

# 品質管理の基本

プロジェクト成功に不可欠な品質とは？



# 今日のゴール




- ✓ 品質とは何かを理解する
- ✓ 品質管理の考え方とプロセスを学ぶ
- ✓ 品質を高める具体的な手法を知る
- ✓ 品質を測定する「メトリクス」の概念を理解する

# 品質とは？



 **定義：**「顧客の要求を満たすこと」


 **例：**製品が壊れない、サービスが丁寧、納品物が正確



品質 = 要求の  
達成度

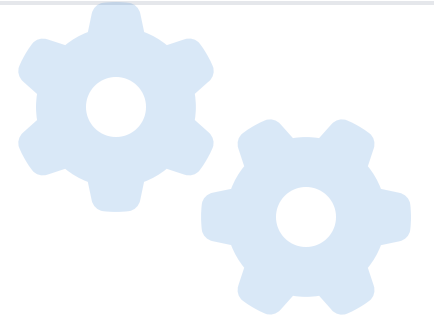
# 品質管理とは

 品質を計画・管理・保証する活動

 **目的：**不良の発生を未然に防ぎ、成果物の信頼性を高める

 **プロセス：**

品質計  
画 → 品質保  
証 → 品質管  
理






# 品質計画とは

- 🎯 品質目標を決定
- ⚙️ どのように品質を達成・評価するかを設計
- 📋 品質基準・手法・責任分担を明確に



# 品質保証とは

-  定めた品質計画が実行されているかをチェック
-  内部レビュー・監査・ガイドライン遵守
-  第三者によるレビューやテストなど



# 品質管理とは

🔍 実際の作業や成果物が基準に達しているかを監視・測定

🔧 進行中の問題を検知し、改善する活動



監視



測定



検知



改善

# 品質管理の必要性



## 顧客満足向上

高品質な製品・サービスは顧客からの信頼と継続的な取引につながる



## 再作業コストの削減

初めから品質を確保することで、修正作業や返品対応などの追加コストを削減








## 信頼性・ブランド価値の向上

長期的な競争力強化につながり、市場での評判と企業価値を高める





# 品質の観点

-  **機能性**：必要な機能を備えているか
-  **信頼性**：安定して動作するか
-  **使いやすさ**：直感的な操作が可能か
-  **保守性**：改修しやすいか
-  **パフォーマンス**：処理速度は十分か



# 品質基準の例



## ドキュメントのレビュー基準

- 誤字脱字がない
- 用語の統一がされている
- 図表が正確に参照されている

## バグの発生件数・重大度





- 重大なバグ：0件
- 中程度のバグ：1件/1000行以下
- 軽微なバグ：3件/1000行以下

## 応答時間などの性能基準

- 画面表示：2秒以内
- データ処理：5秒以内
- バッチ処理：指定時間内に完了

# メトリクスとは



-  **定義：**品質や進捗を測るための「数値指標」
-  **目的：**定量的に品質を評価・改善するため
-  **重要ポイント：**メトリクスには**下限値と上限値を設定**することが大切
-  **例：**バグ数、テストカバレッジ



バグ数：10件/100ケース → 改善目標：5件/100ケース

# よく使われるメトリクスの種類



## 欠陥密度 (バグ件数 / LOC)

コード1,000行あたりのバグ数を測定し、コード品質を評価



## 納期遵守率 (予定 vs 実績)

予定通りに完了したタスク数の割合から、プロジェクト進行の健全性を評価



## 顧客満足度 (アンケートなど)

調査による定量評価



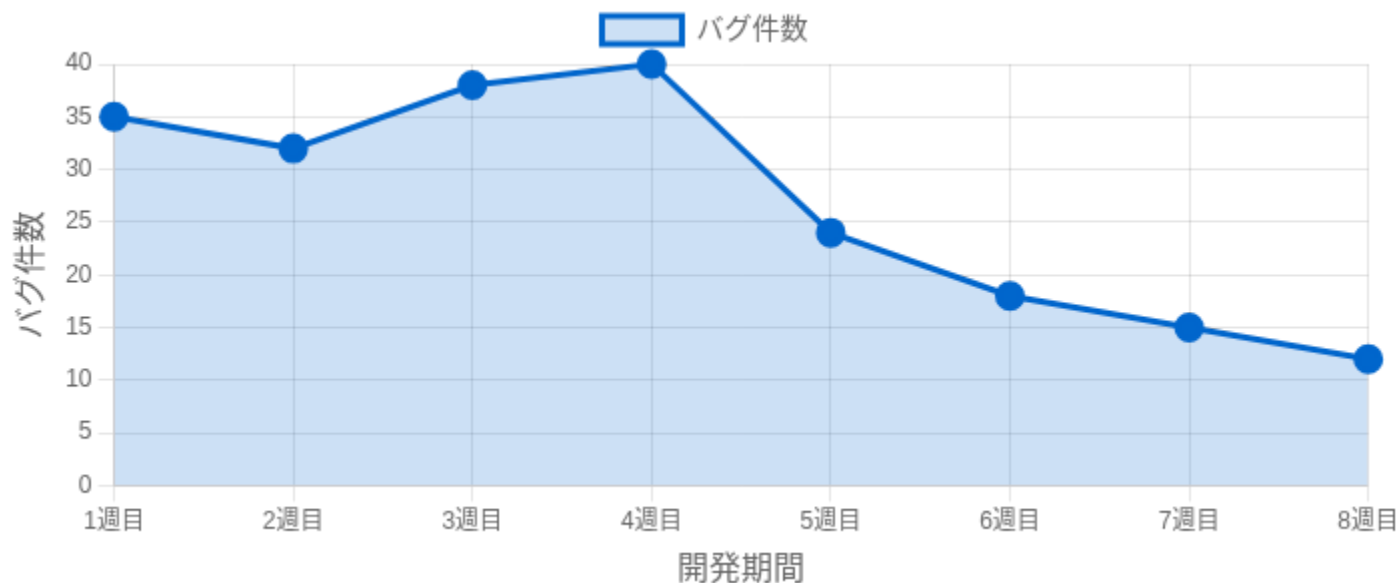
## レビュー指摘密度 (レビュー指摘数 / ページ数)

ドキュメント1ページあたりのレビュー指摘数から品質を評価



# メトリクスの活用例

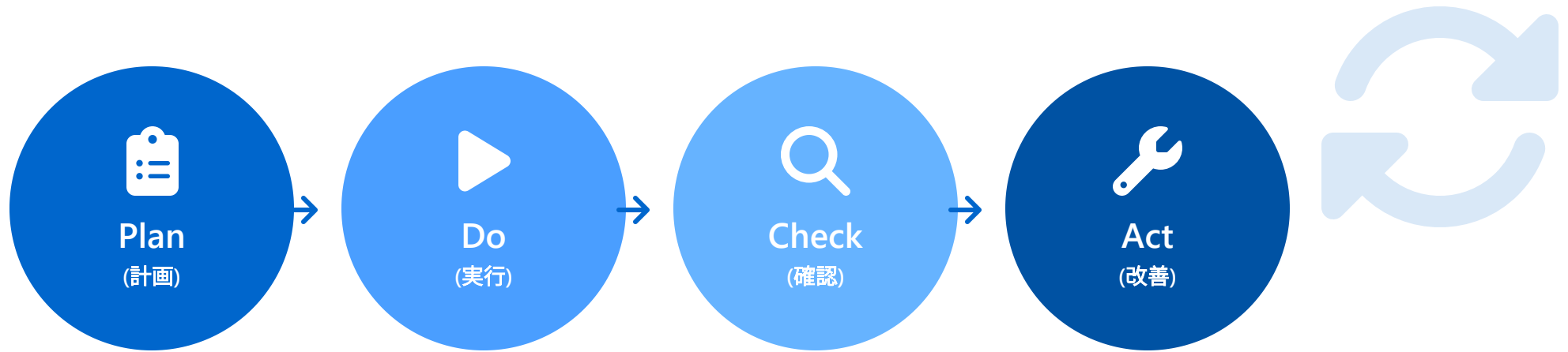
## あるWeb開発プロジェクトでの活用事例





### 改善効果

- コードレビューと自動テスト導入後、バグ数が**60%減少**
- 週次でメトリクスを可視化し、チーム全体で改善状況を共有
- 顧客満足度が向上し、追加開発の受注につながった

# PDCAサイクルと品質



 **Plan (計画)**：品質基準と手法の設定

 **Do (実行)**：開発・レビュー実施

 **Check (確認)**：成果物の品質確認

 **Act (改善)**：改善策を導入

# 品質向上の具体的手法



## コードレビュー

他のメンバーがコードをチェックし、問題点や改善点を指摘する



## テスト駆動開発（TDD）

先にテストを作成してから、それを満たすコードを開発する手法



## バグ管理ツールの活用

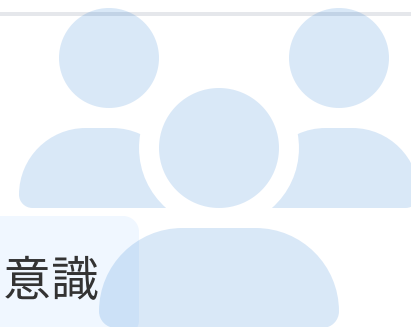
Redmine、JIRAなどで発見された問題を一元管理し解決を追跡



## ペアプログラミング

2人で1つのコードを書き、相互チェックしながら品質を高める

# 品質管理に必要な文化



## 「品質はみんなの責任」

特定の担当者だけでなく、プロジェクトに関わる全員が品質に責任を持つ意識



## 心理的安全性（指摘しやすい雰囲気）

問題点や懸念事項を自由に発言できる環境が高品質を生み出す土壌となる

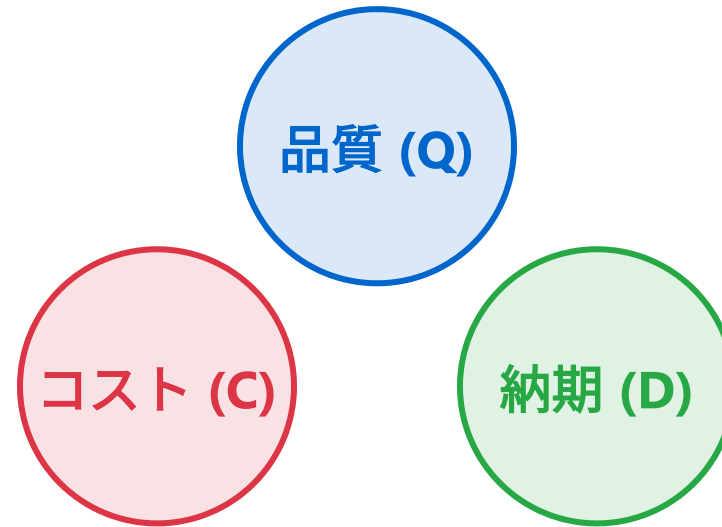


## 継続的改善の意識

常に改善の余地を探り、より良い方法を模索し続ける姿勢を組織に根付かせる



# 品質とQCDのバランス



## ⚠ 品質を上げすぎるとコスト・納期に影響

完璧を追求しすぎると、開発コストの増大や納期遅延のリスクが高まる

## 🎯 目的・要求に応じて適切な品質レベルを設定

プロジェクトの特性や顧客ニーズを考慮して、必要十分な品質目標を定める

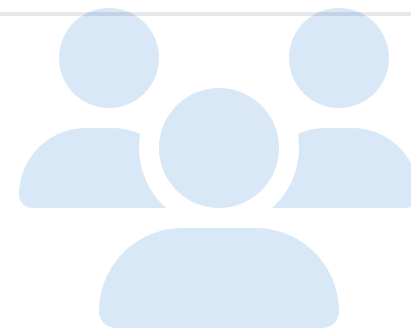
# ケーススタディ：品質問題の発生と対処

## 📁 システム開発プロジェクトの事例

- ❶ 問題：仕様理解不足により、設計に機能の漏れが発生
- ❷ 検出：設計レビューで第三者が不整合を発見
- ❸ 対応：仕様書と設計書の突合せを実施、不足機能を追加
- ❹ 恒久対策：仕様確認プロセスの徹底

- ✔ 要件定義時に仕様確認チェックリストを導入
- 👥 ステークホルダーとの定期的な仕様レビュー会議を設定

「早期発見・早期対応が重要な品質管理の鍵」



# グループワーク

# まとめ



- ★ 品質とは「期待を超える価値」
- 🔄 品質管理は計画・保証・改善のサイクル
- 📈 メトリクスを活用して品質を定量化

「高品質な成果物は、体系的な品質管理活動から生まれる」