

DevExは究極のROI

開発者満足度が紐解く  
驚異の収益化

# 自己紹介

 グンタ ブルンナー  
**Günther Brunner**

 CyberAgent since 2012

-  CTO統括室  > **Developer Productivity室**

 AI  UX  Design  Performance  
 Music  Movies  Sushi  Travel

 @gunta85

 @gunta

 dev.to/gunta

 zenn.dev/gunta

# 目次

1. DevExは究極のROI：開発者満足度が紐解く驚異の収益化

2. 自己紹介

3. 目次

4. 1. 開發生産性の現状（2024）

5. 1. 開發生産性の現状（2024） 続き

6. 2. 従来のフレームワーク: DORA

7. 2. 従来のフレームワーク: DORA Core v2

8. 2. 従来のフレームワーク: SPACE

9. 3. DevExとは何か？

10.

なぜ開発者体験が見過ごされがちか？

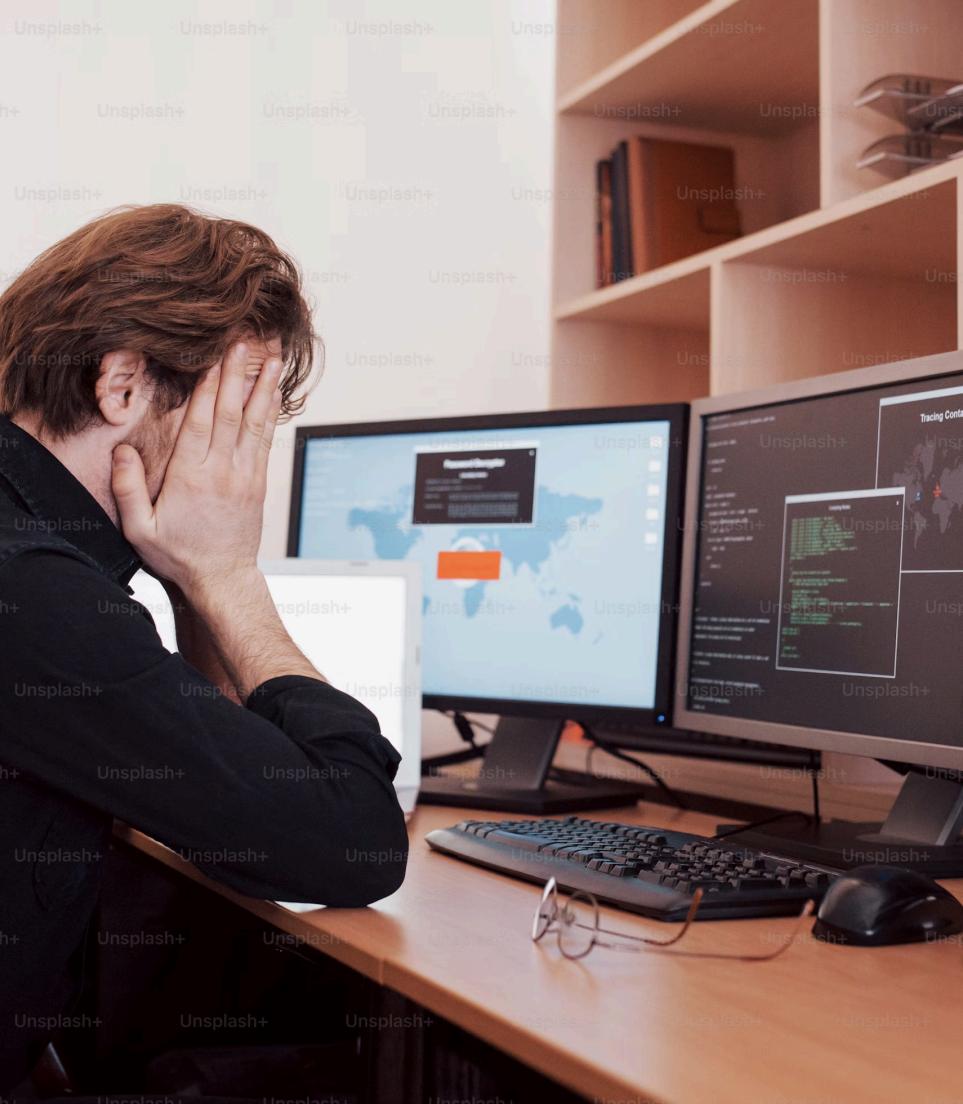
11. DevExの定義と重要性

12. DevEx論文の背景

13. DevEx論文の背景（続き）

# 1. 開發生産性の現状 (2024)

- デジタルトランスフォーメーション (DX) の加速
  - 日本企業のDX投資額が前年比20.3%増加
- 開発者の役割の重要性増大
  - ソフトウェアがビジネス成功の鍵 🔑
- 開発者体験 (DevEx) の重要性
  - 開発者の生産性と満足度に直結 😊
- 参考 : State of developer experience report 2024 -  
Atlassian & DX



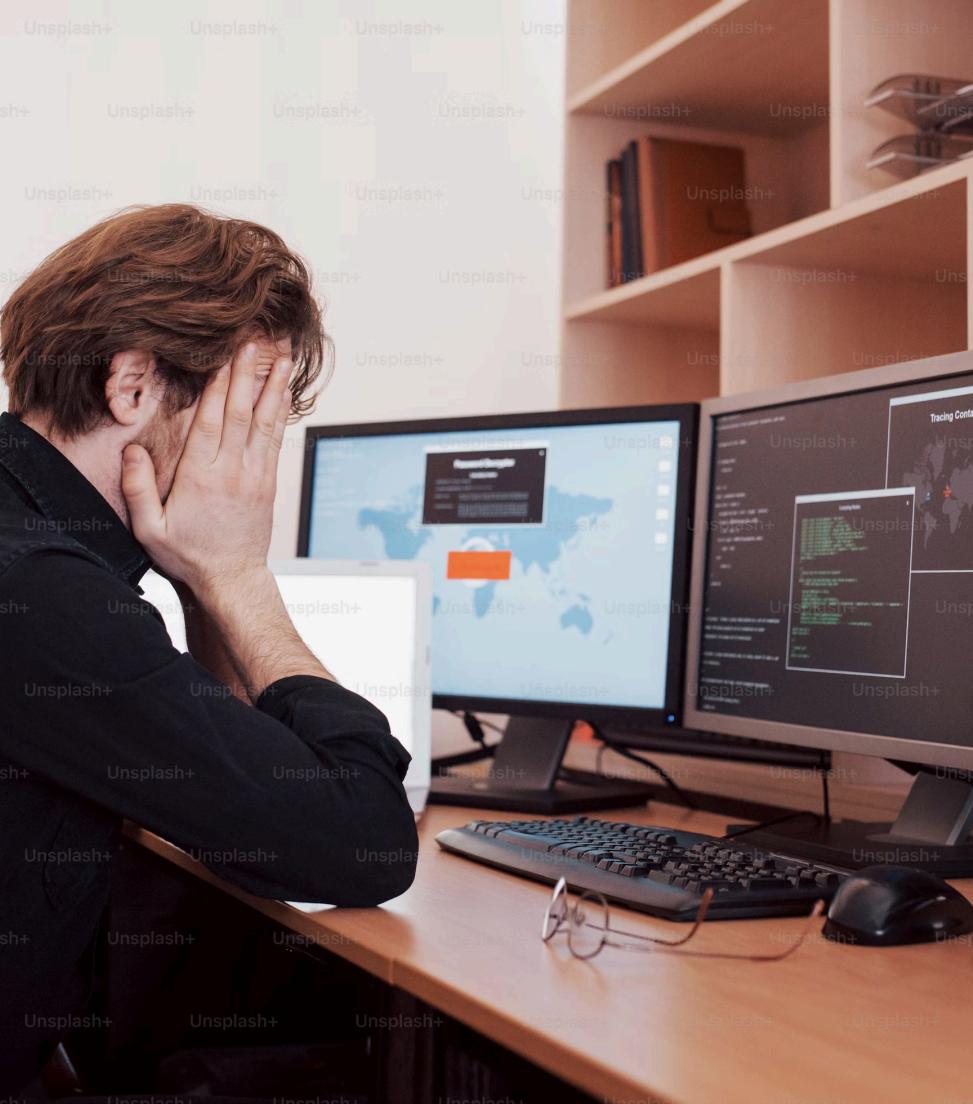
# 1. 開發生産性の現状 (2024) 続き

## ■ 開発者の課題

- **69%**が週8時間以上を**非効率的作業**に費やす ⏳
- **44%**が**技術的負債**に悩まされている 🚧
- **41%**が**不十分なドキュメンテーション**に苦労 💬

## ■ AIツールの導入状況

- **62%**がAIツールによる**生産性向上**を実感できていない 🤖
- **61%**が今後2年以内にAIツールが**生産性**を向上させると期待 💡



## 2. 従来のフレームワーク: DORA

### 主要指標 説明

デプロイ頻度 🚀

コードをプロダクション環境にデプロイする頻度

リードタイム ⏳

コミットからデプロイまでの所要時間

変更失敗率 💀

デプロイ後に修正が必要となる割合

平均復旧時間  
(MTTR) ⏳⌚

障害発生から復旧までの平均時間

### 利点 👍

- DevOpsパフォーマンスの包括的評価 🔎
- 業界ベンチマークとの比較が可能 📊

### 限界 💬

- ソフトウェア配信パフォーマンスに特化 🏁
- 開発者の日常的な課題を捉えきれない 🤷?

## 2. 従来のフレームワーク: DORA Core v2

主要要素 説明

継続的デリバリー 🚀 高頻度で安定したソフトウェアリリース

アーキテクチャ 🏢 スケーラブルで保守可能なシステム設計

製品とプロセス 💾 顧客中心の開発プロセス

リーンマネジメント 🍎 効率的な作業フローと継続的改善

文化 ☀️ 協力的で学習重視の組織文化

利点 👍

- より包括的なアプローチ ✅
- セキュリティとコンプライアンスの強化 ✅
- 組織文化と技術的実践の統合 🚀

課題 🚫

- 実装の複雑さが増加 ⚡
- 全要素のバランスを取ることが難しい 🛡

## 2. 従来のフレームワーク: SPACE

次元

説明

満足度 (S) 😊

仕事への満足度と幸福度

パフォーマンス (P) ✅

成果と貢献度

アクティビティ (A) 💡

日々の活動と行動

コミュニケーション (C)

✉️

情報共有と協力

効率性 (E) ⚡

作業のフロー状態

利点 🌟

- 開発者生産性の多面的な捉え方 📈
- 主観的要素と客観的要素の統合 🚧

課題 🚫

- 具体的な測定方法が不明確 🚫
- 組織への適用が複雑 🏙️

# 3. DevExとは何か？

Developer Experience (開発者体験) の略称

なぜ開発者体験が見過ごされがちか？

# 開発者体験が見過ごされる理由 パラダイムシフトの必要性

## グローバルな課題と日本の特殊性

### ■ 世界共通の課題

- 開発者体験の重要性軽視 😕
- 短期成果偏重 ⏳

### ■ 日本特有の文化

- 我慢の美德: 困難を美化する傾向 🌞
- 集団主義: 個人より調和重視 🧑
- 階層構造: 上意下達の文化 🏙

### ■ 日本の誤解

- 開発者の要求は「**わがまま**」ではない 🙄 ✗
- 実際は**生産性向上**のための**合理的な要求**💡
- 開発環境の改善は**組織全体の利益**につながる 🚀
- **生産性向上** ≠ **長時間労働**⌚ ✗
- ソフト開発では、**創造性**と**問題解決**が重要 💡
- **長時間労働**はむしろ**生産性を低下**させる 📈

### ■ 日本CTO協会の取り組み 🏠

- **DX Criteria**の策定
- 代表理事: **松岡 剛志**  
(CyberAgent Capital, Coincheck等)

### ■ DevExの重要性 ✨

- 持続可能な開発環境の構築 🌱
- イノベーションの促進💡
- グローバル競争力の向上 🌎

### ■ 新しい価値観 NEW

- 「**無理をする**」から「**効率的に働く**」へ ➡
- **個人の成長**が**チームの成長**につながる ↗



# DevExの定義と重要性

- 定義: 開発者が自分の仕事について考え、感じ、価値を見出す方法
- 25以上の社会技術的要因が影響

## なぜDevExが重要か

- 86%のリーダーが優秀な人材の獲得・維持にDevExが不可欠と認識 🧑
- 63%の開発者が現職継続の判断にDevExを重視 💼

## DevExの影響範囲

- 生産性向上 📈
- 従業員満足度の増加 😊
- エンゲージメントの向上 🔥
- 従業員の定着率改善 🤝

# DevEx論文の背景

## 論文概要

- タイトル: "DevEx: What Actually Drives Productivity"
- 発表日: 2023年5月3日
- 掲載: Communications of the ACM, Vol. 21 Issue 2
- リンク: <https://queue.acm.org/detail.cfm?id=3595878>

# DevEx論文の背景（続き）

## 著者と経歴

### Dr. Margaret-Anne Storey

- ビクトリア大学 CS教授
- ソフトウェア工学、HCI専門家
- SPACEフレームワーク共同著者

### Dr. Abi Noda

- DX社 創設者兼CEO
- 元GitHubのPM
- 開発者生産性専門家

### Dr. Nicole Forsgren

- GitHub リサーチ&戦略VP
- 元Google、MSFT DevOpsリード
- DORA共同創設者
- 『Accelerate』共著者

### Dr. Michaela Greiler

- ソフトウェアエンジニアリングコンサルタント
- 元Microsoft研究者
- コードレビュー、生産性専門家

# DevEx論文の重要性

-  4人の著名な研究者・実務者による共同研究
  - 異なる専門性と経験を持つ専門家の知見を結集
-  学術研究と産業実践の融合
  - 理論と実践の橋渡しを実現し、より実用的な知見を提供
-  DORA、SPACE、その他の先行研究を基に発展
  - 既存の重要な研究成果を統合し、新たな視点を加えて発展
-  開発者体験（DevEx）の包括的な理解と測定方法を提案
  - DevExの多面的な側面を捉え、実践的な改善策を提示

⚠ この論文は、ソフトウェア開発プロセスと開発者の生産性向上に関する重要な指針です。

# DevEx DX25メトリクス

## DXドライバー

バッチサイズ	ビルド	方向性
コードレビュー	コードベース	つながり
チーム協力	深い作業	ドキュメント
リリース	効率性	実験
インシデント	学習文化	フィードバック
ローカル開発	ローカル環境	技術的負債
オンコール	本番デバッグ	タイムライン
要件品質	ロードマップ	テスト範囲
テスト効率		

## DX KPI

スピード
デリバリー容易さ
品質

## 成果

組織パフォーマンス
従業員エンゲージメント
時間の無駄削減
離職率低下

## 4. DevExの3つの次元

### 1. フィードバックループ

- 定義: 行動に対する応答の速度と品質
- 例:
  -  コードレビューのターンアラウンドタイム
  -  CI結果生成時間
  -  デプロイメントリードタイム

### 2. 認知負荷

- 定義: タスク実行に必要な精神的処理量
- 例:
  -  コードベースの複雑さの認識
  -  デバッグの容易さ
  -  ドキュメンテーションの理解しやすさ

### 3. フロー状態

- 定義: 完全に没頭し、集中できる状態
- 例:
  -  中断のない深い作業時間の満足度
  -  タスクや目標の明確さ
  -  オンコール待機の中止度

### DevExの測定方法

1. **知覚指標** (開発者の態度や意見)
2. **ワークフロー指標** (システムと過程の動作)
3. **KPI** (主要業績評価指標)

# DevExがROIに与える影響 (1/5)

McKinsey  
& Company

## 調査結果

- 優れた開発者体験を提供する企業は、競合他社と比較して**4~5倍の収益成長**を達成 ✓

## 具体的な数字:

- 開発者生産性: **最大20%向上** 🚀
- 新機能開発時間: **50%短縮** ⏳
- 顧客満足度: **30%向上** 😊

ebay

### ■ DevEx改善の結果 :

- リリース頻度が**2倍**に増加 🚀
- デプロイのリードタイムが**6分の1**に短縮 ⏳

### 具体的な改善:

- CI/CDパイプラインの最適化** ⚙️
- 自動化テストの拡充** ✂️
- 開発環境の標準化** 💻

# DevExがROIに与える影響 (2/5)

## 経済的インパクトの分析

### 1. 人材コストの削減

- 500人規模の開発チームの例：
  - 週8時間の非効率性 = 年間約**9億8,000万円**のコスト 
  - DevExの改善により、このコストの**60-70%**を削減可能 

### 2. 離職率の低下によるコスト削減

- 平均的な開発者の採用・トレーニングコスト：約**426万円**
- DevEx向上により離職率が20%低下した場合：
  - 100人チームで年間**8,520万円**以上の節約 

# DevExがROIに与える影響 (3/5)

## 経済的インパクトの分析

### 3. 市場投入時間の短縮による収益増

- 新機能のリリースが1ヶ月早まった場合:
  - 中規模のSaaS企業で**10-15%の追加収益**の可能性 

### 4. バグ修正コストの削減

- 開発段階でのバグ修正コスト: **1.4万円**
- 本番環境でのバグ修正コスト: **142万円**
- DevExによる品質向上で、本番環境のバグを30%削減:
  - 1000件のバグで**3億8,340万円**の節約 

# DevExがROIに与える影響 (4/5)

## 長期的な競争力向上

### 1. イノベーション能力の向上

- **Google**の事例
  - **20%ルール**による新製品開発 
  - **Gmail**、**Google News**などの誕生
- DevExが良好な環境では、**新アイデアの提案が3倍に増加** 

### 2. 顧客満足度の向上

- **a**の事例
  - **1秒のページロード遅延で売上1%減少** 
- DevEx向上により、**パフォーマンス最適化**に注力

### 3. セキュリティの強化

-  Microsoftの例
  - DevSecOpsの導入で**セキュリティ脆弱性を40%削減** 
- 良好的なDevExにより、**セキュリティベストプラクティス**の採用が容易に 

### 4. スケーラビリティの向上

- **NETFLIX**の事例
  - **クラウドネイティブアーキテクチャの採用** 
  - **1時間あたり数百回のデプロイを実現** 
- DevEx向上により、**急成長に対応可能な柔軟な開発体制**を構築 

# DevExがROIに与える影響 (5/5)

## 費用対効果計算例

項目	投資（年間）	リターン（年間）
🔧 DevExツール導入	1,420万円	-
🎓 トレーニング	710万円	-
⚙️ プロセス改善	2,130万円	-
📈 生産性向上	-	7,100万円
🚶 離職率低下	-	2,840万円
🐛 バグ削減	-	4,260万円
💰 新規収益	-	5,680万円
📊 合計	4,260万円	1億9,880万円

$$ROI = (\text{リターン} - \text{投資}) / \text{投資} \times 100 = (198,800,000 - 42,600,000) / 42,600,000 \times 100 = 366\%$$

DevExへの投資は、1年で **3.66** 倍のリターンを生み出す。

# DevExがもたらす具体的な利益

## 1. 開発者の生産性向上 ✅

- ⚡ より多くの機能をより早くリリース
- 🔧 技術的負債の削減

## 2. イノベーションの促進 💡

- 🧠 フロー状態の増加により創造性向上
- 🚀 新製品や機能の開発が加速

## 3. 従業員の定着率向上 🤝

- 💰 採用・トレーニングコストの削減

## 4. 品質向上 ✓

- 🐞 バグの減少とセキュリティの向上
- 💰 カスタマーサポートコスト削減

## 5. ビジネスの競争力強化 🏆

- ⌚ 市場投入までの時間短縮
- 🎯 顧客ニーズへの迅速な対応

## 6. DX Core 4：統合フレームワークの紹介

# DX Core 4：最先端の統合フレームワーク

- 最も最新の開発者生産性測定フレームワーク



- 現時点では学術論文ではなく、未公開記事



- DX Core 4

- 論文化は計画中



- 著者の一人である**Abi Noda**にDMで

確認済み



Abi Noda



The DORA four keys are included in the DX Core 4 table if you look carefully.



Jul 23, 2024, 11:47 PM

Thanks! Is there a paper planned for DX Core 4?  
We would love to know about the details and the science behind it

Jul 25, 2024, 5:00 PM

Yep

Jul 28, 2024, 1:34 AM



Start a new message



# DX Core 4の概要

- DORA、SPACE、DevExを統合した新しいフレームワーク ✨
- 4つの主要次元
  1. スピード 🚗
  2. 効果 🎯
  3. 品質 ✓
  4. ビジネスインパクト 💰

## 特徴

- 多次元的アプローチ 🌎
- 組織の全レベルをサポート 🏢
- 数週間で導入可能 ⏳
- 恐怖やゲーミフィケーションを回避 ➡️

# なぜDX Core 4が必要か

- **DORAの限界:** 主にデプロイメントに焦点を当て、開発プロセス全体を捉えきれない 🔎
- **SPACEの課題:** 包括的だが、具体的な測定方法が不明確 ❓
- **DevExの弱点:** 開発者の体験に重点を置くが、ビジネス成果との関連性が弱い 📈
- **DX Core 4の利点:** 上記の限界を克服し、包括的かつ実用的なフレームワークを提供 ✓

## DX Core 4の必要性

- **統合:** 既存フレームワークを組み合わせ、開発プロセスを包括的に把握 🧩
- **実用性:** 具体的な指標で改善の方向性を明示 📈
- **ビジネス連携:** 生産性とビジネス成果を直結 🤝
- **柔軟性:** 組織に応じてカスタマイズ可能 💪

# DX Core 4の指標例



## スピード 🚗

⌚ リードタイム

🚀 デプロイ頻度

🚗 配信の知覚速度

## 効果 🎯

😎 開発者体験指数 (DXI)

📊 エンジニアあたりのDiff数

📦 配信の容易さ

## 品質 ✅

⚠️ 変更失敗率

🚑 インシデント復旧時間

😈 エンジニアあたりのインシデント数

## インパクト 💼

⌚ 新機能に費やす時間の割合

📈 イニシアチブの進捗とROI

💰 エンジニアあたりの収益 \*

\*組織レベルのみ

- **スピード**: 開発サイクルの速さを測定 ⏳
- **効果**: 開発者の生産性と体験を評価 🚀
- **品質**: ソフトウェアの信頼性と安定性を確認 🛡️
- **インパクト**: ビジネス成果への貢献度を測定 💰

### ベースラインの確立



自己報告データを活用  
システムデータの段階的な統合

### 継続的な改善



定期的なデータ収集と目標設定  
大規模な目標に対する進捗確認

## DevEx導入の ベストプラクティス



### 小規模から始める

- 共通の問題点や改善領域の特定
- 影響の大きい変更に焦点当てる



### 透明性のあるコミュニケーション

- メトリクスの収集方法と使用目的の明確化
- 全組織メンバーへの情報共有



### 開発者の参加促進

- 匿名の調査結果公開
- 改善案への投票機会の提供

## 8. まとめ：DevExが切り開く未来



- DevExは開発者の**生産性**と**満足度**を向上させる鍵 
- **3つの次元**（フィードバックループ、認知負荷、フロー状態）に注目 
- **適切な測定方法**の採用が重要 
- DevEx改善は**大きなROI**をもたらす 
- **DX Core 4**は包括的で実用的なフレームワーク 
- **AI時代**における開発者体験の**重要性**が増大 

# 次のステップ

CTO統括室 DP室 Enabling Teamの動き

## 1. 生産性測定・改善

- チーム可視化支援
- 全社効率測定

## 2. DevEx向上

- 課題特定・優先順位付け
- ケイパビリティ評価改善

## 3. 生産性コミュニティ形成

- DevExツール発見ポータル構築

## 4. ツール導入簡易化

- DevExツール申請ポータル構築

## 5. 生成AI全社活用

- ツール選定・試験導入



## カテゴリー別おすすめツール

ツールを検索...

The image displays two cards showcasing recommended tools. The left card, titled 'IDE', features a screenshot of a code editor with Python code related to Stripe API setup and SQLite database creation. The right card, titled 'プロジェクト管理' (Project Management), shows a screenshot of a project management interface with multiple windows for 'Inbox', 'Roadmap', and 'Active issues'.

**IDE**

統合開発環境で効率的なコーディング

```
# Set up Stripe API key
stripe.api_key = 'your_stripe_api_key_here'

# Connect to SQLite database (or create it if it doesn't exist)
conn = sqlite3.connect('transactions.db')
cursor = conn.cursor()

# Create table if it doesn't exist
cursor.execute('''
CREATE TABLE IF NOT EXISTS transactions (
    id TEXT PRIMARY KEY,
    ...
)
```

ツール

**プロジェクト管理**

プロジェクトの組織化と追跡のためのソリューション

Active issues × 45 + Filter

In Review × 4

Active Issues × 45 + Filter

In Review × 4

Active Issues × 45 + Filter

In Review × 4

ECM-54 Change app icon for Big Sur

CA Productivity Portal

開発 クリエイティブ Coming Soon ビジネス Coming Soon

# </> IDE比較

## IDE比較ツール比較

**Cursor**

AIを活用したコーディング支援機能を持つ次世代IDE

**長所:**

- AIによるコード補完と生成
- Githubとの統合
- 軽量で高速

**短所:**

- 比較的新しいため、機能が限定的
- 大規模プロジェクトでの使用実績が少ない

**最適な用途:** AIの支援を受けながら迅速に開発したい個人開発者や小規模チーム

セキュリティチェック: 要対応  
法務チェック: 未確認  
契約ステータス: 要対応

詳細を見る →

利用申請する

**Visual Studio Code**

Microsoftが開発した無料で多機能なコードエディタ

**長所:**

- 豊富な拡張機能
- 多言語サポート
- Gitとの連携

**短所:**

- 大規模プロジェクトで少し重くなることがある
- 設定の複雑さ

**最適な用途:** 幅広い言語やフレームワークを使用する開発者

セキュリティチェック: 要対応  
法務チェック: 未確認  
契約ステータス: 確認中

詳細を見る →

利用申請する

**IntelliJ IDEA**

JetBrainsが開発した高機能Java IDE

**長所:**

- 強力なコード分析と最適化提案
- 優れたデバッグ機能
- 豊富な統合ツール

**短所:**

- 有料版は高価
- リソース消費が大きい

**最適な用途:** Javaを中心とした大規模エンタープライズ開発

セキュリティチェック: 承認済み  
法務チェック: 承認済み  
契約ステータス: 要対応

詳細を見る →

利用申請する

Cursor - ツール詳細 | CA Productivity Portal

CA Productivity Portal 開発 クリエイティブ Coming Soon ビジネス Coming Soon

# Cursor

 Cursor  
@cursor.com

AIを活用したコーディング支援機能を持つ次世代IDE

 セキュリティチェック  
確認中  
セキュリティチームによる詳細な評価を実施中です。

 法務チェック  
承認済み  
利用規約に問題がないことを確認済みです。

 契約ステータス  
確認中  
契約条件の最終確認を行っています。

**注意事項**

- ⚠ AIが生成したコードは必ず人間がレビューしてください。
- ⚠ 機密情報をAIに入力しないよう注意してください。
- ⚠ 定期的にソフトウェアを更新し、最新のセキュリティパッチを適用してください。

**長所:**

- AIによるコード補完と生成
- GitHubとの統合
- 軽量で高速

**短所:**

- 比較的新しいため、機能が限定的
- 大規模プロジェクトでの使用実績が少ない

**最適な用途:**

AIの支援を受けながら迅速に開発したい個人開発者や小規模チーム

🚀 利用申請する

2x1

# 次のステップ

あなたが今できること

## 1. 自身の開発環境の最適化

-  効率的なツールとプラクティスの積極的な採用

## 2. チーム内でのDevEx文化の醸成

-  ベストプラクティスの共有と実践
-  DevExに関する定期的な議論の場の設定と促進

## 3. DevExメトリクスの導入と追跡

-  チームと個人レベルでの客観的な測定
-  データに基づく継続的な改善サイクルの確立

## 4. 新しい技術やツールの積極的な探求

-  DevEx向上に寄与する革新的なソリューションの調査
-  小規模な実験的導入と効果検証

## 5. 組織全体へのDevExの推進と定着

-  成功事例と具体的な事業価値の共有、DevEx改善によるROIと生産性向上の実証
-  経営層へのDevExの重要性の説明と戦略的投资の要請

### マネージャーの重要な責務

DevExの重要性を組織全体に浸透させ、継続的な改善を推進することで、チームの生産性とエンジニアの満足度を向上させる。

ご清聴ありがとうございました

質問やディスカッションをお待ちしています

