ローコードの現状と活用の方向性

株式会社クロスフィールド
庄司 堅太郎

1. はじめに

2010 年代以降、自動ソースコード生成にとどまらずアプリケーション開発サイクル全般を 1 つの統合環境で完結できる、ローコードアプリケーションプラットフォーム（以下、LCAP）が登場し、DX 時代に求められる頻繁な変化や高速開発、世界的な IT 人材不足などに対応できる開発手法として、近年急速な広がりを見せています。米ガートナーは 2024 年までに世界のアプリ開発の 65%以上がローコードで開発されるとみており[1]、日本国内でも大手企業の導入事例を耳にすることも多くなりました。
そこで今回、LCAP としてグローバルマーケットリーダーに位置づけられている[2]ポルトガルのアウトシステムズ社製「OutSystems」（試用版）を使用して筆者が実際にサンプルアプリケーションを複数作成しながら、最新の LCAP の機能や使用感を調査してみました[3]。またそこから得られた LCAP のメリット・デメリットや今後の活用の方向性についても考察いたします。

2. LCAP によるアプリケーション開発の現状

2.1 開発・実行環境

LCAP では開発環境のフロントエンドこそローカル PC にインストールしますが、その全容はクラウドアプリケーションであるため、開発物の格納や実行形式への変換、アプリ実行等全てがサーバー上で行われます。そのため生成できるアプリケーションもブラウザ上で実行する Web アプリケーションかモバイルアプリとなりますが、LCAP ではその開発・実行環境が一元的・統合的に提供されているため、従来の 3 ランドスケープ的な発想やそれに伴う種々の作業（テストやリリースのために別環境へファイルを配置する等）はなく、全てが機能として提供されています。開発機能としてはソースコード管理や様々な自動テストおよびテスト補助機能、各環境へのリリース（デプロイ）機能、構成管理などがあり、実行機能ではアプリケーション動作の監視だけではなくアクセス分析、実行時間分析などの分析機能も多く備えています。また一般的な Web サーバーが有している設定項目も一通り揃っているため、インフラ基盤側の設定と合わせればパフォーマンス改善やスケーリングといったインフラ施策にも対応可能です。

このように、LCAP は開発・実行の全ての工程を統合した環境を提供しており、複数のサーバー・ソフトウェア間における環境差異や設定値差異のような、無用なカベ・落とし穴が存在しません。ここが LCAP の開発スピードが速いと言われる理由のひとつとなっています。

図 1：OutSystems 機能アーキテクチャ
（出典：OutSystems）

---

1 日経コンピュータ 2020.7.23 号

2 Gartner, *"Magic Quadrant for Enterprise Low-Code Application Platforms"*, 30 Sep. 2020

3 他の LCAP 製品についても主要機能を調査の上記述しておりますが、使用感および開発環境イメージは OutSystems のものとなります。また OutSystems 特有と思われる機能についてはその旨記載しております。

なお LCAP 自体の構築環境として、各社が製品とともに提供するクラウド環境の他、MS Azure や AWS といった一般的な PaaS/IaaS 環境やプライベートクラウド環境も選択できるため、自社インフラ環境の都合やセキュリティポリシーなどに合わせて構築できます。

2.2 設計・製造

① 画面・UI

LCAP には一定のビジネスシナリオに基づいた豊富なアプリテンプレートが準備されており、そこから適しているものを選択するだけで、大まかな UI や画面遷移ロジック（一覧画面から詳細画面への遷移やデータの新規作成・更新・削除等）まで含めてアプリケーションの枠組みが出来上がってしまいます。ですが、業務アプリケーションの場合はテンプレートそのままというものは少ないと思われますので、そこを起点として改修していくか、またはテンプレートを選択せずにまっさらな状態から画面をスクラッチしていくかのどちらかになります。その際画面・UI オブジェクトに準備されている多数のプロパティを設定していくことになりますが、生成されるのは HTML 画面ですので、大枠はドラッグ＆ドロップなどでグラフィカルに調整できるものの、細部の調整には HTML や CSS[4]の知識が必要となります。

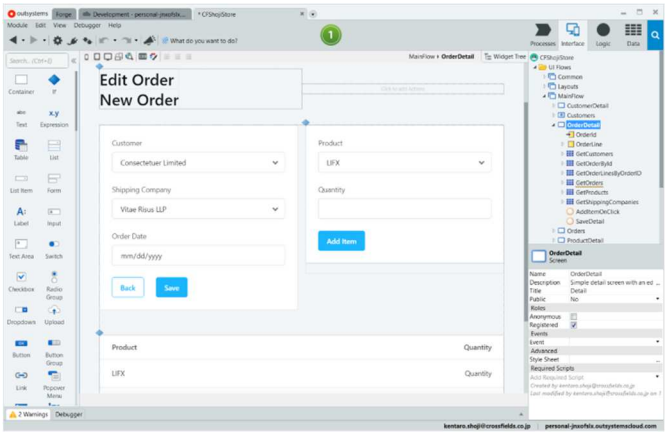

図 2：OutSystems 画面設計環境

② テーブル、エンティティ

データを格納するための各種テーブルを作成する方法として、①テーブル定義を作成する方法と、②既存のデータから作成する方法の 2 つがあります。①は Microsoft Access で作成する手順と大差はなく、項目一つずつ、名称や属性、項目長等を指定して作成しますが、②は既存の Excel ファイルや CSV 等を開発環境にドラッグ＆ドロップするだけで、テーブル定義がコーディングされます。なおこちらも Access のインポート機能と同様のイメージですが、OutSystems の場合、項目名称から属性やテーブル間リレーションの判定を行っており、LCAP の方がだいぶ進化している印象です。

また LCAP でのテーブル作成時の特徴として、テーブルは「エンティティ」として生成され各項目は「エンティティ属性」として認識されると同時に、そのテーブル行の追加・更新・削除などの処理[5]も自動作成されることが挙げられます。これらは画面からイベントドリブンで実行したり、後述するロジックに組み込んだりすることが可能で、LCAP アプリではこれによりデータ操作を行います。なお OutSystems では、エンティティを「インターフェイス」画面にドラッグ＆ドロップするだけで、そのテーブルのメンテナンス画面（一覧形式から詳細画面へ飛んで変更・削除できるような、一連のテーブルメンテ機能）が自動生成されます。このように、ある程度データモデルが明確なアプリケーションの場合は、エンティティ関連の自動生成機能を活かしてデータ構造周りから作成した方が効率的に開発できます。

③ ロジック、外部連携

一般的な Web アプリ同様、ロジックはローカル PC またはモバイル機器側で実行される「クライアントロジック」とサーバー側で実行される「サーバーロジック」の 2 つが実装できます。クライアントロジックは UI（画面）オブジェクトに組み込んで、主に UI イベント

[4] Cascading Style Sheets の略。Web ページのスタイルや見た目を定義しておくことができるコード。
[5] 技術的には、オブジェクト指向言語におけるクラスオブジェクトの Get/Set メソッドのイメージです。

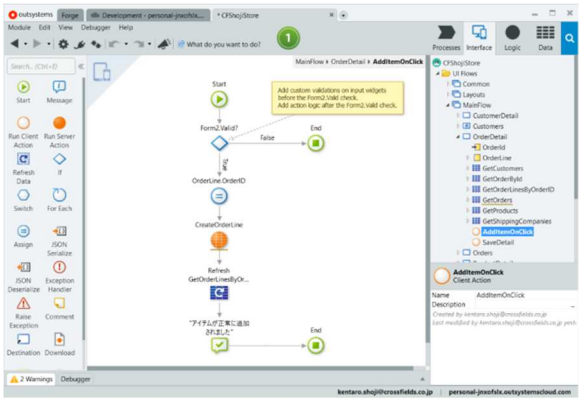

図 3：OutSystems ロジック作成環境

（クリック、スワイプや画面・オブジェクトのリフレッシュ等）を起点として実行されます。一方のサーバーロジックは、主に画面生成時の処理（ログイン・ログアウト、データベース処理、データ集計等）やバッチ処理のようなタイマー設定による定時処理が作成可能です。どちらのロジックも GUI 上でロジックフローをグラフィカルに作成することが可能で、実際には画面作成時やエンティティ定義時に自動生成された基本的なイベントや処理に対して、画面間で受け渡す変数を追加したり前述したデータベース更新処理等をドラッグ＆ドロップで追加したりして組み立てていきます。ただし見た目は非常にわかりやすいのですが、比較的容易な画面遷移や仕様を組み込む際においても、変数の考え方やデータベース処理の基礎知識（更新処理前の参照はロックをかけておくこと(select for update)等）が求められます。

また外部連携ですが、こちらはサーバーロジックに組み込むことが可能で、OutSystems では Web API としての SOAP、REST と SAP 連携部品が標準で準備されています。Web API は Offic365 や Google 連携等を含む多数のテンプレートが用意されていて、これらもドラッグ＆ドロップでロジックに追加していきます。SAP 連携機能も同様に追加できますが、こちらは SAP 標準の API である「BAPI」を直接呼び出すため、ERP のほぼ全領域においてデータや各種伝票の登録・参照が可能となっています。なお LCAP 各製品とも多数のコネクタ部品が準備されており柔軟な外部連携が可能で、既存システムとの連携だけでなく部品化した外部の複雑ロジックとの連携なども比較的容易に実現できます[6]。

3. 今後の活用の方向性に関する考察

3.1 どのようなアプリケーションが向いているか

今回調査してわかったことの一つは、当たり前のように聞こえますが、「単純なアプリケーションであればあるほど LCAP のメリットを活かせる」ということです。筆者の所感としては、ポータルサイトやダッシュボード機能、会議室予約やチケット管理など、Web 画面での入力・共有とサーバーサイドでの簡単な計算やデータベース操作程度であれば、クライアント側での画面遷移やデータハンドリングのロジックの大半を自動生成でまかなえ、コーディング範囲を極小化できるため、文字通り十分に「高速開発」できるでしょう。同様に、販売伝票登録や注文書発行のような既存システム（ERP 含む）のフロントエンドとしても、画面項目や UI の開発や仕様変更は LCAP で迅速に対応し、入出力は API 連携で、といった活用方法が考えられます。

また最近業務系でも活用が進んでいるモバイルアプリに限れば、かなり LCAP を有効利用できるのではないかと感じています。これはモバイル上での画面形態やアクション（クリック・長押し・スワイプ等）が限られており必要なイベントロジックが少なくてすむことと、モバイルアプリには一定の「型」があるためテンプレートを十分活用しやすい、というのが理由です。実際筆者が最も簡単な（といっても小規模のチームでは実用に足るレベルの）課題管理のモバイルサンプルアプリを構築したところ、15 分のテンプレートカスタマイズ後にデプロイボタンを 1 クリックしたのみで、すぐに私物の携帯電話から利用することができました。

[6] OutSystems の統合開発環境は Microsoft Visual Studio と連携できるため、複雑なロジックを Visual Studio で構築して LCAP で部品として利用することが容易に開発できるようになっています。

一方で LCAP での開発に向いていないと考えられるものは、画面の遷移・動線が複雑なものや動的な Web 系アプリです。これらの複雑な動きは当然自動生成ではまなかえず、スクラッチで画面のオブジェクトやプロパティ、イベントを大量に作成する必要があるためです。また同様に、ERP のようにバックエンドでの処理ロジックが複雑かつ多岐に渡るものも、ドラッグ＆ドロップでロジックフローをビジュアル的に作成するよりプログラマがコードを書いた方が早くて保守性も高くなると思われます。

3.2 誰が開発するか

LCAP での設計・製造は 2 章で見ていただいたとおり、「半自動生成」の残り半分を開発する際に必ず HTML の知識やオブジェクトの考え方（プロパティや属性、クラスとメンバ等）が求められるため、教育ゼロで「今すぐ誰でも」できるものではありません。例えば、今回取り組んだ簡易受注管理システムのサンプル開発では、*「OrderDetail オブジェクトの Properties タブで、OrderId の値を[GetOrders.List.Current.Orders.Id]に設定します」*といったレベルのプロパティ設定がいくつかありました。サンプル開発時はこの指示に従って一字一句入力すればよいのですが、これが何を意味しているのかを理解しない限り、全く応用が効きません。似たようなスキルのイメージでいえば、「Excel で簡単な画面付きマクロを作れる」程度のオブジェクト関連知識が最低限必要になりますので、IT リテラシに応じた開発者教育は必須です。

一方で LCAP での開発は既にスキルを持った現在の開発者が行う、という選択肢もありますが、せっかく設計・開発のハードルが下がった LCAP で開発するならば、業務担当者が多少の開発スキルを修得して画面やデータ構造の大枠まで設計・製造し、その後詳細を開発者が詰める、という役割分担が理想的だと思われます。この場合、当初の要件定義や設計だけでなく、アジャイル的な機能追加・仕様変更もかなり効率的に実行できると思われます。また全体の効率が上がればトータルの開発工数も下がることが予想されますので、人数的にも社内で開発者を確保しやすくなり、システム内製化の流れに十分寄与できるのではないかと考えます。

3.3 どの製品を選ぶか

現在市場にはたくさんの LCAP 製品があり、主要製品は機能的に大差ないというもののそれぞれ特徴があるのは事実です。例えば「Salesforce Lightning Platform」は当然ですが Salesforce との連携が容易で、顧客コミュニティ機能や音声対話機能など、CRM 関連の機能が備わっています。また「Microsoft PowerApps」は Excel 等の Office 製品との親和性が高く、ライセンスが Office365 サブスクリプションに含まれるなど、簡易社内システムの構築には敷居が低いかも知れません。よって、LCAP 選定時にはコストや知名度のみならず、特徴のある機能や社内システムとの親和性といった条件も精査する必要があるでしょう。

3.4 さいごに

LCAP は実行環境を含んだサービスであるが故に、ライセンス契約を止めてしまうと開発だけではなく実行もできなくなってしまいます[7]。この「ロックイン」を防ぐためには、LCAP 上で稼働させるアプリケーションと LCAP 自体のライフサイクルを事前に計画しておくことが重要です。また規模に応じたライセンス体系を取る LCAP であれば、それぞれを小さく運用することで複数の LCAP を並行利用でき、そもそも 1 つの製品に依存しない状況を作ることも可能です。

---

[7] OutSystems では、利用を中止する場合でも標準的な C#または Java 環境で実行できるソースコード一式を出力できます。ただしソースコードを出したとしても、大量の自動生成部分を手動でメンテナンスすることはほぼ不可能なため、事実上そのまま動かし続けることしかできないと思われます。
（なお試用版ではこの「Detach 機能」は試すことができなかったため、未検証です）