事例で学ぶ
Delphi / C++Builder 開発手法

第36回 エンバカデロ・デベロッパーキャンプ

エンバカデロ・テクノロジーズ

embarcadero®
DEVELOPER CAMP

皆さんのシステム開発は「ルーチン化」していませんか？

- Delphi / C++Builderの基本開発スタイルは90年代後半に確立していました。多くの開発はその延長線上に位置することができますが、その「定石」感が、ツールの可能性をひとところにとどめてきたという印象はぬぐえません。
- 本日は、この1時間で、「定石」から「革新」へと飛躍する事例を考察しながら、Delphi / C++Builderの開発スタイルの新しい「定番」を構築していきたいと考えています。

最初は定番の事例から…

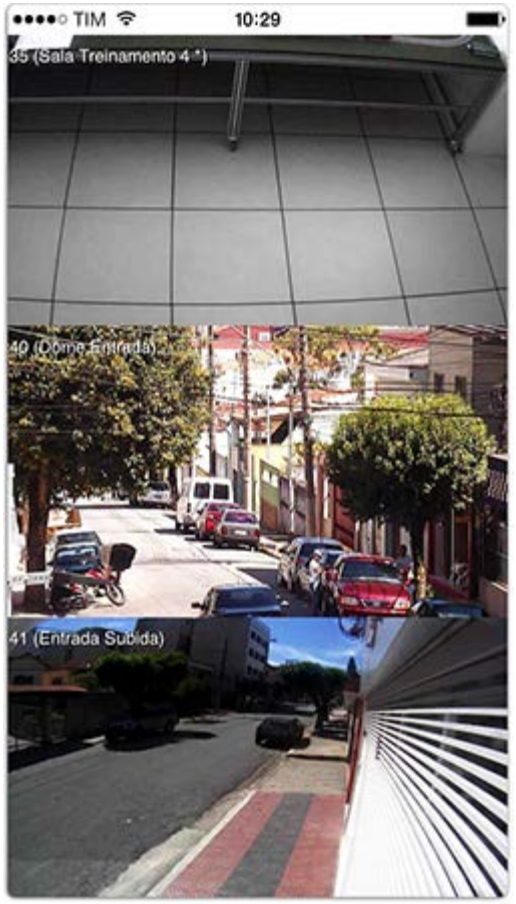
“

C++を使っていたら、今の開発チームの規模では、現在提供している製品の半分も完成できなかっただろう。同じことをするのに数十人の増員が必要になっただろう。我々にとって12年前、Delphiでプロジェクトをスタートできたことは幸運だった。

Éric Fleming Bonilha,
Director of Development,
Digifort

Digifort

- ブラジルに本社を置き、グローバル展開するIP監視ソリューションの大手
  - 北・南米、欧州、アジアの85ヵ国以上に展開
  - TCP／IPベースのビデオ監視システム、生体認証、映像分析ツール、車両ナンバー認識ツールなど
- モバイル化の市場要求に対し、約１ヵ月で「Digifort Mobile Camera」をリリース
  - これまで培ってきたDelphiの開発経験を活用
  - 単一コードで開発できる利点を活かし、コードを再利用
  - ユーザーインターフェイスの構築に、Delphiの効率的なマルチデバイスビジュアルデザイナを利用



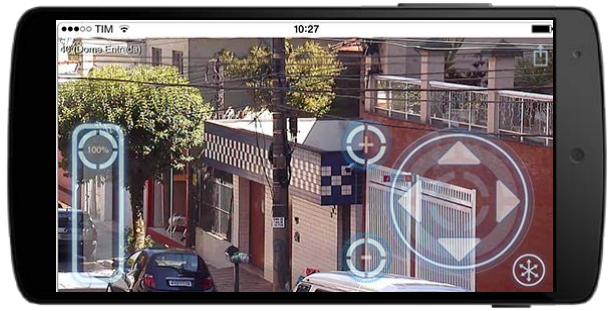
3 cameras on vertical orientation

ビデオ監視システムの位置付けを変える

- 従来は…
  - Windowsベースの監視システム
  - ネットワークを使った集中監視
  - 監視センターでの業務に使用

- 現在では…
  - モバイルベースの監視システム
  - いつでもどこでも監視可能
  - 警備員が現場での状況判断に使用

監視システムが現場に



この大きなイノベーションをもたらすために要した開発期間はわずか1ヵ月

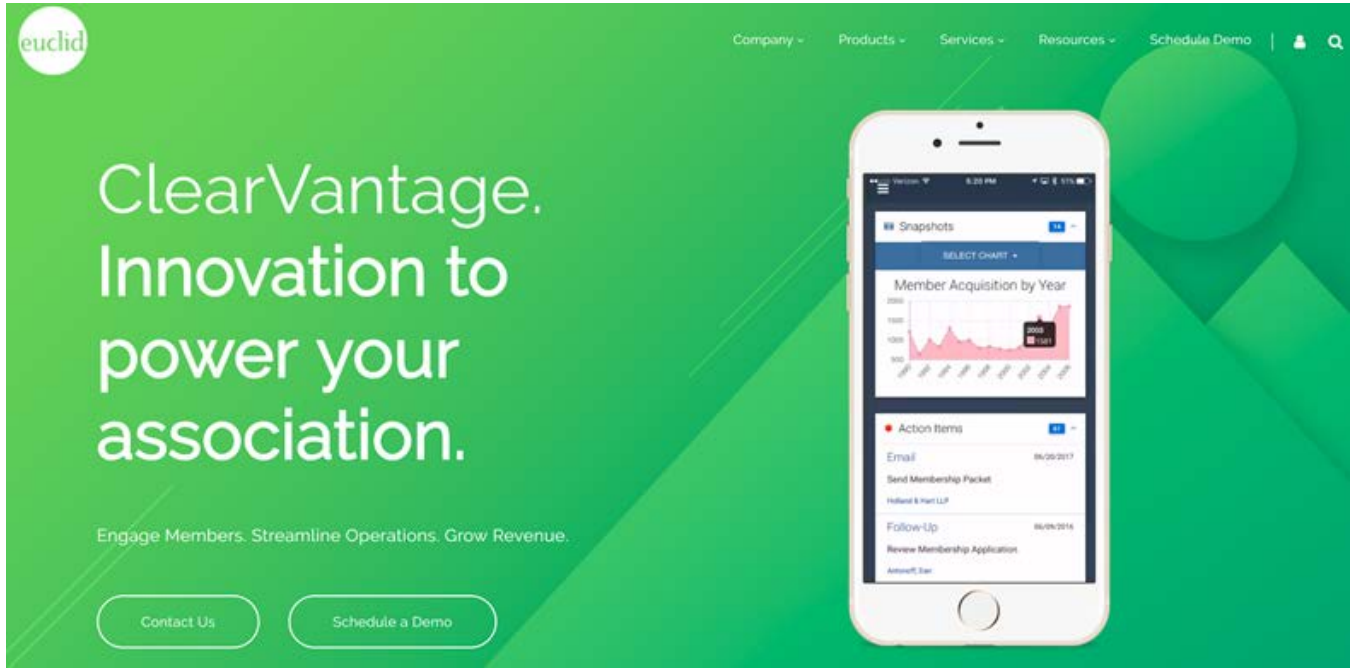
考察

- 既存のノウハウを活かせることは最初のアドバンテージ
  - 例えば、Javaを選択していたら、Delphiでの開発期間が終了しても、Javaによるモバイル開発を習得すらできなかったかも！
  - 開発の「定番」を知っていることで、アイデアを容易にカタチに落とせる
- しかし、それだけでは真の意味でのイノベーションは起きない
  - 既存のWindowsアプリケーションをモバイルに焼き直しても、それは単に動作するOSやCPUが変わっただけ
  - モバイル化するメリットは何か？モバイル化して何が変わるのかを見据えることが大事！

業務とテクノロジーの双方に精通していることが、イノベーションの鍵

続いて2つの先進テクノロジーを融合させた例...

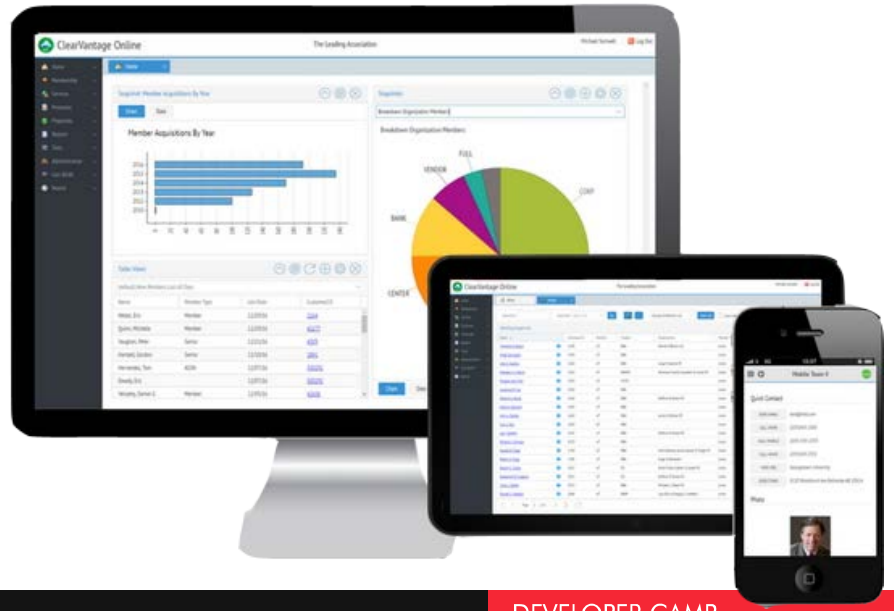
Case Study:



- 既存のDelphiアプリケーションにSenchaによるWebサポートを追加。
  それでもなお選択の自由は担保

Euclid Technology

- Delphiで構築されたCRM、ERP、eコマース、Web CMSを網羅した統合システム「ClearVantage」を提供
- 業界団体および非営利団体ユーザー向けに考案され、職員数15人～1000人規模の100以上の組織に導入

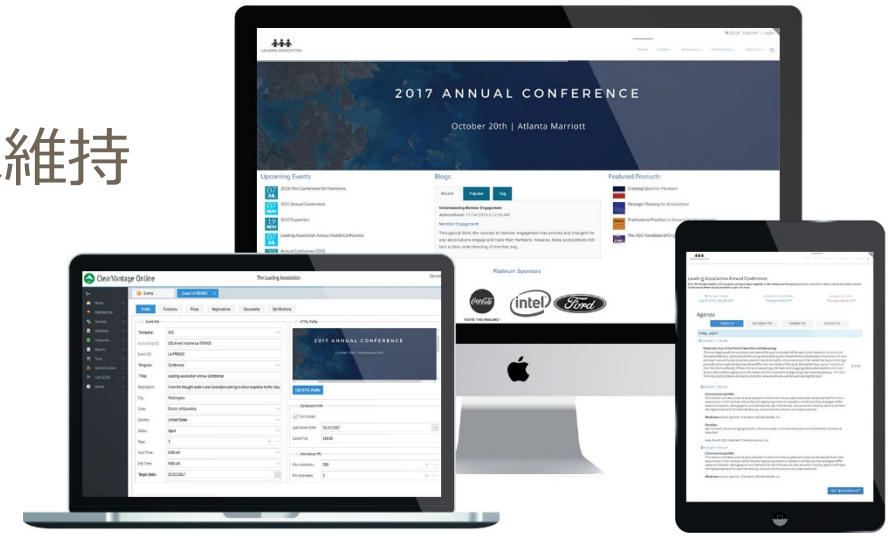


フェーズ１：Delphiによる開発

- 小規模開発チームで高い生産性を実現
  - Delphiにより10倍の規模をもつ開発チームにも対抗できる生産性を獲得
  - 顧客ニーズに合わせたカスタマイズにも俊敏に対応
- 複数のニーズに対応
  - Web、クライアントサーバー、Webサービスなど多様な要求に対応
  - コア機能をDelphiで提供することで、ネイティブの性能を享受
- オープン性を提供
  - 多様なアクセス性を提供するために、
    アプリケーション機能はREST APIとして提供
  - XML、JSON、HTMLでデータを提供
  - ASP、ASP.NET、ColdFusion、PHP、Perlを含む、
    ほとんどすべてのプラットフォームから利用可能に

フェーズ２：Senchaの導入

- SenchaによるWebアプリケーションを構築
  - Ext JSを用いて高性能なWebユーザーインターフェイスを提供
  - 従来C/Sアプリケーションとして提供してきたような機能をWeb経由でアクセス可能に
  - ユーザーの利用形態を拡張
- バックエンドは元来オープン性を提供
  - もともとWebサービスとして提供してきたオープンなインターフェイスを利用し、すばやくSenchaからアクセス
  - Senchaなら、標準で、XML、JSONなどオープンなデータ形式を主要なプロトコルで利用可能
- Senchaを採用しても、引き続きオープン戦略は維持
  - Senchaを導入したからといって、オープンなWebインターフェイスは堅持
  - その結果、他のテクノロジーからも容易に利用可能に



考察

- 既存アプリケーション資産を活かしながらWeb化する「定番」パターン
  - システム機能をWebサービス化すること
  - 機能をモジュール化していることが前提となる
- Senchaは、Web APIとのオープンなインターフェイスを備える
  - Senchaにおいて接続性を考慮するより、既存システムにオープンな接続性を用意することが重要
- クラウドなどのサービス展開が容易に
  - 現在市場にあるクラウドサービスと同じような機能形態、サービス提供が可能に

Web APIがビジネスモデルを変えることにも着目すべき

ちょっと事例から離れて考察...

- 最近のプロジェクトのトレンド「マイグレーション」

いわゆる「マイグレーション」だと若干後ろ向き

「モダナイゼーション」で積極的な展開に

最近のデベロッパーキャンプセッションから…

- 株式会社フォーラムエイト
  「UC-win/Road」
  - 3次元リアルタイムVRソフトウェア
  - 業界に先駆け2000年にリリース
  - Delphiによりアプリケーションを構築
- 64-bit化のチャレンジ
  - 従来バージョンは32-bit演算で処理速度、処理能力に限定
  - 新バージョンで64-bit化を実現
  - 膨大なVRデータを処理可能に
    - 長距離道路シミュレーションが20×20kmから400×200kmに拡大
    - 防災、都市計画など、新しいVRソフトウェアの需要に対応し、津波・氾濫、風、音響など解析結果の長時間可視化、地形空間の拡大、分析能力の向上を実現

https://youtu.be/qa2XcKAo-20

64-bit化とは？

- Delphiのビルドターゲットを「64-bit Windows」にすることなのですが…
  - 以下のような処理を64-bit対応にすることが必須

✓ メモリ割り当て（広義にはデータの保持の仕方）
✓ ポインタ演算
✓ 数値計算（数値データのサイズも関連）
✓ 使用するライブラリ（64-bit版）
✓ 使用するコンポーネント（64-bit版 – ただしIDE内で使用する32-bit版も必要）
✓ 使用するドライバ（64-bit版 – ただしIDE内で使用する32-bit版も必要）

64-bit化のメリットを正しく理解し、64-bitサポートを実装するべき

C++の場合…

- C++では、さらにコンパイラのベースアーキテクチャの変更が加わるため、マイグレーションにもう少しフェーズ分けの考え方が必要

STEP 1

- bcc32により最新バージョンへ移行
- ゴール：
  - 最新のWindowsプラットフォームのサポート
  - 最新のC++Builderベースのコード

最新のWindowsプラットフォーム
サポートとソフトウェアアップデート

STEP 2

- STEP 1のコードをベースにbcc64へ移行
- ゴール：
  - 64-bit Windowsプラットフォームのサポート
  - 新しいC++言語標準への対応

64-bitメモリ空間を利用した
アプリケーション性能の強化

STEP 3

- STEP 2のコードをベースにbcc32xへ移行
- ゴール：
  - 64-bit / 32-bitでの単一コードベースの実現

単一コードベースによる
メンテナンス性の向上

コード品質に関する問題

- 既存アプリケーション資産を「マイグレーション」する場合…
  - 「なるべくコードを改修しないようにしたい」というのは願い
  - とはいえ、コードの品質はどうだろうか？
- 品質に問題があるコードをそのまま「マイグレーション」した場合
  - 問題のあるコードが「ブラックボックス」化
  - 潜在的なバグを生み出す遠因となる
  - 余計なメンテナンスコストが発生する恐れ
- 最低限やっておきたいのは既存コードの品質チェック
  - 必要に応じてリファクタリングも実施
  - 定量的な品質チェックを実現するには、Kiuwanに期待

Idera, Inc.
Test Tools

kiuwan
www.kiuwan.com

マイグレーション後のアプリケーションの寿命、用途を考えてスコープを設定すべき

再び事例に…

- Miniatur Wunderland Hamburg
  - 世界最大級のジオラマワールド
  - 15分で昼夜サイクルを完了するミニチュアワールド
  - 列車、車、飛行機、船舶、385,000個以上のライトが連動
  - これらすべてを集中管理／制御するシステムをDelphiで構築

実質的に最大規模の「分散IoTシステム」と言える！

車両制御システム

交通状況に応じて現実的な
交通パターンを採る制御を実施

超コンパクトな
バイナリプロトコル

指令分配システム

ナフィンゲン空港用のディスプレイパネル

空港の車両制御システムと同期

HSVスタジアム ジャンボトロン

ジャンボトロン画面に表示される
エンターテインメントをDelphiで制御

照明制御システム

385,000個以上のライトを
時間軸に応じて個別に管理

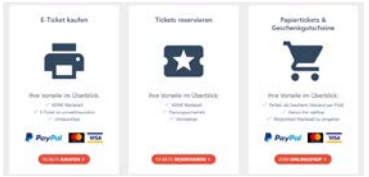
列車制御システム

稼働する列車の位置情報を
入力とし適切な運行を制御

気候制御システム

温度、湿度を
適切に管理

チェックアウト／予約／発券システム

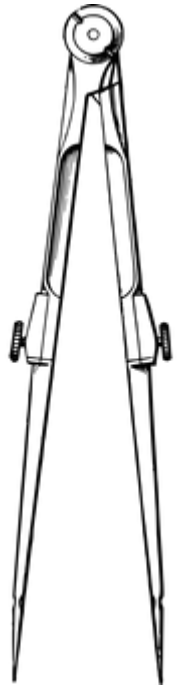


ギフトショップ、レストラン、
予約、発券を管理

ITインフラストラクチャは故障から守るために意図的に分散配置

考察

- 肥大化するシステムで、分散アーキテクチャによるリスク分散は有効
  - システムの依存関係を把握すること
  - 個々のシステムの自律性の確保と依存性の低減を設計思想に盛り込む
- システムを構成する要素技術はいずれも既存のDelphiでおなじみのもの
  - デバイスI/Oの実装方法は千差万別
  - システムで利用するデータはRDBMS等に格納される（普通の多層C/Sシステム）
  - とはいえ、これらを制御するプロトコルを統一することで「システム系」を容易に制御可能に
- 接続性がシステムの堅牢性と拡張性を提供
  - つながる仕組みを持つシステムを構築していくことで、日々拡張し、機能強化するシステムの構築、メンテナンスが可能に



システムの設計（アーキテクチャ）が長期的な成功を生み出している好例

接続性に関するヒント

- 通常のRDBMS、クラウドデータ、エンタープライズデータへの接続は
  - いずれも現在ではFireDACに統一すべき
  - ターゲットデータソース、システム形態、プラットフォームなど、あらゆる点で汎用性、可搬性が高い
  - 旧システムからの移行、最新バージョンへの対応、パフォーマンスなどの点でも有利
- 各種デバイスとの接続については
  - RS-232Cなど従来からの接続法も引き続き有効
    （コーディングレベルでは、文字コードの問題に対処する必要あり）
  - Bluetooth、WiFi、IoTなど専用のコンポーネントを使うのも手
    （抽象度が高まり効果大）
- オープンな接続性を持たないレガシーシステムの場合でも
  - バッチ処理など、古いアーキテクチャをカプセル化するインターフェイス層を用意することで、新しいシステム系へと引き入れることが可能

もうひとつ事例を...

- ManagementPlus
  - 眼科向けシステムのトップ企業「Eye Care Leaders」のグループ企業
  - 眼科・アイケア専用電子健康記録／実務管理ソフトウェアを提供
  - QAスペシャリスト１名とシニアレベルの常勤開発者３名の小規模開発チームで製品を開発
- 増大する法令要件への対応が重要な課題
  - 法令要件の変更、増大に迅速に対応する開発スピード
  - 要件を満たしていることを確実にするQA
  - 小規模チームで生産性を高め、カバレッジを上げるためのツールの力を活用

小規模チームでの生産性向上／品質向上のポイント

- RADの生産性はもちろんだが、それだけでは十分ではなく…
- チームとしての生産性を高めるために、構成管理ツールの利用は必須
  - 分業を可能にする構成管理ツール
  - バージョンの差異に起因するトラブルはつきもの
  - 細かいリビジョン管理は、複雑な要件に対応する際の手助けとなる
- テストプロセスを整備することは品質向上とその管理工数削減に寄与
  - 開発プロセスに組み込まれた単体テストの実行
  - UIテストの自動化やコードセキュリティチェックなど、ツールで解決できる点は積極的に採用

ASSEMBLA

Ranorex

Assembla、Ranorexなど、活用できるツールをいくつも用意！

現実的な解決策が見つけられない方のために…

- 既存アプリケーション資産を抱えつつ、次へ進まなければならない場合
  - 現在の資産をどれだけ有効活用するべきか？
  - 新しい要求に対応するための労力は？
  - スクラッチで開発すべき？
  - パッケージアプリケーションに切り替えるべき？

「過去を捨てずに未来へ」をベースに大局的視点で選択を…

THANKS!

www.embarcadero.com/jp

第36回 エンバカデロ・デベロッパーキャンプ