

【セッションNo. 1】

モバイルアプリ開発を成功させるポイント ～ これをやらないと失敗する ～

エンバカデロ・テクノロジーズ
シニア・セールスコンサルタント
伊賀 敏樹 様

■ エンバカデロ・テクノロジーズについて

ポーランドの開発ツール部門を引き継ぎ、ビジュアル開発ツールをマルチデバイス対応の新しい次元へと進化

ビジュアル開発の生産性

コンポーネントのドラッグ&ドロップによる
効率的な開発

真のネイティブ開発

中間コードや仮想マシンを必要としない
真のネイティブコードを生成。デバイス
機能を100%発揮



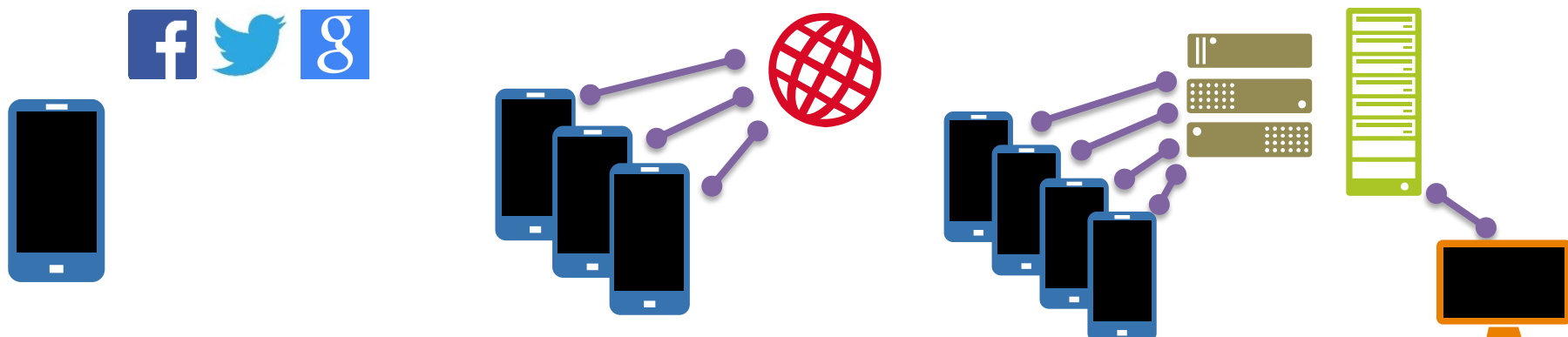
強力なデータアクセス機能

ビジネスアプリで必須となる広範な
データベースに効率的にアクセス可能

そして、マルチデバイスサポート…

Windows、Mac、iOS、Androidアプリを
単一のコードベースから構築可能

企業におけるスマートフォン活用の変化



第1段階

スマートフォンの企業導入

- ・ 社員にスマートフォンを支給
- ・ BYODを含むデバイスの管理

モバイルな情報端末の獲得

第2段階

Webベースのアプリを活用

- ・ 従来のWebシステムにアクセス
- ・ モバイル向けに最適化

業務にアクセスできる範囲の拡大

第3段階

ネイティブアプリを構築

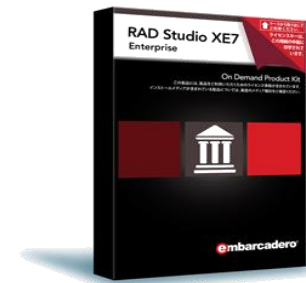
- ・ 業務を変える専用アプリを構築
- ・ 企業のバックエンドに接続

業務そのものの改革

そんな中エンバカデロは…

- ビジュアル開発ツール **Delphi XE5** を 2013年よりスマートフォン・タブレット開発に対応

- Windows、Mac、iOS、Android向けネイティブ開発を実現
- 従来のドラッグ & ドロップ開発でモバイル開発も可能に



- Delphiを用いて多くの企業がモバイル開発の先陣を

- エンバカデロはその成功だけでなく、いくつもの「**苦勞話**」を見てきました



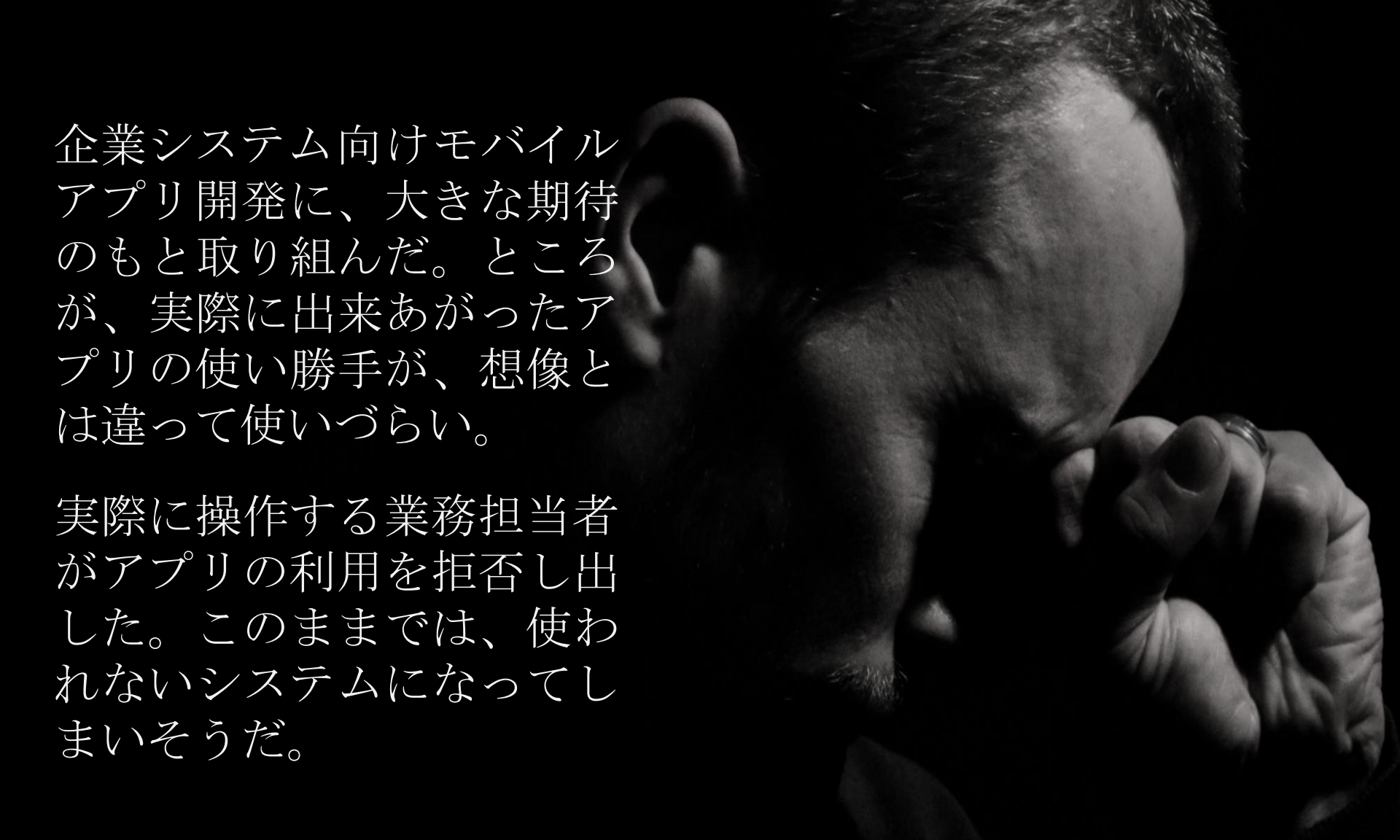
そこで本日は…

先陣が陥ったモバイルアプリ開発の失敗例を
「アンチパターン」
として分析

- モバイルアプリ開発を成功させるためのポイントを解説します。

© Matthew Wilkinson

アンチパターン 1



企業システム向けモバイル
アプリ開発に、大きな期待
のもと取り組んだ。ところが、
実際に出来あがったア
プリの使い勝手が、想像と
は違って使いづらい。

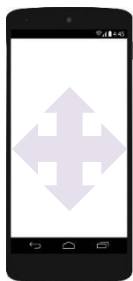
実際に操作する業務担当者
がアプリの利用を拒否し出
した。このままでは、使わ
れないシステムになってし
まいそうだ。

なぜこんな事態に至ったのか…

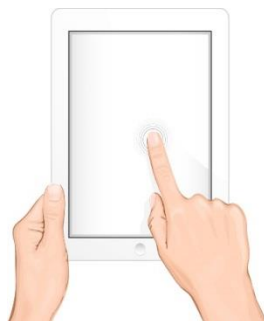
- モバイルアプリのUXは異なる

- Windows向けアプリケーションでは、UX (※)は**暗黙知**であることが多い
- その延長でモバイルアプリのUXを設計すると業務に合わないものを作ってしまう恐れが…

※UX(ユーザーエクスペリエンス) : システムユーザーの体験
(操作性、わかりやすさ、使用環境など)を総合的に表すもの



限られた画面サイズ



© freedesignfile.com

タッチ中心の操作



© Vector Open Stock

ながら業務



© webdesignhot.com

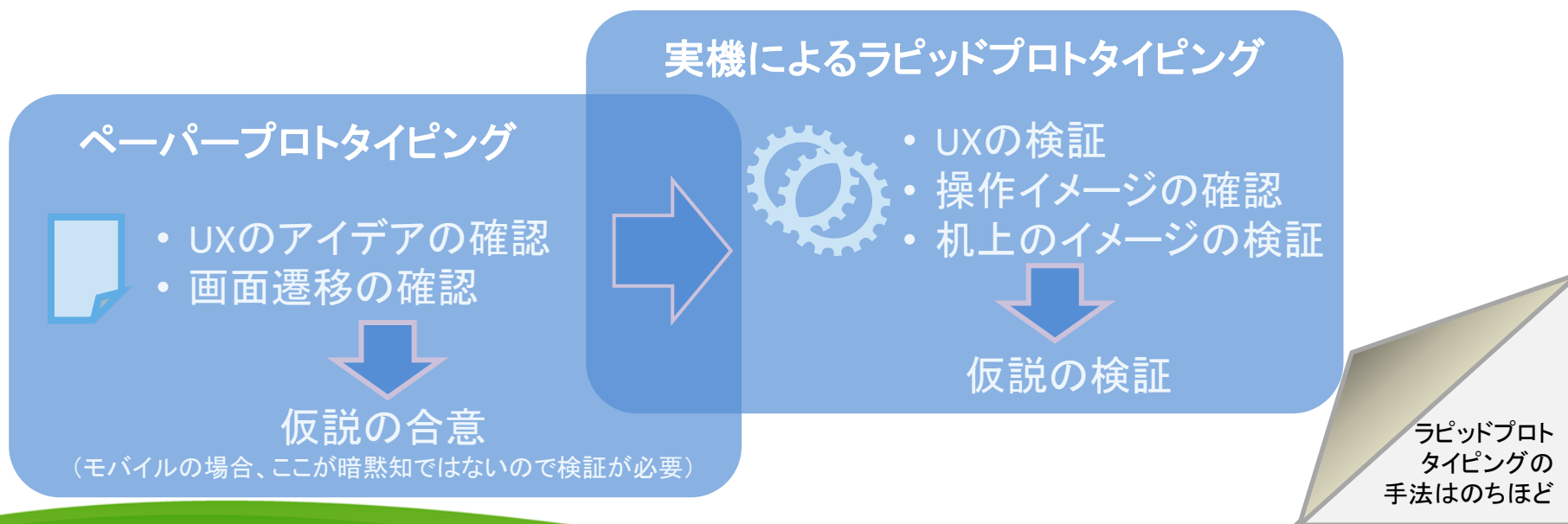
すぐれた外観

1 アンチパターン

UXの違いを理解した検証を怠る

モバイルアプリ開発を成功させるポイント

- モバイルのUXを検証するためのプロトタイピングを実施する
 - 単なるモックアップよりも、動くアプリのほうが効果的
 - 短期間で効率的にプロトタイピングを行うには、RADツールを活用した**実機デバイス上での**ラピッドプロトタイピングの手法が有効



アンチパターン 2

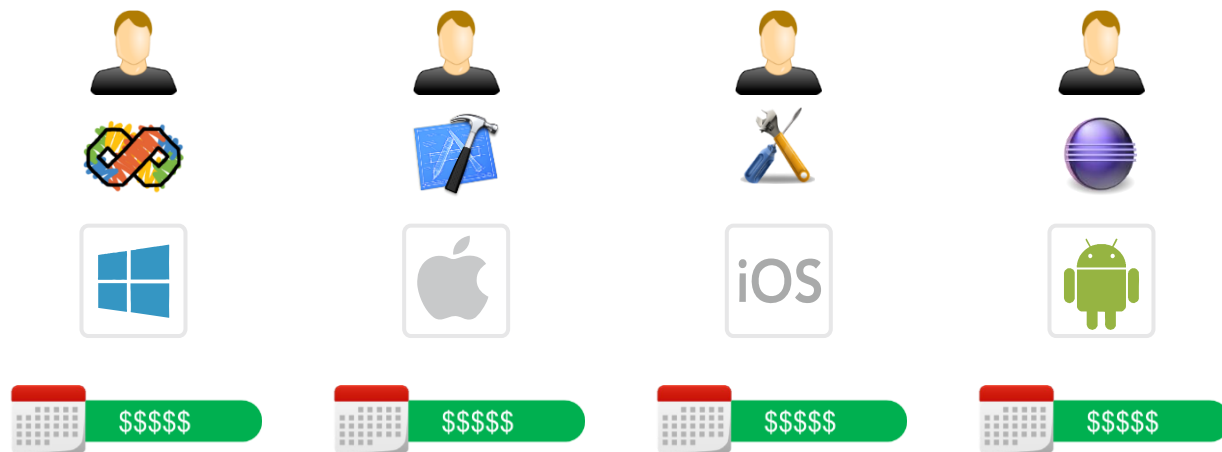
開発した iOS アプリが
ヒット。Androidにも対
応することになった。

しかし、予想以上にコス
トがかかり、今後のメン
テナンスも大変そう...



どのような事態に陥るのか？

- iOSからAndroidへ移植のためのコスト
- 2種類のソースコードを継続的にメンテナンスしていくコスト
- さらに、新しい画面サイズ/解像度、新バージョンOSへの対応コスト
- 将来新しいモバイルデバイスプラットフォームが台頭してきたら？



コーディング費用例

50,000 行のコード

140 万ドルのプロジェクト費用

184 人月の工数

コード1行当たり費用 28.31ドル

2 アンチパターン

マルチデバイスを想定しない 開発計画

モバイルアプリ開発を成功させるポイント

- マルチデバイスを考慮した**ワンコードベース**の実現
 - ひとつのコードでマルチデバイスに対応する
- しかし同時にプラットフォームごとの差異を扱えなければならない
 - OSごとにUXは違う
 - 複数画面サイズへの対応も思わぬ落とし穴

プラットフォーム独立性



- 共通のUI
- 共有の画面遷移
- ロジック

vs.

プラットフォーム独自性



- OSごとに異なるUX
- OS固有の機能や制限
- 画面サイズ

これらを包含するフレームワークの活用が鍵

マルチデバイス
対応の具体的な
方法はのちほど

3 アンチパターン



当初は既存システムのモバイルへの移植にすぎなかった。しかし、せっかくモバイルなんだからと、カメラやGPSなどの機能を使いたいという要求があがってきた。

しかし、選択した開発環境の制約で、デバイス機能をフルで使うことができない。

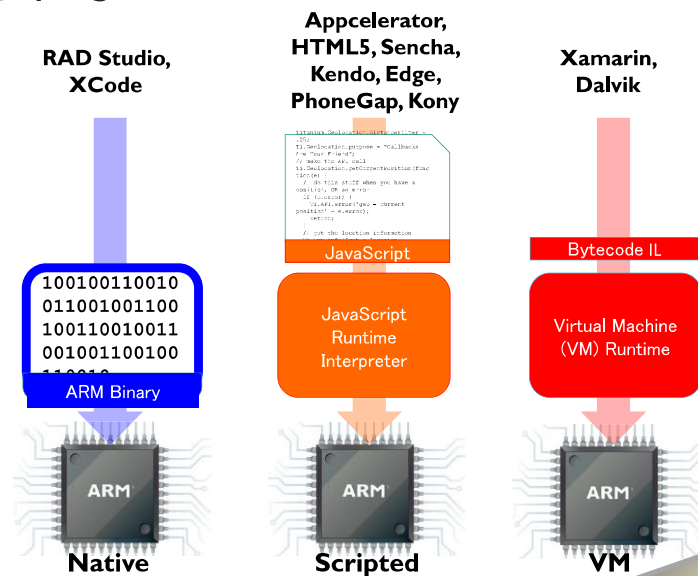
© j0sh (www.pixael.com)

3 アンチパターン

モバイルの特徴を考慮しない
開発環境選定

モバイルアプリ開発を成功させるポイント

- 第3段階のモバイル活用は「業務そのものの改革」
 - モバイルデバイスの機能を活用して、仕事のありかたを変える
 - 従来のWebアプリの延長では成し遂げられない
- そのためのツールは**モバイルデバイスの機能を100%活かせること**
 - ツールがこれらの機能をサポートしていることはもちろん、OSレベルのAPIへのアクセスも保障していることが重要



デバイス機能を
活かすツールに
ついてはのちほど

アンチパターン 4



開発も中盤に差し掛かったころ、データ量の増加に伴って著しい性能劣化が発生した。通信速度が不十分だと、もはや使いものにならない。

© Zuerichs Strassen

なぜこんな事態に至ったのか…

- デスクトップと同様のアーキテクチャを**熟慮することなく**使っていないか?
 - 通信速度は十分ではないかもしれない
 - 通信は切断されるかもしれない
 - オフラインかもしれない
 - CPUを使いすぎるとバッテリーを消費してしまう
- これらのことを考えずにアプリを開発すれば、うまくいくはずがない

従来のデスクトップアプリの方式



これをそのままモバイルに適用しようとする...

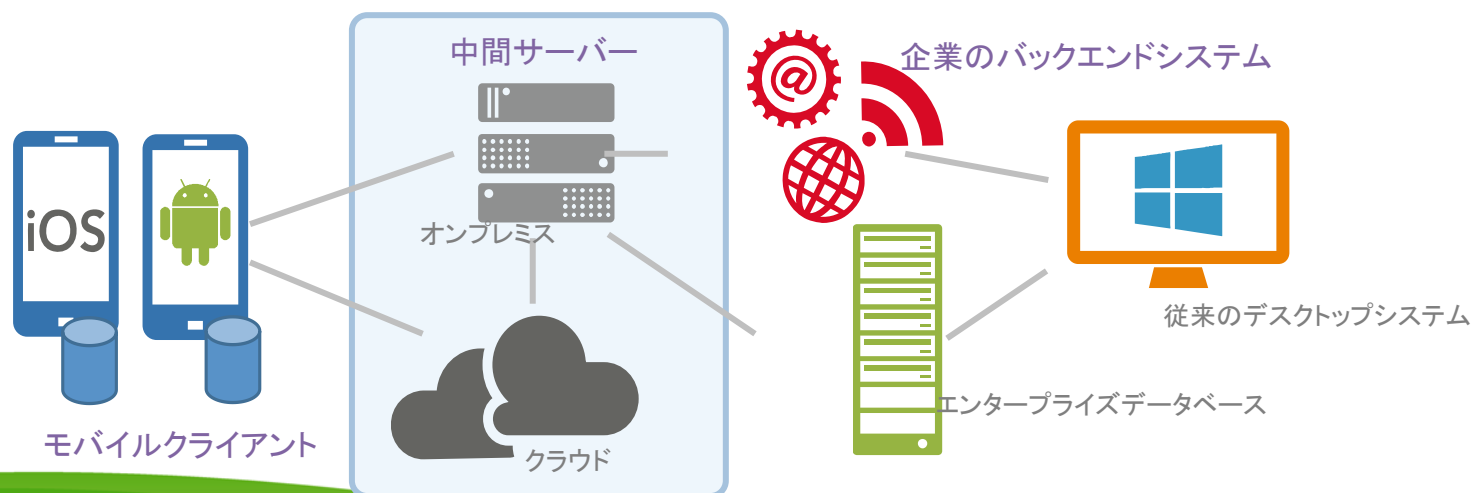


4 アンチパターン

モバイルアプリで
接続方式や通信量を考慮しない

モバイルアプリ開発を成功させるポイント


- モバイルアプリの定番通信方式を前提に**リアーキテクチャ**する
 - 大原則として、モバイルアプリは通信しすぎてはならない
 - **中間サーバー**を配置してバックエンドシステムとの連携を行う
- 従来の方式がどこまで通用するかではなく、モバイルの定番を出発点に
 - 中間サーバーで何をやらせるか、言い換えれば「モバイルクライアントに何をやらせないか」の大原則を最初に立てる





モバイルアプリ開発 成功の鍵

モバイルアプリ開発成功の鍵

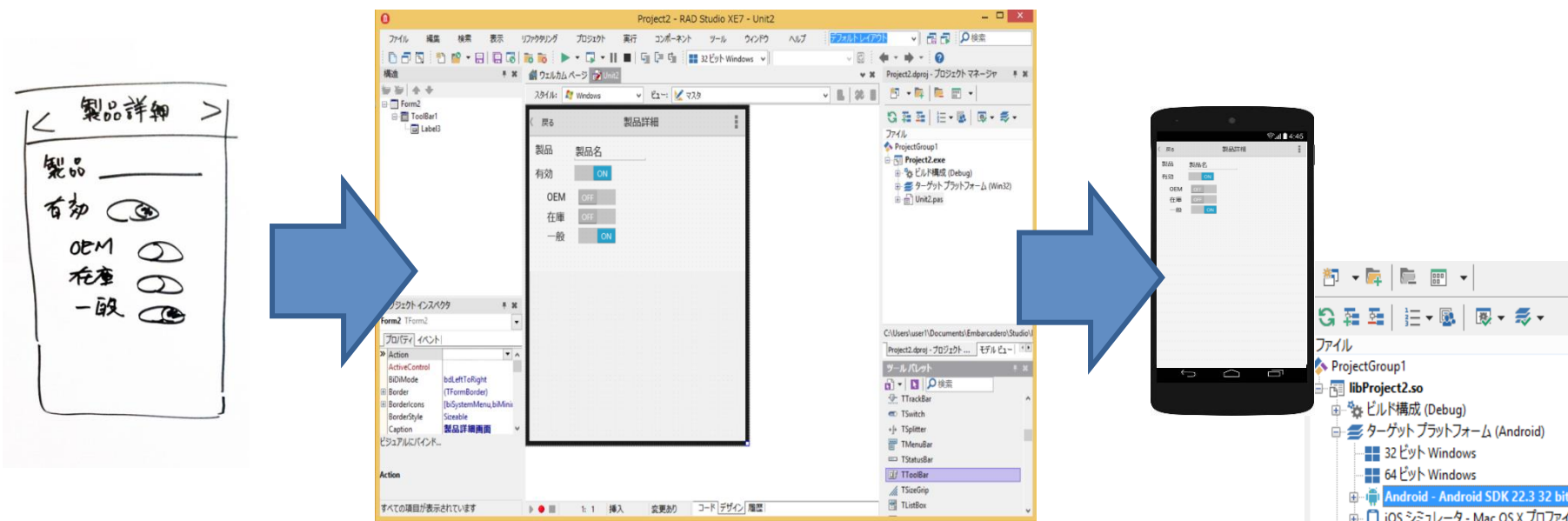


アンチパターン	成功の鍵
UXの違いを理解した検証を怠る	モバイルのUXを検証するためのプロトタイピング • ラピッドプロトタイピング
マルチデバイスを想定しない開発計画	マルチデバイスを考慮したワンコードベースの実現 • ワンコードベースのためのフレームワーク
モバイルの特徴を考慮しない開発環境選定	デバイス機能を100%活かせる開発環境 • マルチデバイスネイティブ開発環境
モバイルアプリで接続方式や通信量を考慮しない	モバイルアプリの定番通信方式を前提にリアーキテクチャ • 中間サーバーの構築

これらを確実に実現するための仕組み=ツールを活用しよう！

活用例 1 : ラピッドプロトタイピング

Delphi の効率的なツール環境により、この工程を数分から数10分で完了させることが可能



ペーパープロトタイピング

はじめにイメージを可視化

ラピッドプロトタイピング

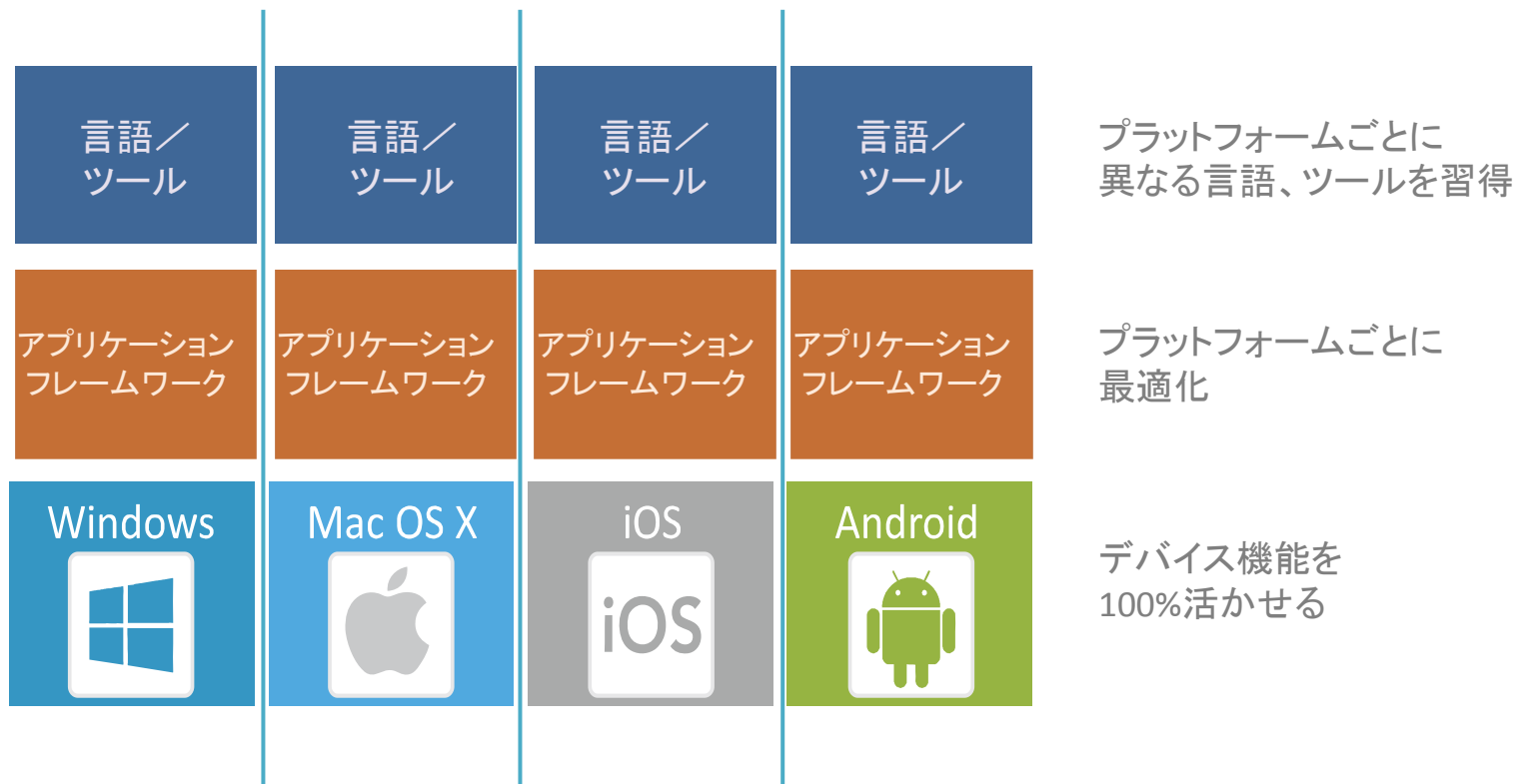
Delphiを用いることで、ビジュアル操作ですばやくプロトタイプを作成

実機で確認

Delphiで各デバイスにすばやく転送

活用例 2 : ワンコードベースのための フレームワーク

従来のシングルデバイスアプローチ



活用例 2 : ワンコードベースのための フレームワーク

最適化されたマルチデバイスアプローチ

ただし、これだけだとプラットフォームごとに異なるUIに対応できず破綻

言語／ツール／コンポーネント

単一の言語、ツールを使用

プラットフォームごとにUIを最適にカスタマイズできる仕組み

マルチデバイスアプリケーションフレームワーク

プラットフォームごとの
差異を抽象化

Windows



Mac OS X



iOS



Android



活用例 2 : ワンコードベースのための フレームワーク

Delphiでは、ワンコードベースを実現する
FireMonkeyフレームワークを搭載

Delphi 言語

FireUIマルチデバイスデザイナー

FireMonkeyフレームワーク

Windows



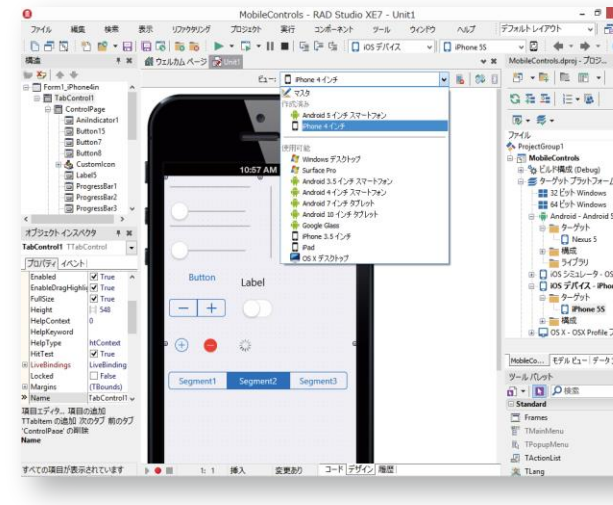
Mac OS X



iOS



Android



活用例 2 : ワンコードベースのための フレームワーク

FireUIマルチデバイスデザイナーを使って複数の画面解像度、
複数のOSプラットフォームのUI設計を効率化



マスター

はじめにマスターUIを定義



Android 5インチスマートフォン

デバイスごとのカスタマイズを作成



iPhone 4インチスマートフォン

- マスターUIを継承
- マスターの修正は継承先にも反映
- 継承先の変更はマスターには影響しない
- ロジック(コード)は共通

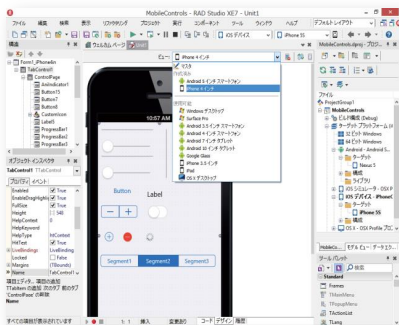


Surface Pro



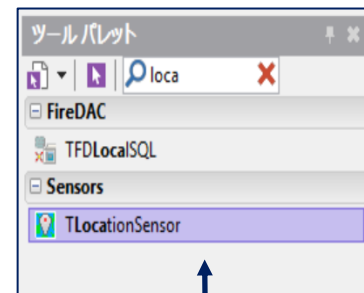
iPhone 5.5インチスマートフォン

新しいデバイスが登場したら
新たにカスタマイズを作成するだけ



活用例 3 : マルチデバイス ネイティブ開発環境

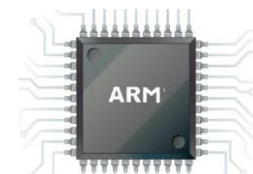
Delphi の場合、各プラットフォーム向けに
最適化されたネイティブコードが生成される



OS固有の機能を内包

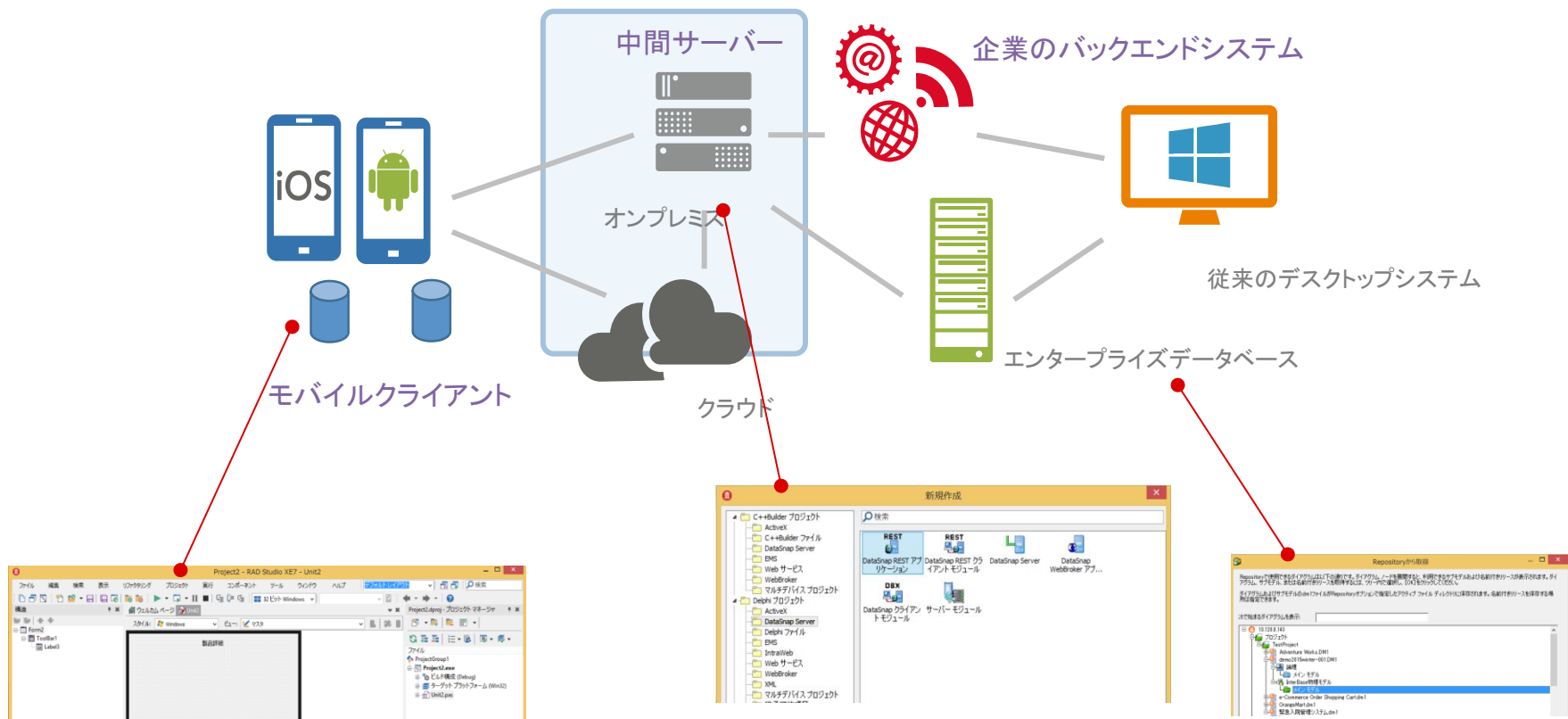
プログラマレベルでは
抽象化されたコンポーネ
ントによってアクセス

生成されるコードはネイティブ
Win32/Win64 Intel、OS X Intel、iOS
ARM、Android ARMコンパイラを搭載



活用例4：中間サーバーの構築

Delphi はモバイルクライアント開発機能だけでなく、
中間サーバーの構築、バックエンドとの連携などの機能が搭載されている



これからも続く Delphiの進化



プラットフォームの「進化」に適応

- Windows 10
- iOS 9
- Android M
- OS X El Capitan...



「接続性」の拡張

- ビッグデータ
- クラウド
- IoT (Bluetooth, WiFi)...



「多様化」への対応

- プラットフォームネイティブコントロールの強化
- AppAnalytics
- Linuxサポート(サーバー側)

IT環境の進化に対して、ビジュアル開発の技術を用いて効率化、単純化を推進

まとめ

- モバイルアプリ開発はもう怖くない
 - 幸いなことに多くの先陣がさまざまな苦勞をしてくださいました
 - アンチパターンを理解してプロジェクトを成功させましょう
- ツールを正しく活用しよう
 - ツールはあくまでも開発者を支援する道具
 - 正しいビジョンを持って開発に取り組めば、ツールは活かされます
- モバイルの導入で業務を改革しよう
 - モバイルの普及によりITがかつてないほど業務にかかわるようになりました
 - モバイル導入が第3段階にあることを理解して、積極的に取り組みましょう



© avaxhome.ws