACE	Activity	Automated CI/CD and deployment workflows	Platforms may increase completed deployments and reduce manual tasks, but activity counts alone do not demonstrate productivity	Mixed or context-dependent	Forsgren et al. (2021); Srinivasan et al. (2025)

SPACE	Communication and collaboration	Service catalogues, shared standards and platform-team support	Shared workflows can improve coordination, but excessive dependency on the platform team may create bottlenecks	Positive to mixed	Greiler et al. (2023); DeBellis et al. (2024)
SPACE	Efficiency and flow	Self-service infrastructure, environment automation and reusable templates	Reduced setup time, waiting and repetitive work were reported; benefits depended on usability and developer independence	Generally positive	Ghanbari et al. (2026); Srinivasan et al. (2025); DeBellis et al. (2024)
DevEx	Feedback loops	CI/CD feedback, logs, diagnostics and observability	Faster and clearer feedback supports independent problem resolution; unclear platform errors weaken the benefit	Positive when well implemented	Noda et al. (2023); Greiler et al. (2023)
DevEx	Cognitive load	Golden paths, standard environments and infrastructure abstraction	Standardisation can reduce infrastructure complexity, but rigid paths and additional tools may introduce new cognitive demands	Mixed	Noda et al. (2023); Ghanbari et al. (2026)
DevEx	Flow state	Automated provisioning, fewer handoffs and developer independence	Self-service can reduce interruptions and waiting; dependence on tickets or enabling teams disrupts flow	Generally positive	Noda et al. (2023); DeBellis et al. (2024)
DORA	Change lead time	Automated pipelines, self-	Automation may shorten delivery	Mixed	DeBellis et al. (2024);

		service provisioning and integrated approvals	time, while migration work and additional governance gates may increase it		Srinivasan et al. (2025)
DORA	Deployment frequency	Automated CI/CD and reusable deployment paths	Platforms enable repeatable deployment, but the 2024 evidence showed an overall negative throughput association	Mixed to negative during adoption	DeBellis et al. (2024)
DORA	Failed-deployment recovery time	Observability, rollback automation and standard operating procedures	Integrated diagnostics and repeatable recovery processes may reduce recovery time, although direct IDP-specific estimates remain limited	Potentially positive; limited evidence	Rüegger et al. (2024)
DORA	Change failure rate	Automated testing, policy enforcement and secure defaults	Strong testing and governance may reduce failures; incomplete testing or poorly designed controls can weaken stability	Mixed	DeBellis et al. (2024)
DORA	Deployment rework rate	Testing automation, standardisation and production feedback	Platforms may reduce unplanned corrective deployments, but direct empirical estimates remain scarce	Inconclusive	DeBellis et al. (2024); Anjum (2026)

Note. Direction is a qualitative synthesis, not a pooled effect. DORA percentages originate from one 2024 observational study; framework and implementation records are used according to the evidentiary roles shown in Supplementary Table S1.

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Figure 3. DORA-reported percentage differences associated with IDP use. Positive and negative values are associations from one study and do not establish causation.

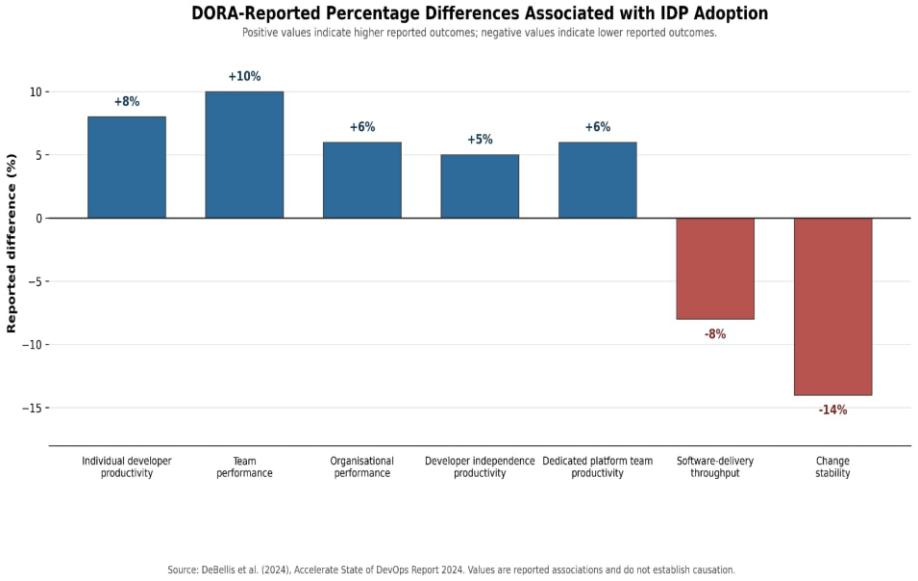
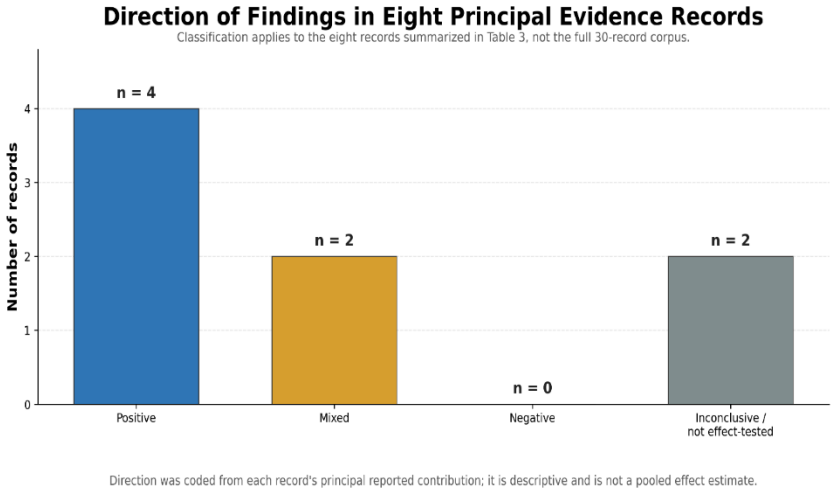


Figure 4. Direction of findings in eight principal evidence records.



Note. Four of the eight Table 3 records were classified as mainly positive, two as mixed, none as exclusively negative, and two as inconclusive or not designed to test an IDP effect. This descriptive classification is limited to the selected eight records; it is neither a count of all 30 records nor a pooled effect estimate.

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Table 6. Research-question-to-evidence map

Research question	Results location	Principal evidence	Interpretive boundary
RQ1: Individual and team productivity	Sections 6.3-6.4; Tables 4-5	DORA associations; Srinivasan case; Ghanbari action design; SPACE/DevEx studies	Associational and case evidence; no pooled causal estimate
RQ2: Delivery throughput and stability	Sections 6.3 and 6.5; Tables 4-5	DORA associations; Rüegger and Sallin measurement studies	DORA supplies the only direct platform-level delivery association
RQ3: Strongest capabilities	Sections 6.2 and 6.5; Tables 3 and 5	Self-service, automation, supported paths, observability and governance across the corpus	Frequency and mechanism synthesis; capabilities were not experimentally ranked
RQ4: Independence and maturity	Sections 6.4-6.5	DORA independence association; conceptual framework; implementation studies	Independence is associated with productivity; maturity remains a proposed moderator
RQ5: Risks and trade-offs	Section 6.5; Table 5	DORA mixed outcomes; qualitative, review and design evidence	Risks are conditional and unevenly measured

7. Discussion

7.1 Interpretation of Findings

The evidence supports a narrower conclusion than a general claim that IDPs improve productivity. In the 2024 DORA study, platform use was associated with 8% higher individual productivity, 10% higher team performance and 6% higher organisational performance (DeBellis et al., 2024). Independently, one single-service case reported shorter build and vulnerability-handling times (Srinivasan et al., 2025). Together these findings indicate plausible productivity benefits, but they do not establish a general causal effect.

IDPs may reduce cognitive load when self-service workflows and supported paths remove repeated infrastructure decisions. The DevEx and development-environment records support this mechanism conceptually and qualitatively (Noda et al., 2023; Ghanbari et al., 2026), but the corpus does not provide a controlled estimate of how much cognitive load a complete IDP removes. Poor documentation, opaque abstraction or unreliable workflows may instead add another layer of cognitive work.

Developer independence was associated with 5% higher productivity within the DORA analysis (DeBellis et al., 2024). This is consistent with the design proposition that a platform should remove routine dependencies instead of transferring them from operations to a platform team. Independent replication is still required.

Integrated testing, deployment status, logs and policy feedback may shorten feedback loops, but the delivery evidence was not uniformly positive. DORA reported lower throughput and change stability among platform users. A more convenient local workflow can therefore coexist with bottlenecks or instability elsewhere in the delivery system.

Platform maturity is a plausible explanation for heterogeneous outcomes, not a moderator formally estimated by this review. New platforms can disrupt established workflows, whereas mature platforms may improve after integrations, documentation and operating practices stabilise. Longitudinal measures are needed to test that proposition.

7.2 Relationship with Previous Research

The synthesis is consistent with the SPACE framework, which rejects a single activity measure as a complete representation of productivity and instead considers satisfaction, performance, activity, communication and efficiency (Forsgren et al., 2021). Reported developer and delivery outcomes moved in different directions, illustrating the value of multidimensional evaluation; the review does not show that all SPACE dimensions improved because of IDP adoption.

The results also agree with the DevEx framework. DevEx research identifies feedback loops, cognitive load and flow state as central conditions affecting developers' ability to perform effectively (Noda et al., 2023). Self-service provisioning and automated deployment can improve flow by reducing waiting, while golden paths can reduce the mental effort required to complete routine tasks. However, rigid templates, unclear errors and platform instability may reverse these benefits.

The findings extend DORA's previous research on DevOps automation. Deployment automation, continuous integration, version control, observability and continuous testing have long been associated with stronger delivery performance. An IDP can distribute these capabilities across multiple teams, but simply

placing them inside a portal does not guarantee their effective use. Testing quality, architecture and working practices remain important.

Recent implementation records report improvements in particular workflows, but their scope is narrow. Srinivasan et al. (2025) observed shorter build and vulnerability-handling times in one service, while Ghanbari et al. (2026) developed principles for codified development environments in one company. Anjum (2026) found that the broader literature remains dependent on grey evidence and lacks longitudinal, independent evaluation. The results should therefore be read as promising mechanisms and associations rather than conclusive platform effects.

7.3 Why Productivity Does Not Always Improve Delivery

Developers may feel more productive because they can create environments, access services and initiate deployments without submitting infrastructure tickets. Yet the resulting software must still pass through the wider organisational delivery system. Bottlenecks outside the developer's immediate workflow can prevent local productivity gains from improving overall throughput.

Testing is one possible constraint. An IDP may accelerate code deployment into a pipeline, but slow, unreliable or incomplete tests can continue to delay releases. Security approval may create another bottleneck when automated checks are followed by lengthy manual reviews. Similar problems arise when change-management procedures require multiple approvals regardless of the risk or size of a change.

Deployment governance can also reduce throughput when platform controls are too rigid or poorly integrated. Developers may complete their work faster but wait for release windows, compliance reviews or production access. Production support presents a further limitation. If operational responsibility is separated from development teams, incidents and failed deployments may still require escalation to another group.

These conditions show the difference between local and system-level optimisation. Improving one stage of development does not improve the entire delivery process when constraints remain elsewhere. Platform evaluations should therefore include end-to-end lead time, deployment frequency, recovery performance and rework, rather than relying only on developer surveys.

7.4 Technical Implications

An effective IDP requires deep integration with existing CI/CD systems. The platform should automate build, testing, security and deployment activities without forcing teams to maintain duplicate pipelines. Interoperability is equally important because organisations commonly use multiple cloud providers, repositories, monitoring tools and ticketing systems. Open interfaces and reusable APIs can reduce dependence on a single technology provider.

Scalability concerns both technical capacity and organisational reach. The platform must support increasing numbers of teams, services and deployments without producing slow response times or centralised queues. Observability should cover the platform itself and the applications deployed through it. Developers need clear logs, status information and actionable error messages to diagnose failures independently.

Policy as code can improve governance by applying security and compliance requirements consistently. However, policies should be tested, version-controlled and accompanied by clear explanations. Platform reliability is also essential. If the platform becomes unavailable, it may interrupt deployment activities across the organisation. Reliability targets, redundancy, recovery procedures and platform-specific service-level objectives should therefore be established.

7.5 Organisational Implications

For practice, an IDP should be treated as an evolving internal product. Organisations should begin with common developer journeys, preserve escape routes for atypical work, collect continuous user feedback and expand only after reliable task completion is demonstrated. Evaluation should combine SPACE and DevEx measures with the five DORA metrics and should avoid using commits or lines of code as individual targets. In summary, the evidence supports cautious investment in user-centred self-service platforms, coupled with explicit monitoring of delivery stability and stronger causal evaluation rather than an assumption of automatic improvement.

Developer feedback should shape platform priorities. Surveys, interviews, support requests and task-success data can reveal whether the platform solves real problems. Executive sponsorship is also necessary because platform engineering requires sustained investment, cross-functional cooperation and changes to established responsibilities.

Training should accompany adoption. Developers need practical guidance on platform workflows, while platform teams require technical, product-management and communication skills. Cultural change is equally important because teams must accept shared standards while retaining responsibility for application outcomes.

7.6 Trade-Offs and Risks

IDP adoption may initially reduce performance because teams must learn new processes and migrate existing services. Platform teams can also become bottlenecks when self-service functions are incomplete or exceptions require manual support. Excessive standardisation may create a “golden cage” that prevents teams from selecting appropriate technologies.

Low developer adoption represents another risk. Teams may retain unofficial tools when the platform is difficult to use, producing duplicated systems and inconsistent governance. Vendor lock-in can arise when workflows depend heavily on proprietary services or platform products.

Measurement practices also require caution. If commits, deployments or ticket counts become individual targets, developers may alter their behaviour to improve the metric rather than the underlying outcome. Finally, centralised platforms collect operational data, user activity and application metadata. Access controls, data minimisation, audit logging and privacy requirements must therefore be incorporated into platform governance.

8. Practical Implications

Organisations should begin with a minimum viable platform focused on a small number of high-value developer journeys. Attempting to deliver every possible capability at once may increase complexity and delay useful feedback. Suitable starting points include creating a new service, provisioning a development environment, deploying an application and viewing production logs.

The initial platform should address common sources of delay that affect several teams. These problems can be identified through developer interviews, workflow mapping, support-ticket analysis and delivery telemetry. Platform priorities should reflect observed developer needs rather than assumptions made by infrastructure or management teams.

Self-service workflows should allow developers to complete routine activities without waiting for another team. Provisioning, deployment, secret management and observability access can be automated within approved boundaries. Self-service does not require the removal of governance. Security and compliance requirements can be incorporated through templates, policy as code and automated checks.

Developer independence should remain a central design objective. Organisations can measure it by examining how often developers complete common lifecycle tasks without escalation, the time spent waiting for platform support and the proportion of failed self-service attempts. A platform team should provide assistance for complex situations without becoming a required intermediary for routine work.

Continuous user feedback is necessary because developer needs change as applications and technologies evolve. Short surveys, interviews, platform analytics and usability testing can reveal where developers encounter friction. Adoption alone should not be treated as evidence of value, especially when platform use is mandatory.

Measurement should combine productivity, developer experience and software-delivery performance. Productivity indicators may include task-completion time, waiting time, satisfaction and cognitive load. Delivery assessment should include change lead time, deployment frequency, failed-deployment recovery time, change failure rate and deployment rework rate. Tracking stability alongside speed reduces the risk of interpreting faster activity as better performance.

Commits, lines of code and pull-request counts should not be used as individual performance targets. These measures are easy to manipulate and do not show whether developers produced valuable, reliable or maintainable software. They may also discourage collaboration and encourage unnecessary changes.

Finally, platform adoption should be gradual. Pilot implementation with a small number of willing teams allows technical and usability problems to be corrected before wider expansion. Each adoption stage should have a baseline, defined objectives and a review period. Expansion should occur when the platform demonstrates reliable task completion and acceptable productivity and stability outcomes.

9. Limitations and Future Research

This review has six important limitations. First, it depends on secondary evidence, and only a small portion of the corpus directly evaluates a complete IDP in operation. Framework, review and methods records clarify constructs but do not estimate platform effects. Conclusions are therefore bounded by the designs and reporting choices of the underlying sources.

Second, the headline productivity and delivery percentages all originate from the same 2024 DORA observational study and include self-reported constructs. Perceptions reveal friction that telemetry can miss, but they can be affected by expectations, culture and recent experience. The percentages are associations, not independent replications or causal effects.

Third, independent IDP-specific peer-reviewed evidence remains sparse. The strongest independent numerical implementation evidence in this corpus is one uncontrolled service case, while other studies examine partial capabilities, technical feasibility or measurement. Industry and grey literature add practical coverage but may reflect sponsor, vendor or publication incentives.

Fourth, organisational heterogeneity restricts generalisation. Team size, legacy constraints, application complexity, testing maturity, governance and existing DevOps capability may alter outcomes. The corpus did not measure these moderators consistently enough to rank them or test the proposed maturity relationships.

Fifth, the evidence does not establish causation. Organisations that adopt effective platforms may already have stronger engineering culture, leadership and automation. Positive implementations are also more likely to be reported than abandoned initiatives, creating publication and industry-reporting bias.

Finally, the original source-specific search exports and retrieval totals were unavailable, so the review reports a verified corpus audit rather than a complete PRISMA identification flow. Screening and coding were performed by one author without dual-review reliability testing. Future research should preregister searches, preserve exports and use independent screeners. Longitudinal before-and-after studies should combine CI/CD telemetry, validated surveys and platform-usage data with matched control teams, and should test platform maturity, adoption mode, developer independence and application type over sufficient follow-up periods.

10. Conclusion

This review finds credible but qualified evidence that IDPs may improve aspects of developer productivity when they provide reliable self-service and allow developers to complete routine lifecycle tasks independently. The strongest platform-level quantitative results are associations from the 2024 DORA study: higher reported individual, team and organisational performance occurred alongside higher productivity associated with developer independence and dedicated platform ownership. A separate single-service case reported shorter build and vulnerability-handling times. Neither source establishes a general causal effect.

Productivity gains also did not guarantee stronger software-delivery performance. The same DORA analysis associated platform use with lower throughput and change stability. Local reductions in setup or infrastructure friction can coexist with constraints in testing, security approval, change management, governance and production support. The review therefore distinguishes developer experience from end-to-end delivery outcomes.

Self-service, supported paths, automation, observability and embedded policy offer plausible mechanisms for reducing friction and standardising routine work. Their effectiveness is likely to depend on usability, testing quality, reliability, adoption and organisational context. However, the proposed moderating role of platform maturity was not consistently measured and remains a proposition for future longitudinal research.

Organisations should treat the platform as an evolving internal product. Adoption should begin with common developer journeys, expand gradually and incorporate continuous user feedback. Evaluation should combine SPACE, DevEx and DORA measures rather than relying on commits, lines of code or a single delivery indicator. Overall, IDPs offer a credible means of improving developer productivity, but lasting value depends on whether local productivity gains are translated into reliable, end-to-end software delivery.

References

Anjum, M. A. (2026). Platform engineering and internal developer portals: A multivocal literature review. Frontiers in Computer Science, 8, 1814498. https://doi.org/10.3389/fcomp.2026.1814498

Bayer, F. (2024). How metamodeling concepts improve internal developer platforms and cloud platforms to foster business agility. In Metamodeling: Applications and Trajectories to the Future: Essays in Honor of Dimitris Karagiannis (pp. 1-18). Springer Nature Switzerland.

Cheng, L., Murphy-Hill, E., Canning, M., Jaspan, C., Green, C., Knight, A., et al. (2022). What improves developer productivity at Google? Code quality. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 1302-1313).

Ciancarini, P., Giancarlo, R., Grimaudo, G., Missiroli, M., & Xia, T. C. (2025). The design and realization of a self-hosted and open-source agile internal development platform. IEEE Access.

Cuadra, J., Hurtado, E., Sarachaga, I., Estévez, E., Casquero, O., & Armentia, A. (2024). Enabling DevOps for fog applications in the smart manufacturing domain: A model-driven based platform engineering approach. Future Generation Computer Systems, 157, 360-375.

DeBellis, D., Storer, K., Lewis, A., Good, B., Villalba, D., Maxwell, E., et al. (2024). Accelerate State of DevOps Report 2024. DORA. https://dora.dev/research/2024/dora-report/

Dursun, H. (2023). Full spec software via platform engineering: Transition from bolting-on to building-in. In Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering (pp. 172-175).

Erich, F. M., Amrit, C., & Daneva, M. (2017). A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 29(6), e1885.

Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution.

Forsgren, N., Kalliamvakou, E., Noda, A., Greiler, M., Houck, B., & Storey, M. A. (2024). DevEx in action. Communications of the ACM, 67(6), 42-51.

Forsgren, N., Storey, M. A., Maddila, C., Zimmermann, T., Houck, B., & Butler, J. (2021). The SPACE of developer productivity: There's more to it than you think. Queue, 19(1), 20-48.

Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology, 106, 101-121.

Ghanbari, H., Terimaa, T., & Koskinen, K. (2026). Using development environment as code for enhancing developer experience: An action design research study. Journal of Systems and Software, 236, 112803. https://doi.org/10.1016/j.jss.2026.112803

Greiler, M., Storey, M. A., & Noda, A. (2023). An actionable framework for understanding and improving developer experience. IEEE Transactions on Software Engineering, 49(4), 1411-1425. https://doi.org/10.1109/TSE.2022.3175660

Hicks, C. M., Lee, C. S., & Ramsey, M. (2024). Developer thriving: Four sociocognitive factors that create resilient productivity on software teams. IEEE Software, 41(4), 68-77.

Jabbari, R., Bin Ali, N., Petersen, K., & Tanveer, B. (2016). What is DevOps? A systematic mapping study on definitions and practices. In Proceedings of the Scientific Workshop Proceedings of XP2016 (pp. 1-11).

Jaspan, C., & Green, C. (2023). Developer productivity for humans, part 2: A human-centered approach to developer productivity. IEEE Software, 40(1), 23-28.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report.

Kumar, A., Nadeem, M., & Shameem, M. (2025). A systematic literature review for investigating DevOps metrics to implement in software development organizations. Journal of Software: Evolution and Process, 37(1), e2733.

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. ACM Computing Surveys, 52(6), 1-35.

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., et al. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114, 217-230.

Meyer, A. N., Barton, L. E., Murphy, G. C., Zimmermann, T., & Fritz, T. (2017). The work life of developers: Activities, switches and perceived productivity. IEEE Transactions on Software Engineering, 43(12), 1178-1193.

Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. Computer Science Review, 38, 100308.

Mori, V. S., & Kittlaus, H. B. (2024). A framework for managing platforms as products in IT organizations. In International Conference on Digital Product Management (pp. 59-74). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-71515-0_6

Noda, A., Storey, M. A., Forsgren, N., & Greiler, M. (2023). DevEx: What actually drives productivity. Queue, 21(2), 35-53.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., et al. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. BMJ, 372, n71. https://doi.org/10.1136/bmj.n71

Razzaq, A., Buckley, J., Lai, Q., Yu, T., & Botterweck, G. (2024). A systematic literature review on the influence of enhanced developer experience on developers' productivity: Factors, practices, and recommendations. ACM Computing Surveys, 57(1), 1-46.

Rüegger, J., Kropp, M., Graf, S., & Anslow, C. (2024). Fully automated DORA metrics measurement for continuous improvement. In Proceedings of the 2024 International Conference on Software and Systems Processes (pp. 36-45).

Sallin, M., Kropp, M., Anslow, C., Quilty, J. W., & Meier, A. (2021). Measuring software delivery performance using the four key metrics of DevOps. In International Conference on Agile Software Development (pp. 103-119). Springer. https://doi.org/10.1007/978-3-030-78098-2_7

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Access, 5, 3909-3943.

Skelton, M., & Pais, M. (2019). Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution.

Srinivasan, V., Rajkumar, M., Santhanam, S., & Garg, A. (2025). PlatFab: A platform engineering approach to improve developer productivity. Journal of Information Systems Engineering & Business Intelligence, 11(1).

Storey, M. A., Zimmermann, T., Bird, C., Czerwonka, J., Murphy, B., & Kalliamvakou, E. (2019). Towards a theory of software developer job satisfaction and perceived productivity. IEEE Transactions on Software Engineering, 47(10), 2125-2142.

Tilak, P. Y., Yadav, V., Dharmendra, S. D., & Bolloju, N. (2020). A platform for enhancing application developer productivity using microservices and micro-frontends. In 2020 IEEE-HYDCON (pp. 1-4). IEEE.

van de Kamp, R., Bakker, K., & Zhao, Z. (2024). Paving the path towards platform engineering using a comprehensive reference model. In Enterprise Design, Operations, and Computing (pp. 177-193). Springer. https://doi.org/10.1007/978-3-031-54712-6_11

CogNexus
Multilingual, International, Peer-Reviewed, Open Access Journal
Vol. 2(1) 2026, 132-177

Supplementary Material

Supplementary Table S1. Full characteristics and quality matrix for the 30 formal-period records (1 January 2018-26 July 2026)

Source	Evidence role / study type	Context or sample	Capability / construct	Outcomes	Principal finding or role in synthesis	Quality
Anjum (2026)	Multivocal review	International; 88 sources	Platform engineering and internal developer portals	Productivity, DevEx and delivery	Growing adoption, but limited independent and longitudinal IDP evidence	Moderate
Bayer (2024)	Conceptual book chapter	No participant sample	Metamodels for internal/cloud platforms	Business agility and design	Proposes capability modelling; no outcome estimate	Limited
Cheng et al. (2022)	Empirical developer-productivity study	Google developer setting; source-specific sample	Code quality and technical debt	Perceived productivity and quality	Shows why productivity cannot be reduced to coding activity; indirect to IDPs	Moderate
Ciancarini et al. (2025)	Design and implementation	Italy; technical prototype	Self-hosted open-source IDP and portal	Workflow support and feasibility	Demonstrates integrated platform feasibility; no causal effect estimate	Limited
Cuadra et al. (2024)	Model-driven platform evaluation	Smart-manufacturing fog applications	Model-driven CI/CD and platform automation	Deployment workflow	Supports DevOps enablement in a specialised domain; not a complete-IDP effect test	Moderate
DeBellis et al. (2024)	Recognised industry observational study	Global practitioner survey; platform subsample modelled	Platform use, dedicated team and independence	Productivity, team/organisation performance, throughput and stability	Reports positive productivity and negative delivery associations; not causal	Moderate

Dursun (2023)	Conceptual conference paper	No participant sample	Built-in requirements and policy	Workflow and governance	Explains embedded governance mechanism; no empirical outcome estimate	Limited
Forsgren et al. (2018)	Industry research synthesis / book	Multi-organisation DevOps evidence	Automation, lean delivery and feedback	Delivery performance and organisation outcomes	Provides indirect DevOps-performance context rather than IDP-specific evidence	Moderate
Forsgren et al. (2021)	Measurement-framework development	Multidisciplinary research and industry evidence	SPACE productivity framework	Satisfaction, performance, activity, collaboration, efficiency	Defines multidimensional productivity measurement	Moderate
Forsgren et al. (2024)	Applied measurement article	Industry examples; no single participant sample	DevEx measurement in practice	Feedback loops, cognitive load and flow	Operationalises DevEx measurement; no IDP effect estimate	Moderate
Garousi et al. (2019)	Review-method guideline	Software-engineering grey-literature methods	Multivocal review quality	Authority, accuracy, objectivity and coverage	Supplies quality criteria; not IDP outcome evidence	High
Ghanbari et al. (2026)	Action design research	One Finnish software company	Development Environment as Code	Setup effort, errors and developer experience	Develops four design principles for codified environments; narrower than an IDP	Moderate
Greiler et al. (2023)	Qualitative interviews	21 industry developers	Development environments, tooling and work practices	Developer experience and effectiveness	Identifies technical, organisational and social factors shaping DevEx	Moderate
Hicks et al. (2024)	Empirical developer study	Software-team setting; source-specific sample	Sociocognitive team conditions	Thriving and resilient productivity	Identifies four sociocognitive factors; indirect to platform effects	Moderate

Jaspan & Green (2023)	Human-centred measurement perspective	Google engineering practice; no single sample	Productivity measurement	Human and system measures	Argues for triangulated, human-centred productivity evaluation	Moderate
Kumar et al. (2025)	Systematic literature review	DevOps-metrics literature corpus	DevOps metrics	Delivery and operational measures	Shows metric heterogeneity and implementation considerations	High
Leite et al. (2019)	Literature survey	DevOps research corpus	Automation, collaboration and shared operations	DevOps concepts and challenges	Provides platform-engineering context; indirect to IDP effects	High
Lwakatare et al. (2019)	Multiple case study	Five companies	DevOps practices and automation	Benefits, challenges and delivery context	Outcomes depend on technical and organisational conditions	Moderate
Mishra & Otaiwi (2020)	Systematic mapping	DevOps and quality literature corpus	DevOps practices	Software quality	Synthesises quality relationships; indirect to complete IDPs	High
Mori & Kittlaus (2024)	Conceptual framework / book chapter	IT-organisation context; no effect sample	Platform as a product	Governance, adoption and lifecycle	Provides product-management framework; no effect estimate	Limited
Noda et al. (2023)	Developer-centred framework synthesis	Literature and industry evidence	DevEx	Feedback loops, cognitive load and flow	Defines three DevEx dimensions and measurement logic	Moderate
Page et al. (2021)	Reporting guideline	Systematic reviews	PRISMA 2020	Search and selection transparency	Supplies reporting method; not IDP outcome evidence	High
Razzaq et al. (2024)	Systematic literature review	Developer-experience literature corpus	DevEx practices and factors	Developer productivity	Synthesises DevEx-productivity factors and recommendations	High

Rüegger et al. (2024)	Design and technical evaluation	Automated telemetry implementation	DORA data collection	Delivery metrics	Shows feasibility of continuous automated measurement, not improvement	Moderate
Sallin et al. (2021)	Industrial measurement case	Swiss Post Scrum team; 10 members, 6 survey participants	Four-key-metric implementation	Original DORA delivery metrics	Demonstrates metric operationalisation; not an IDP effect estimate	Moderate
Skelton & Pais (2019)	Practitioner book	Organisational patterns and cases	Platform team and team interaction modes	Flow and team dependencies	Provides influential organisational design guidance; no causal estimate	Limited
Srinivasan et al. (2025)	Uncontrolled implementation case	India; one budgeting service	Portal, CI/CD, containers and infrastructure automation	Build time, vulnerability handling and downtime proxy	20% shorter build and 80% shorter vulnerability handling; no control	Limited
Storey et al. (2019)	Empirical theory development	Software developers; source-specific sample	Work conditions and satisfaction	Job satisfaction and perceived productivity	Supports multidimensional human productivity measurement; indirect to IDPs	Moderate
Tilak et al. (2020)	Technical prototype	Microservice and micro-frontend platform	Developer platform and reusable services	Developer-productivity proxy	Demonstrates technical approach; limited outcome validation	Limited
van de Kamp et al. (2024)	Reference-model development	Enterprise platform-engineering context	Comprehensive capability model	Planning and maturity	Structures platform capabilities; no causal productivity estimate	Limited

Note. Quality indicates methodological transparency and fitness for the record's assigned role, not certainty of a causal IDP effect. Framework and methodology records were not counted as effect

estimates. 'Source-specific sample' is used where the full publication reports a sample but the present synthesis does not rely on its exact count.