

2022年、ローコードの一步先へ



デザイン/プロトタイピング/コード生成を簡単に統合するシステム

ローコードへの移行が、ここ数年急速に進んでいます。Research and Markets社の予測によると、2030年にはローコード開発プラットフォーム市場のCAGR(年平均成長率)は31.3%に達するといえます。とはいえ、新しいツールを採用するのは、どんなものでもなかなかハードルが高いもの。慣れ親しんだデザインツールやGitHub、お気に入りのIDE(統合開発環境)をデザイナーや開発者がなかなか手放せないというもまた事実です。

しかし、もし、今まで使ってきたツールや開発プロセスを変えることなく、その上に軽量なツールを積み重ねるようにしてソフトウェアデリバリーのプロセスを劇的に改善できるとしたらどうでしょう？

既存のツールやシステムにデザインからコードまでの一貫したシステムを上乗せする方法 - これが実現できれば、SketchやFigma、Adobe XDといった慣れ親しんだデザインツールを組み込みながら、Visual Studio CodeやGitHubなど、自分が使いたい開発ツールを使い続けることができるはずです。さらに、AngularやBlazor用のコードが生成でき、最新のWYSIWYG(見たままを得られる)エディターも使えるとすれば、デザイナーと開発者のコラボレーションもスピードアップできるはずです。

今回のレポートでは、この点を踏まえ、以下のポイントを見ていきましょう。

- ・なぜ開発工程でバックログ(未対応案件)が滞留してしまうのか？
- ・なぜローコードツールでは不十分なのか？
- ・どうすればアジャイル開発やデザインスプリントの要素を取り入れてデザインからコード生成までを統合できるのか？
- ・なぜ今、Design-To-Codeツールなのか？

1. 開発工程でバックログが滞留してしまう

プログラムの開発部門が長年苦慮してきた問題、それは、この工程でバックログ(未対応案件)が滞留してしまうという問題でした。一方で、今後(2022年以降)は、ローコードやノーコードのツールを採用する組織が増えるとともに、いわゆる「シチズンデベロッパー(市民開発者)」の数も増加すると予想されます。『[App Developer Magazine](#)』の最近の記事でも「普通のシチズンデベロッパーが用いているツールが、アプリ開発のさまざまな側面を担うようになるだろう」と指摘されています。もちろんエンタープライズ・アプリケーションの全てがそうなのと言っているわけではありません。ただ、複雑なアーキテクチャを必要としないアプリや、SLA(合意書)をしっかりと交わすようなミッションクリティカルなアプリ以外については、そうなる可能性が高いということです。記事は、次のように続いています。

“デザイン段階からコーディング段階に移る際に時間とコストをセーブするための最大のポイント

- それは、「コード化」ができて、そこからのカスタマイズもできる精巧なUIパターンのセット、すなわちUIコントロールを用いることだ。“

これにまさに当てはまるアイテムが、[App Builder™](#)などのDesign-To-Codeツールです。この種のツールを用いれば、UIコントロールと共通のUIパターンにマッピングされた定義済みのデザインシステムによって、大幅な時間の短縮が図れるのです。デザインとコードが一体であれば、デジタル製品開発時の2つの大きな課題に一度に対処できるわけです。

2. ローコードツールでは不十分

技術チームのリーダーたちには、長年にわたるある傾向があります。ブラックボックス化した過去のコード生成アプリケーションに固執することが多いのです。一方で、90年代や2000年代に出てきたORM(オブジェクト関係マッピング)タイプのツールは極めて「有言不実行」なケースが多く、本番環境ですぐに使いこすれ、編集や再利用がきかず、自動テスト製品にも全く対応していないのが実情です。最近のツールは、本番環境ですぐに使えるのに加え、テスト可能なコードを出力することでアプリのデリバリーを大幅に短縮できるようになりましたが、旧来のコードはそういった新しいツールの足かせになってしまっているのです。



最近出た『[Information Week](#)』の記事に当社の見解として掲載された通り、進化を遂げてきたローコードやシチズンデベロッパーには、まだまだ成長の余地があります。実際、ローコードやノーコードのプラットフォームには進化が必要で、特にローコードに関しては、消費者向けだけでなく社内向けのアプリケーションを提供していく上でも、UXやデザインの専門家からのインプット(ならびにUXプロセス)が欠かせません。本番環境ですぐに使えるテスト可能なコードでも、ユーザーの期待に応えられるようなUXを提供できなければ、全てが失敗に終わってしまいます。

デザインチームが普段から行っているイテレーション(反復型)プロセスの一部にユーザーやステークホルダーを含めることができれば、開発チームが使用可能なコードから得た柔軟性も相まって、これらのツールに対する認識も変わってくるでしょう。典型的なアプリ開発サイクルにおいて数週間～数カ月の時間節約がどれだけの価値のあることか——そういったことを技術リーダーが理解できれば、Design-To-Codeツールやローコードツールの採用はさほど難しい判断ではないはずです。

当社のIndigo.Designは、UIプロトタイピング、デザインシステム、ユーザーテスト、アプリ構築、コード生成を統合したシステムです。Indigo.Designを使えば、Sketchのような絶対的なコンポーネントを持つアートボード群でさえ、ヒューリスティック(AIによる経験則)を適用することで、適切なウェブレイアウトに変換されます。このような複雑な問題をうまく解決してくれるツールであれば、信頼性やユーザーの依存度も高まるはず。ガートナー社やフォレストラー社による先の調査結果を見ても、ローコードは今後5年間でソフトウェアを構築する上での重要な選択肢になると考えられます。より良い体験をより速くステークホルダーに提供すべく、UXデザインやユーザーテスト、コード出力の観点からローコードに注目が集まるようになれば、技術リーダーとしてもローコードへの動きに抗い難くなるでしょう。

3. デザインからコード生成までを統合する方法

どんなツールでも、新しいものを採用するのはなかなかハードルが高いものです。しかし、もし今まで使ってきたツールや開発プロセスを変えることなく、その上に軽量なツールを積み重ねるようにしてソフトウェアデリバリーのプロセスを劇的に改善できるとしたらどうでしょう？当社デベロッパーツール部門シニアバイスプレジデント、ジェイソン・ペレスが『Code Magazine』に寄稿した記事「[Eliminating Waste During Designer to Developer Handoff](#)」(デザイナーから開発者への引き継ぎ時の無駄をなくす)では、イテレーション(反復型)デザインスプリントの概念と、これをプロセスに追加する方法、デザインスプリントのコラボレーションに有効な[Indigo.Design](#)などのツールについて、さらに、本番アプリで最終的に使用できるコードをデザインから取得する方法などが紹介されています。

デザインスプリントとは、新製品や新サービス、新機能を市場に投入する際のリスク軽減を目的としたアプローチで、限られた時間内でデザイン思考を実践していく5段階のプロセスです。当社では、この考えを一步進め、このプロセスをアジャイル開発のスプリントと同様のものとして捉えています。すなわち、開発プロセスを先取りする形で、問題への対応や仮定の検証、プロトタイプ作成、フィーチャー(機能)やエピックのユーザーテストを実施していくのです(これを2～3週間、あるいはそれ以上のスプリントで行います)。「アジャイル」という言葉を、アプリケーション開発部門だけでなくデジタル製品開発全体に当てはめるわけです。[Indigo.Design](#)では、このような特性をSketch、Figma、Adobe XDなどの既存のデザインツールに加える形で、ユーザーテスト、コラボレーション、開発者のWYSIWYGなどを追加することができます。開発者の方も、Visual Studio CodeやGitHubなど、お気に入りのツールを今まで通り使い続けることができます。

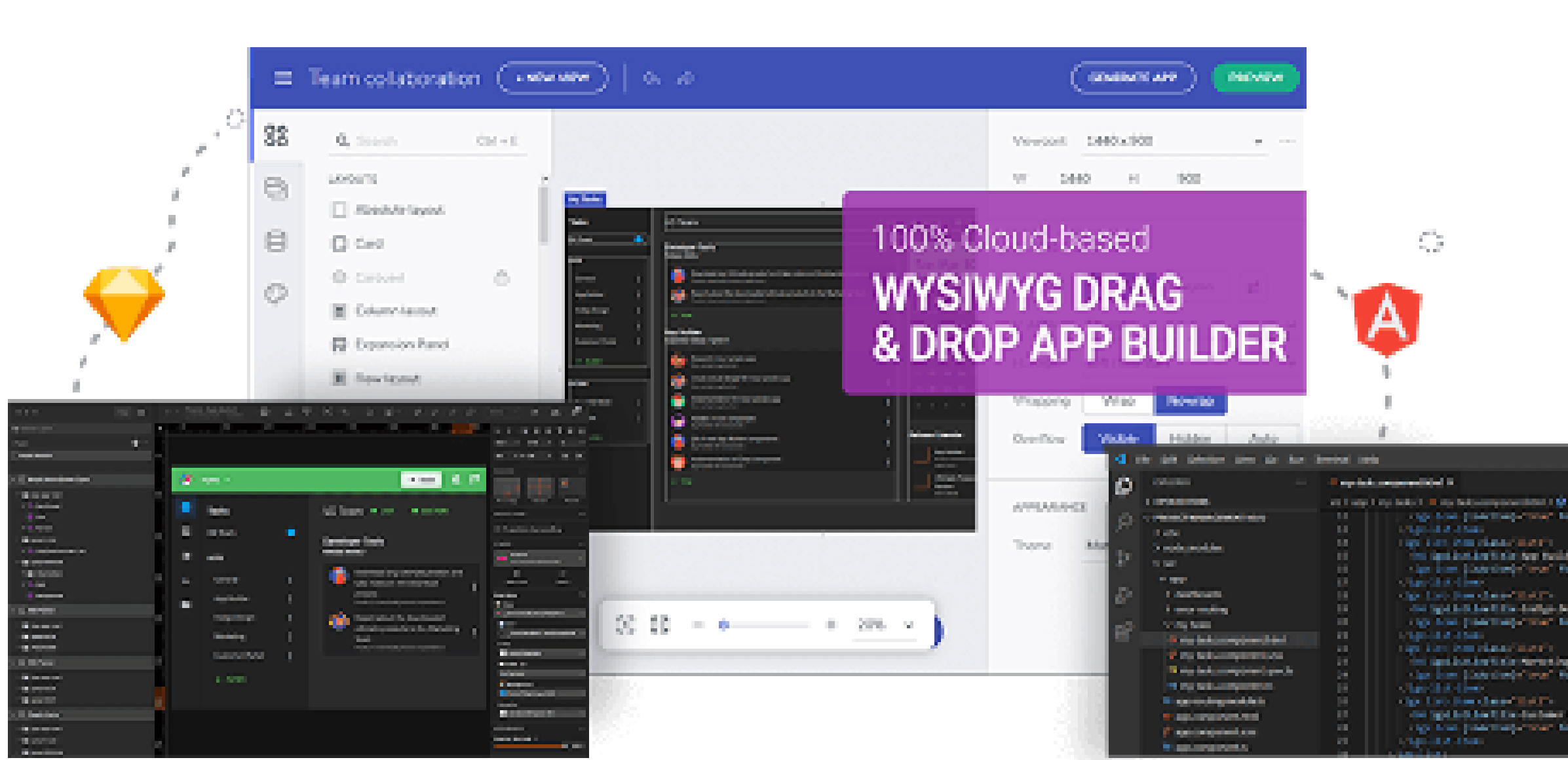


実際のデザイン制作は、ステークホルダーからのフィードバックやユーザーテストに基づく変更を加えながら、イテレーション(反復作業)で行うものです。アジャイル開発においては、ユーザーテストを伴うデザインイテレーションを、開発プロセスより1スプリント(サイクル)先に進めるのがポイントです。まず、1つのスプリントで最初のフィーチャーセットをデザインしてステークホルダーからのフィードバックを受け、あまり時間をかけずにユーザーテストを行った上で、再びデザインに戻ります。ユーザービリティテストは、低忠実度のローファイ・プロトタイプで手間をかけずに、ユーザーテストを経た最終デザインをベースに、開発チームが次のスプリントで開発を行います。このようにデザイン用と開発用の2つのスプリントを回すことで、標準化されたツールやしっかりとしたデザインも併い、優れた成果物を生み出すことができるのです。

4. 今こそDesign-To-Codeツールの出番

アプリケーションはひとりで作るものではありません。さまざまなステークホルダーのほか、顧客も取り込んで実施するデザインイテレーション、効果的なユーザーテスト、シームレスなコラボレーションなどがあって、初めて優れた製品が生まれるのです。しかし、『Proto.io』のブログ記事「[Things Developers Need Designers to Know](#)」(開発者がデザイナーに知ってほしいこと)に当社の見解として掲載されている通り、開発者に対する需要の高まり(一般には技術者不足と捉えられています)は、開発者とデザイナーの役割や両者間の力関係を根本から変えています。かつてはインターフェイスの見た目だけを考えていれば良かったデザイナーですが、最近ではフロントエンドのUI/UX全体を定義する役割を期待されることも少なくありません。また、大きな負荷がかかる開発者への直接の橋渡し役となるため、プロジェクトを開発チームに渡す前に余計なコーディングや不明瞭な箇所を排除する役割を果たす重要な存在でもあります。

今、技術リーダーにとって重要なのは、プロセスの改善だけでなく、コラボレーションやコード出力を改善できるツールに着目することではないでしょうか。開発者の新規採用は難しく、デザイナーはウェブ開発やウェブレイアウトの専門家ではありません。開発ツールを使えば、このような複雑な問題が解消され、今まで数週間から数カ月かかっていた作業を数分から数時間で済ませることができます。



[Indigo.Design](#)や[App Builder™](#)があれば、デザインからコード生成までのアプリケーション開発のスピードアップが図れます。繰り返しが多く時間のかかるルーティンワークを減らし、イノベーションや市場投入までの時間を短縮できるのです。[Indigo.Design](#)は、UIプロトタイピング、デザイン機能、ユーザーテスト、アプリケーション構築、コード生成を統合したデジタル製品デザインプラットフォームです。デザインの引継ぎ作業を省き、コストのかかるイテレーションを減らすことで、UXのためのデザイナーと開発者の真のコラボレーションを実現させます。

開発者は、クラウドベースのWYSIWYG [App Builder™](#)を使うことで、SketchやAdobe XDファイル、あるいはあらかじめ用意されているテンプレートを実際のUIコンポーネントにすばやくマッピングし、デザインを実際のコードとして瞬時にプレビューすることができます。さらに、本番環境で使用できるAngularやBlazor用のコードを生成してGitHubにアップロードしたり、お気に入りのIDEにダウンロードしたりすることもできるようになります。

インフラジスティックスについて

インフラジスティックスは、UX/UI開発ツールキットやコラボレーションツールを30年以上にわたって全世界で提供しているリーディングカンパニーです。全世界で200万人以上の業務アプリケーション開発者が利用しているUX/UI開発ツールキット「[Infragistics Ultimate](#)」、[Ignite UI](#)」、「[Indigo.Design](#)」、「[App Builder™](#)」のほか、組込分析ソリューション「[Reveal](#)」やコラボレーションツール「[Slingshot](#)」等のビジネスツールを提供し、業務ソフトウェア開発における生産性向上や優れたUXデザインの実現を支援しています。

当記事及びインフラジスティックス製品に関するお問い合わせ

JapanSalesGroup@infragistics.com

電話: 050 (1745) 6258

インフラジスティックスの最新情報はこちらから

 [/IGJP](#)

 [/company/igjp](#)

 [/IGJapan](#)