ローコード開発基盤

「OutSystems」の本当のところ

tdi 情報技術開発株式会社

経済産業省のレポートによる“2025 年の崖”に代表される、日本の企業が直面している IT 課題の解決に加え、DX（デジタルトランスフォーメーション）を加速させるための大きな可能性として注目されているローコード開発。tdi は、ローコード開発基盤「OutSystems」を主軸にローコード開発を推進し、国内で 50 以上のプロジェクトを手掛けています。その中でお客様からよくいただく、OutSystems に関する疑問への回答を「OutSystems の本当のところ」としてまとめました。OutSystems の導入を検討されているご担当者様のご参考となれば幸いです。

内容

1. 業種・業界問わず導入できるが、モバイルゲームは難しい

OutSystems は特定の業務や業種に特化したパッケージ製品ではありません。ローコード開発・保守・運用の統合基盤です。お客様の業務や要望に沿ったアプリケーションを自由に開発でき、業種・業界を問わず導入できます。

業務に特化したパッケージ機能はありませんが、豊富なデザインフレームワークやテンプレートがあり、社内外の利用を問わず、多種多様な業務アプリケーションの開発が可能です。OutSystems の導入事例には、自動車部品メーカーの生産管理システム、美容・ヘルスケア事業のマスタ管理システム、電気・ガス業の資材受発注システム、製造業の作業管理ボードシステム、セキュリティサービス業の防犯・通報システムなどが挙げられ、実に多彩です。

しかし、OutSystems にも不得意なアプリケーションがあります。それは、モバイルゲームに代表されるグラフィカルでインタラクティブな処理が必要なアプリケーションです。これらは専用ツールによる開発が適しています。ただ、ユーザー情報など、バックエンドでアプリケーションを管理する機能については OutSystems による開発が可能です。

2. バッチ処理も作成できる

OutSystems は、画面の UI に関わるロジックの作成同様、視覚的にバッチ処理のロジックを作成できます。また、Java や.NET で作成したバッチと同様の処理スピードを期待できます。

ただし、ジョブ管理ツールのような機能は搭載されていないため、注意が必要です。例えば、複数のバッチをつないでジョブとして実行する、処理の結果を受けて次処理を判断させる、といった作り込みが必要なバッチ処理は、ジョブ管理ツールで作成したほうが効率的です。

また、従来メインフレーム等で稼働していた大量データを夜間バッチとして処理するケースでも注意が必要です。OutSystems では C#でコードが自動生成されます。そのため、COBOL 等の旧言語と比べて、処理に時間がかかります。メインフレーム等から OutSystems にシステム移行し、大量のデータをバッチ実行したい場合は、予め十分に検証・検討することが必要です。

とはいえ、夜間バッチで所定のファイル情報を DB に登録する処理であれば、OutSystems のスケジュール機能を利用して行えます。

3. 帳票出力は OutSystems 公開パーツの利用か帳票製品との連携が必要

OutSystems は、画面作成機能は優れていますが、帳票製品のような機能はありません。作成したい帳票が、明細ごとにページ数を増やすような可変帳票である場合や、改ページ/合計などを出力する場合などは、帳票製品と組み合わせるか、独自に開発をする必要があります。ただし、単一ページの Excel 帳票のような単純なものであれば、OutSystems 社が公開しているパーツを使うことで実現できます。

4. モバイルアプリ作成は得意

OutSystems では、ネイティブアプリケーションのように、利用するデバイスのネイティブ機能（カメラ、GPS、指紋認証など）を使ったアプリケーションを作成できます。ネイティブアプリケーションのような見栄えで利用でき、社内向けのアプリケーションであれば、URL や QR コードで簡単に配布可能です。

特に OutSystems が優れているのは、iOS、Android のアプリケーションを 1 つのモデル情報で作成できる点です。OS ごとに開発環境を構築する必要も、開発要員を確保する必要もありません。さらに、レスポンシブ Web デザインに対応したアプリケーションも開発できるので、画面サイズの違うデバイスも、 1 つのモデル情報で対応可能です。

5. 開発生産性は、製造・単体テスト工程で向上

OutSystems によって工数削減が見込める開発工程は、内部設計と製造・単体テストです。tdi の開発実績では、従来のスクラッチ開発に比べ、内部設計と製造工程は約 5 割、単体テスト工程は約 7 割程度の工数で開発でき、生産性が向上しています。

図 1 OutSystems で開発工数を削減できる工程

OutSystems では、プロセス、インターフェース、ロジック、データの 4 種類の基本モデルを視覚的に定義することでソースコードを自動生成し、アプリケーションを開発できます。そのため、内部設計（＝モデルの定義）が製造工程を兼ね、工数が削減できます。また、OutSystems が提供する機能を利用した場合、コードの動作が保証されるため、単体テストが不要となり、工数が削減できます。

これらの工数削減を支える OutSystems の機能として、スキャフォールディング機能が挙げられます。スキャフォールディング機能とは、ドラッグアンドドロップで一覧画面や詳細画面を簡単に作成できる機能で、設定内容に基づいて自動的にテーブルから値が取得され、画面遷移も保証されるため、テストが不要です。

図 2 スキャフォールディング機能で作成した一覧画面

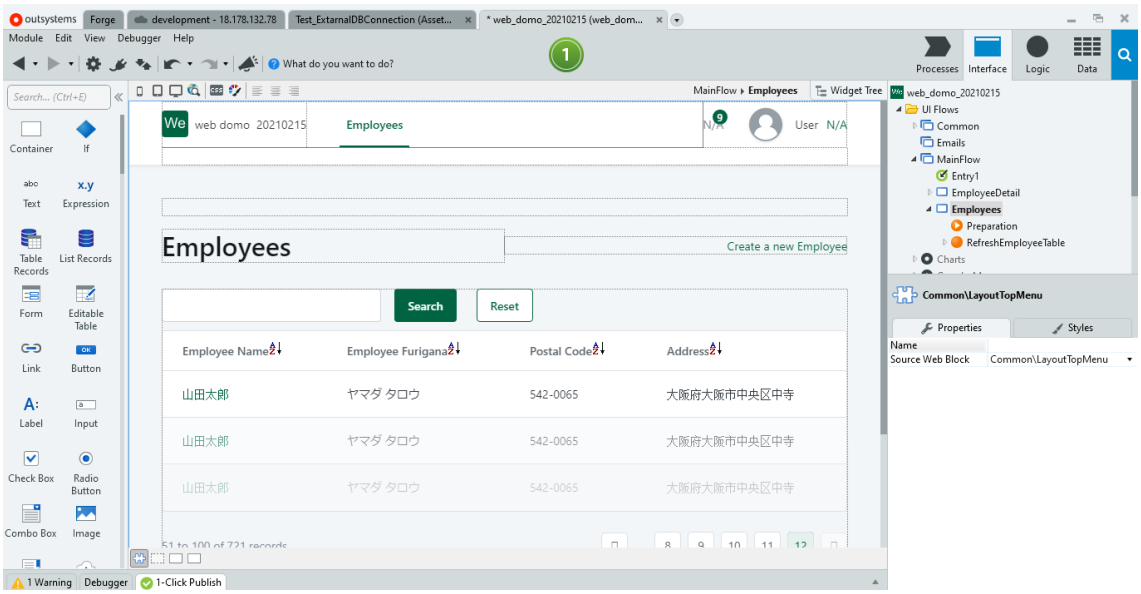
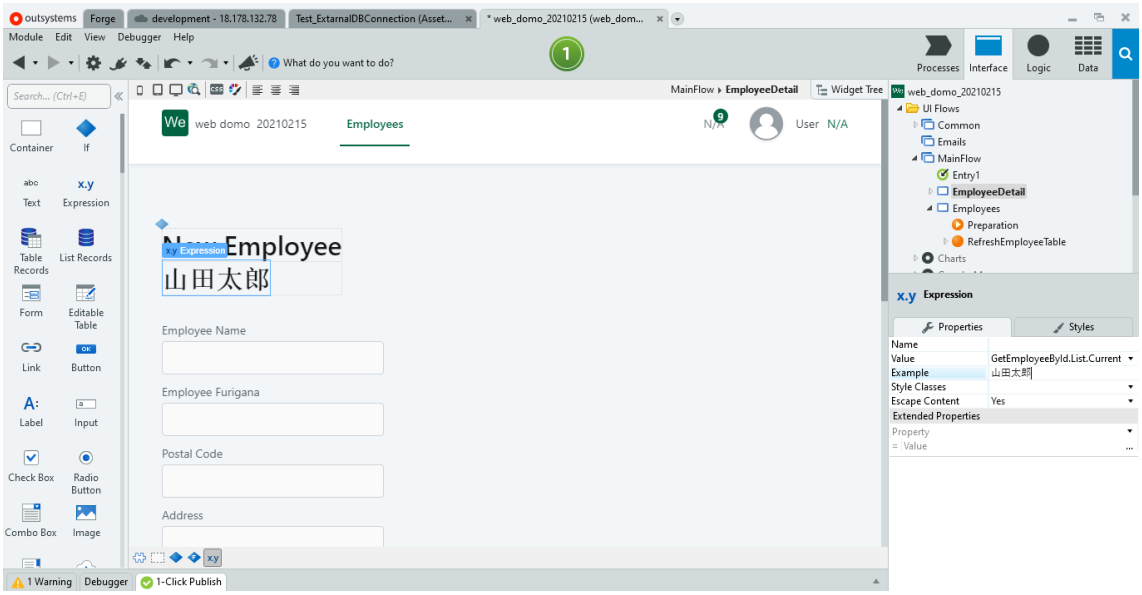


図 3 スキャフォールディング機能で作成した詳細画面



6. 単体テストの実施範囲はロジックを組み込むかで変わる

OutSystems において単体テストが必要になるのは、条件分岐により処理を変更するロジックを組み込んだ場合です。スキャフォールディング機能のように動作保証はされていないため、想定した動作が行われているかをテストする必要があります。例えば、閲覧権限によって詳細画面の表示項目を変える動きをする場合は、ロジックを組み込む必要があるため、単体テストの対象となります。

また、ローコード開発では、データ特性上、テストデータの作成に注意が必要です。OutSystems では、テーブル作成時にデフォルトで主キーが割り当てられます。その主キーを外部キーとして利用している場合は、整合性をとる必要があります。OutSystems でテストデータ作成用のツールを用意し、単体テストを行うこともあります。

7. 自動生成されるコードは動作保証あり

自動生成されるコードの不具合について不安を抱かれるお客様もいらっしゃいますが、心配はありません。自動生成されるコードは、OutSystems によって動作保証されています。tdi が実施した OutSystems プロジェクトは数多ありますが、自動生成されるコードによってアプリケーションの不具合が発生した事例はありません。

不具合に関しては、OutSystems がコードを自動生成する過程よりも、誤ったモデルを作成することによる発生が圧倒的に多く見られます。開発者が作成したモデルからコードが自動生成されるため、モデルが正しければ自動生成されるコードも正しく生成されます。モデルに誤りがあれば、当然生成されるコードにも誤りがあります。

8. アプリケーションのセキュリティ脆弱性対応を実施

セキュリティに関しては、OutSystems 社が「OutSystems を使用して生成したアプリケーションのセキュリティを継続的に改善することに努めています。」とセキュリティに対するコミットメントをしています[1]。また、OWASP（ソフトウェアのセキュリティ環境の現状やセキュアな開発促進に関する情報の共有と普及啓発を目的とした、プロフェッショナルの集まる、オープンソース・ソフトウェアコミュニティ[2]）が掲げる、Web セキュリティとして警戒しなければならない項目 TOP10 に対し、OutSystems ではどのように対応できるかを紹介しています[3]。

図 4 OWASP TOP10 の脆弱性への OutSystems の対応

NO.	カテゴリ	OutSystems の対応
1.	インジェクション攻撃	• コンテンツが UI として表示される前にエスケープさせるデフォルト設定。 • HTML、JavaScript、および SQL のサニタイズ機能あり。 • 開発環境では設計時にコードインジェクション攻撃につながる可能性のあるアプリケーションのパターンを警告。
2.	認証の不備	• URL でセッション識別子を送信しない。 • セッションに有効期限を設け、一定時間経過後にタイムアウトさせる。 • パスワードを処理する際は、必ず強力な暗号化アルゴリズムを使用。
3.	機密データの流出	• セキュアなプラットフォームのインストールをわかりやすく指示。 • 自動生成したコードのエラー処理、例外処理を完全に実施。 • 既存の暗号化 API 連携により、データベースに保存されているデータの安全性を確保。
4.	XML 外部エンティティ	• XML プロセッサライブラリの最新バージョン、SOAP 1.2 の利用をサポート。
5.	アクセス制御の不備	以下のようにアクセス許可が簡単に定義可能。 • 所定の画面へのアクセスに必要なユーザーのロール、または所定のアプリケーションへのアクセス権を持つユーザーの定義。 • ユーザー権限に基づいた、UI 要素の無効化または非表示。 • アクション実行時のユーザー権限の検証。 • ユーザー権限に応じた特定のロジックの実行やデータのサブセットへのアクセス。
6.	セキュリティ設定のミス	• セキュアなプラットフォームのインストールをわかりやすく指示。 • 自動生成したコードのエラー処理、 例外処理を完全に実施。 • 既存の暗号化 API 連携により、データベースに保存されているデータの安全性を確保。
7.	クロスサイトスクリプティング	• インジェクションに関する問題と同様に対処。 • OutSystems のメカニズムにより、アプリケーションが読み込めるソースを限定し、悪意のあるサイトからのスクリプト読み込みを要求する XSS 攻撃を効果的に軽減。 • コンテンツのセキュリティポリシーを使用して、アプリケーションページのフレーム埋め込みを防ぎ、クリックジャッキングを防止。
8.	安全でない逆シリアル化	• 組み込みの JSON 逆シリアル化メカニズムにより、 脆弱性を回避。
9.	既知の脆弱性を持つコンポーネントの使用	• 定期的にスキャンを実行し、スタックのコンポーネント（オペレーティングシステム、アプリケーションサーバー、データベース管理システム、ランタイム環境、ライブラリ、フレームワーク、およびサンプルの OutSystems アプリケーション）をすべて更新。 • ユーザーへの通知を含むパッチ管理プロセスが確立されているため、確実に脆弱性に対処し、通知することが可能。
10.	ロギングと監視の不備	以下により、あらゆるセキュリティの問題を効率的に追跡可能。 • アプリケーション画面へのアクセスの詳細をすべて追跡。 • 外部システムへのアクセスに加えて、プラットフォーム内で実行されているアプリケーションへの Web サービスリクエストをすべてログに記録。

[1] OutSystems 社のセキュリティに対するコミットメント

https://www.outsystems.com/evaluation-guide/building-secure-applications-with-outsystems/

[2] OWASP

https://owasp.org/www-chapter-japan/

[3] OWASP TOP10 脆弱性への対応

https://www.outsystems.com/blog/posts/owasp-10-web-application-security-flaws/
https://www.outsystems.com/evaluation-guide/building-secure-applications-with-outsystems/

9. 開発手法は目的や状況によって変える

ローコード開発製品を利用する場合は、アジャイル開発で行うイメージがあるというご意見をよく伺います。しかし、OutSystems を導入するからにはアジャイル開発を行わなければならない、ということはありません。

tdi の実績では、アジャイル開発によるプロジェクトもありますが、ウォーターフォール開発によるプロジェクトが半数を超えています。理由は、ウォーターフォール開発に慣れている企業の方が多いためです。よって、ウォーターフォール開発を採用した方が、既存の開発体制やプロセスを活かすことができ、導入障壁が下がります。

導入初期はウォーターフォール開発を採用し、段階的にアジャイル開発へシフトしていくケースもあります。その場合に有効な開発手法として、イテレーション開発が挙げられます。これは、ウォーターフォール開発とアジャイル開発の中間に位置し、計画段階でスコープを定義し、その決められたスコープで反復開発を行う手法です。

図 5 開発手法の特徴

開発プロセス	ウォーターフォール開発	アジャイル開発	イテレーション開発
思想	「計画」重視	「価値」重視※アジャイルソフトウェア開発宣言	「計画」重視
スコープ	固定	可変	固定
開発期間	スコープをベースに見積	固定	スコープをベースに見積
コスト	スコープをベースに見積	固定	スコープをベースに見積
開発計画	事前に決定された WBS	毎回スプリントプランニング時に決定	イテレーション単位に計画
リリース	開発終了時	スプリント毎	計画したイテレーション単位
変更	変更管理に従う	受け入れる 価値の高いものから対応	イテレーション単位に変更管理に従う
メリット	大規模開発向き QCD の管理がしやすい	リリースまでが短い 仕様変更が容易	反復なのでリスクを先に潰せる アジャイルよりは管理が容易
デメリット	リリースまでが長い 仕様変更しづらい	スケジュールや規模の管理が難しく、 コントロールの難易度が高い	アジャイルほど柔軟に変更はできない リリースも一定期間必要(WF より短い)

10. 設計書の作成は必要

OutSystems だけでなくローコード開発製品全般に言えることですが、設計書の自動生成や作成範囲に関するお問合せをよくいただきます。ローコード開発では、開発手法により差はありますが、設計書を全く作成しないということはありません。

前提として、開発手法に関わらず、要件定義書、基本設計書は必要です。「何を実現するのか？」を定める要件やシステム全体を俯瞰し、開発の指針とするためです。

一方、実現方法を定める内部設計書は、開発手法によって、範囲が異なります。アジャイル開発の場合、開発のサイクルが短く変更も多いため、設計判断の根拠となる最低限のドキュメントを作成します。例えば、パラメータ等の命名規則や設計情報の作成規則などです。ウォーターフォール開発の場合、各工程で進捗や品質を評価するため、それぞれの工程の成果物として体裁の整った詳細設計書を作成します。

設計書を作成する目的は、設計の意図や思想、根拠などを残すことです。ローコード開発製品を導入すると短い開発サイクルで多くの変更が発生します。都度体裁を整えた設計書を作成しているとそちらに時間が割かれ、開発スピードにも影響するため、設計書の形骸化を招きかねません。そうならないために、形式にはこだわらず、最低限のドキュメントを残していくことが大切です。

終わりに

tdi グループは、 OutSystems の機能や技術について十分な知識を持った多くの技術者を有しており、資格保有者数は国内トップクラスです。ローコード開発が一般的に注目される以前（2016 年）から重ねた OutSystems 開発の実績をもとに、IT 戦略コンサルタントや OutSystems 導入から運用までをトータルサポートします。また、お客様に合わせた人財育成や内製化もご支援いたします。

お困りのご担当者の方は、どうぞお気軽にお問合せください。

【ローコード開発基盤「OutSystems」】

https://www.tdi.co.jp/outsystems/

【お問い合わせ】

https://tdi.smktg.jp/public/application/add/1095

情報技術開発株式会社 **営業本部**

東京:〒163-1332 東京都新宿区西新宿六丁目 5 番 1 号 新宿アイランドタワー32 階
TEL.03-5325-4811(代表) FAX.03-5325-4812

中部:〒451-6027 愛知県名古屋市西区牛島町 6 番 1 号 名古屋ルーセントタワー27 階
TEL.052-571-6871(代表) FAX.052-571-3856

関西:〒530-0005 大阪府大阪市北区中之島二丁目 2 番 7 号 中之島セントラルタワー20 階
TEL.06-6201-7739(代表) FAX.06-6201-7740

九州:〒812-0013 福岡県福岡市博多区博多駅東二丁目 10 番 1 号 福岡ビル S 館 7 階
TEL.092-451-8218(代表) FAX.092-474-7379