



# mabl新機能解説

プロンプトによるテスト生成と  
ローカル/クラウド実行のシームレスな統合

2026年1月28日（水）13:00～14:00

舟木将彦（Sales Engineer, mabl）



# 本日のアジェンダ

- 
1. 品質に対するギャップの拡大
  2. テスト自動化の現状と課題
  3. AIによるテスト生成の進化
  4. 今回のテスト対象：経費管理アプリ
  5. 自然言語からテストへ（デモ含む）
  6. ローカル実行機能の活用（デモ含む）
  7. クラウドとのシームレスな連携（デモ含む）
  8. CLI/MCPを活用した開発ワークフロー
  9. ベストプラクティス



# 開発スピード vs テスト



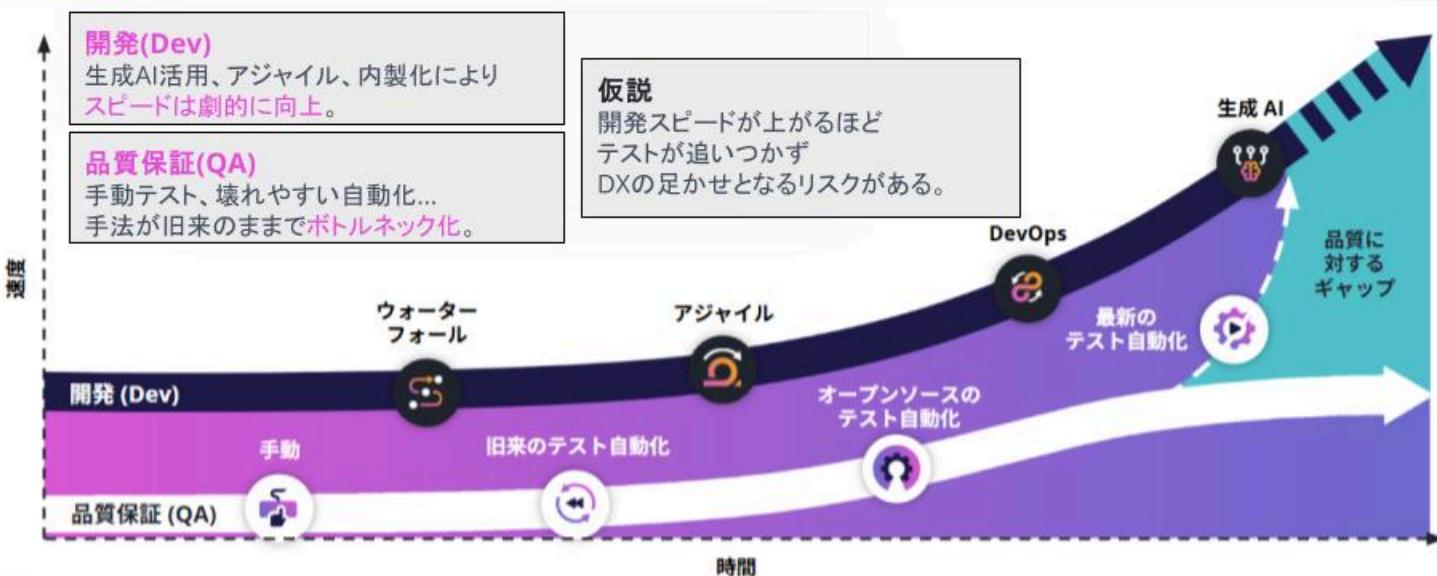
## 開発(Dev)

生成AI活用、アジャイル、内製化により  
スピードは**劇的に向上**

## 品質保証(QA)

手動テスト、壊れやすい自動化...  
手法が旧来のままで  
**ボトルネック化**

## 品質に対するギャップの拡大





# テスト自動化の現状と課題

## 従来のテスト作成の課題

- **時間がかかる:** UIテストの作成・メンテナンスに多大な工数
- **専門知識が必要:** セレクタ、待機処理、アサーションの知識
- **メンテナンス負荷:** UIの変更に追従するコスト
- **実行環境の分断:** ローカルとCI/CDで異なる設定

**結果:** テストが後回しになり、品質担保が困難に



# テスト自動化の進化

従来 → ノーコード → AI時代

観点	従来（コードベース）	ノーコード	AI時代
要素特定	手動でセレクタを記述	操作を記録して自動取得	自然言語で意図を伝える
テスト作成	コードを書いて作成	画面操作をキャプチャ	プロンプトから自動生成
検証方法	固定的なアサーション	UIベースの値チェック	AIによる柔軟な検証
実行環境	ローカル or クラウド	クラウド中心(スケーラブル)	シームレスな統合

# 今回のテスト対象：経費管理アプリ

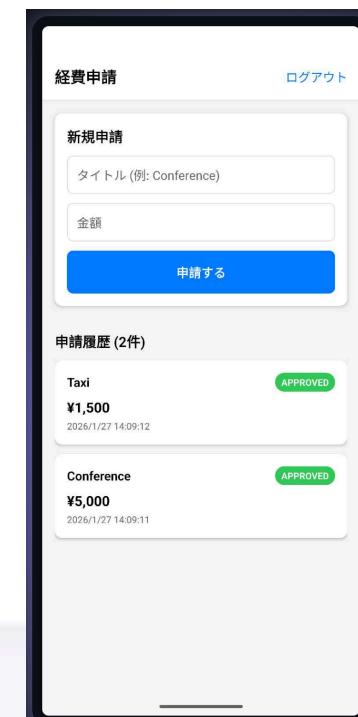
## アプリ構成

アプリ	ユーザー	機能
Web	Manager	経費一覧の閲覧・承認
Mobile	Employee	経費の申請・履歴確認
API	-	データ管理・認証

テストシナリオ: 従業員が申請 → 管理者が承認  
→ 従業員が確認

件数: 2件 (承認待ち: 0件) 合計: ¥6,500

ID	タイトル	申請者	金額	ステータス	申請日時	操作
2	Taxi	employee	¥1,500	APPROVED	2026/1/27 23:09:12	
1	Conference	employee	¥5,000	APPROVED	2026/1/27 23:09:11	





# 自然言語から実装へ

The screenshot shows a terminal window titled "my-mabl-demo-app" running on a Mac OS X system. The window displays the command:

```
git clone https://github.com/mabl/mabl-demos
```

Below the terminal, the file structure of the cloned repository is shown in a file browser:

- docs
- scripts
- tests
- artifacts
- api

  - initial-data-setup
  - mobile
  - web

- prompts
- reports
- test\_design\_improvement\_prompt... M
- test\_design\_improvement\_report... U

The "spec-expense-app.md" file is currently selected in the file browser.

The terminal output at the bottom shows:

```
~/s/g/m/my-mabl-demo-app on [?] via v25.4.0
```



# 仕様書作成のためのLLMとの壁打ち

【mabl-expenseの設計】 ▾

mablでテストするためのウェブアプリ、API、モバイルアプリをモノリソで開発したいと思います。mablでは一つのフローの中で、ウェブアプリ、API、モバイルアプリのテストを組み合わせて、情報を共有しながらテストできるので、その利点が生きるようなアプリを想定しています。モバイルアプリは、クロスプラットフォームという観点からReact NativeでExpoを使って開発したいと思います。ウェブアプリ、APIは別サービスとして(URLを共有しないで)構築したいと思います。ウェブアプリからAPIを呼び出す際に、(Dockerコンテナの別サービスのため)localhostを使わないようにしたいです。

アプリケーションは経費精算アプリケーションで、モバイルアプリに一般社員がログインして、経費を申請、ウェブアプリに上司がログインして 承認 または否決

# 自然言語からテストへ

## プロンプトによるテスト生成

mabl-expenseアプリに、管理者(manager)としてログインし、部下が申請した経費を確認した上で、未承認の申請を承認するテストを作成して

↓ mablが自動生成

- ログイン画面で manager / manager123 を入力
- ダッシュボードで経費一覧を表示
- 経費の件数(未承認件数)、合計金額をチェック
- ステータスが PENDING の経費を探す
- 承認ボタンをクリック
- ステータスが APPROVED に変わることを検証

**ポイント:** 業務フローを伝えるだけでテストが完成

**生成AIアサーション:** 件数や金額の検証もAIが自動判定

The screenshot shows the 'Expense Management App - Manager' dashboard. At the top, it displays '件数: 2件 (承認待ち: 0件) 合計: ¥6,500'. Below this is a table with two rows of expense data:

ID	タイトル	申請者	金額	ステータス	申請日時	操作
2	Taxi	employee	¥1,500	APPROVED	2026/1/27 23:09:12	
1	Conference	employee	¥5,000	APPROVED	2026/1/27 23:09:11	

The screenshot shows the 'mabl' interface for creating a new test. On the left is a sidebar with navigation options: Home, Test Creation, Test Results, Plan, Test (selected), Agents, Insights, Coverage, Accessibility, Test Environment Settings, and Workspaces. The main area is titled '新規テスト' (New Test). It contains a text input field with the following prompt:

今日は何のテストを行いますか？

mabl-expenseアプリに、管理者(manager)としてログインし、部下が申請した経費の表示件数や合計金額、個々の申請の内容を確認した上で、未承認の申請を承認するテストを作成して。出力は必ず日本語で。

生成AIはミスをする可能性があります。使用する前に確認してください。

手動で作成



# mablが生成するテストの概要

**新規テスト**

mabl-expenseアプリに、管理者(manager)としてログインし、部下が申請した経費の表示件数や合計金額、個々の申請の内容を確認した上で、未承認の申請を承認するテストを作成して。出力は必ず日本語で。

- ✓ テスト情報を更新しました
- ✓ フローを検索しました
- ✓ スニペットを検索しました
- ✓ テストアウトラインを更新しました

経費承認マネージャー承認フローテストを作成しました。指定されたmabl-expense Webアプリケーションのプロダクション環境をデプロイメントとして選択し、mabl-expense-managerの認証情報を使用します。

テストアウトラインを生成しました。以下を含む一連のタスクが含まれています。

- マネージャーアカウントでのmabl-expenseアプリへのログイン
- 経費申請一覧の表示、件数、および合計金額の確認
- 個々の未承認経費申請詳細の確認
- 未承認の経費申請を承認し、ステータス更新を検証する
- システムからのログアウト

生成されたテスト情報とテストアウトラインに何か変更を加える必要がありますか？例えば、タスクを追加したり、検証ステップを調整したりできます。

**概要**

**テスト名** 経費申請マネージャー承認フローテスト

**アプリケーション** mabl-expense

**環境** 3. production

**URL** <https://expense-app-web-ixi7x7b23a-an.a.run.app>

**クレデンシャル** mabl-expense-manager

**テストの説明** mabl-expenseアプリにマネージャーとしてログインし、部下の経費申請を承認するプロセスを検証する。

**テストのアウトライン**

**実装についての注意**

このテストは、マネージャーの視点から経費承認フローを検証します。各タスクの検証は、画面上のテキスト要素やステータス表示をアサートすることで行います。特に、経費申請の『PENDING』から『APPROVED』へのステータス変更を検証する際には、テーブル内の該当要素の`innerText`を注意深く確認してください。マネージャーのユーザー名とパスワードは、`app.defaults.username`と`app.defaults.password`のような変数で管理すると、テストの保守性が向上します。

**タスク**

- マネージャーアカウントでmabl-expenseアプリにログインする
 

**検証：**ログイン後、経費承認ダッシュボードのヘッダー「経費承認ダッシュボード」が表示されていることを確認する。

**実装についての注意：**mabl-expense-manager-loginフローを使用して、マネージャーの認証情報を入力しログインします。ログインが成功すると、「経費承認ダッシュボード」のタイトルがページに表示されることを確認します。

**提案されたフロー**

**mabl-expense-manager-login 優先**

このフローは、指定されたmabl-expense-managerアプリケーションへのログインを事前定義された資格情報で実行するため、このタスクに直接適合します。既存のテスト例もこのフローの利用を示唆しています。
- 経費申請一覧の表示、件数、および合計金額を確認する
 

**検証：**ダッシュボード上部にある件数と合計金額のテキスト要素の`innerText`を検証し、期待されるフォーマットと値に一致していることを確認します。また、テーブル内のヘッダー『ID』、『タイトル』、『申請者』、『金額』、『ステータス』、『申請日時』、『操作』が存在することを確認します。未承認（PENDING）の申請がテーブルに表示されていることを確認します。
- 個々の未承認経費申請詳細を確認する
 

**検証：**未承認の経費申請（例: Taxi）のID、タイトル、申請者、金額、ステータス、申請日時がテーブル内で正しく表示されていることを確認する。

**実装についての注意：**経費申請リストの中から、『PENDING』ステータスの申請を特定し、その行の全ての列データ（ID、タイトル、申請者、金額、ステータス、申請日時）が期待通りに表示されていることを確認します。特に『Taxi』という申請の情報を確認する際に、特定の行要素のテキストを検証します。
- 未承認の経費申請を承認する
 

未承認の経費申請（例: Taxi）の「承認」ボタンをクリックする。



# テスト生成環境の選択

The screenshot shows the mabl interface for creating a new browser test. On the left, there's a sidebar with navigation options: テスト作成 (selected), テスト結果, プラン, テスト (highlighted in blue), Agents, インサイト, カバレッジ, アクセシビリティ, テスト環境設定, and ワークスペース. A red arrow points from the "Agents" section towards the "Build approach" section. The main area is titled "新規ブラウザテスト" and includes fields for "ノフワのビューポートサイズ" (Width: 1080, Height: 1440) and "オプションの設定" (Authentication dropdown set to "mabl-expense-manager", "クレデンシャルの追加" button, and "ログインフローを追加する" checkbox). The "Build approach" section is highlighted with a red border and contains two options: "Cloud build" (selected, description: "The agent will build tests in the cloud and alert you when the test is ready or needs assistance. You can view in progress tests from the Agent page.") and "Local build" (description: "The agent will build tests on your device while you observe the trainer").



# MCPによるテスト生成

## Claude Code + mabl MCP

```
# テストの計画
mcp__mabl__plan_new_test

# テストの作成
mcp__mabl__create_mabl_test

# ローカルで実行
mcp__mabl__run_mabl_test_local

# クラウドで実行
mcp__mabl__run_mabl_test_cloud
```

**IDE内で完結:** コードを書きながらテストも作成



# ローカル実行機能の活用

## 開発中のテスト実行

- **即時フィードバック**: コード変更後すぐにテスト実行
- **デバッグ容易**: ローカル環境でステップ実行
- **コスト効率**: クラウド実行前にローカルで検証
- **オフライン対応**: ネットワーク不要で基本テスト
- **CI/CDパイプラインへの組み込みが容易**: CLIインストール/呼び出しで実行可能

```
# mabl CLIでローカル実行
mabl tests run --id <test-id> --environment <env-id> --allow-billable-features
```



# クラウドとのシームレスな連携

## ローカル → クラウドの統合

### 開発環境（ローカル）

- ↓ テスト作成・デバッグ
- ↓ ローカルで動作確認(CLI/MCPから実行)
- ↓

### クラウド実行（よりスケーラブルな実行が可能）

- ↓ 複数ブラウザ・デバイス
- ↓ 並列実行
- ↓ スケジュール実行
- ↓

### CI/CDパイプライン統合（プラン／テストを指定して実行）

同じテストをローカルでもクラウドでも実行可能



# LLM+MCP+CLIでテスト設計改善レポートの自動生成

## テスト設計改善レポート

プロジェクト名: mabl-expense (経費管理システム) レポート作成日: 2026-01-28 分析対象: Web/Mobile/APIテスト

### 1. エグゼクティブサマリー

#### 1.1 分析概要

項目	値
分析対象テスト数	6件 (Web: 4件、Mobile: 1件、API: 1件)
総ステップ数	41ステップ (Web: 22ステップ、Mobile: 11ステップ、API: 8ステップ)
カバーレッジ推定	良好 (主要フローとAPIがカバー、エッジケースに改善余地あり)
全体評価	B+ (良好、APIテスト追加により改善)

#### 1.2 主要な発見事項

##### 良好な点

- ログインから承認までのE2Eフローが適切にカバーされている

Copyright © 2026 mabl inc.

- 日本語UIの検証が適切に行われている



# CLI/MCPを活用した開発ワークフロー

## 実装とテストの並走

ステップ	開発者の作業	テスト作業
1	機能の実装	MCPでテスト計画
2	コードレビュー	テスト生成・ローカル実行
3	マージ	クラウド実行・CI統合
4	デプロイ	本番監視

**ポイント:** テストが開発のボトルネックにならない



## ベストプラクティス

### 実装とテスト作成を並走させるために

1. **data-testidを先に設計**: UI実装前にセレクタを決める
2. **小さなテストから始める**: ログイン→基本操作→複雑なフロー
3. **ローカルで頻繁に実行**: 変更のたびにテストを回す
4. **AIアサーションを活用**: 柔軟な検証で脆いテストを回避
5. **CI/CDに組み込む**: PRごとにテストを自動実行



# テスト戦略の分担

## mabl と Playwright の使い分け

テスト種別	推奨ツール	理由
経費承認フロー	<b>mabl</b>	実際のAPI・DBを検証
クロスプラットフォーム	<b>mabl</b>	Mobile + Web の統合テスト
ログイン検証	Playwright	高速・ローカル完結
UIレイアウト	Playwright	スナップショット比較
入力バリデーション	Playwright	クライアント側ロジック



# まとめ

## 本日のポイント

- **プロンプトでテスト生成:** 自然言語からテストを自動作成
- **ローカル実行:** 開発中の即時フィードバック
- **クラウド統合:** シームレスなCI/CD連携
- **MCP活用:** IDE内でテスト作成から実行まで完結
- **並走開発:** 実装とテストを同時に進める新しいワークフロー

次世代の「Shift Left」で開発者体験を向上



# 参考リソース

## ドキュメント・サポート

リソース	URL
mabl ドキュメント	<a href="https://help.mabl.com/">https://help.mabl.com/</a>
mabl CLI	<a href="https://help.mabl.com/docs/mabl-cli">https://help.mabl.com/docs/mabl-cli</a>
MCP Server	<a href="npmjs.com/@anthropics/mcp-server-mabl">npmjs.com/@anthropics/mcp-server-mabl</a>
経費管理デモアプリ	<a href="github.com/mfunaki/my-mabl-demo-app">github.com/mfunaki/my-mabl-demo-app</a>

サポート: [support@mabl.com](mailto:support@mabl.com)

ご清聴ありがとうございました！