

# 31ST EMBARCADERO DEVELOPER CAMP

第31回 エンバカデロ・デベロッパーキャンプ

【T6】テクニカルケーススタディ

「iPod Touchによる工場内のモバイル化  
- 鋼材品質証明と基幹システム連携 -」

株式会社ミガロ.  
吉原 泰介

**embarcadero**



# 株式会社ミガロ.

<http://www.migaro.co.jp/>

## 会社情報

所在地: 本社 大阪市浪速区湊町2-1-57  
難波サンケイビル13F

東京事業所 東京都港区麻布台1-4-3  
エグゼクティブタワー麻布台11F

## 事業内容

### IBM i 向けのソフトウェア・ツール販売および技術サポート

#### 【開発ツール】

- ・Delphi/400
- ・JC/400、SmartPad4i

#### 【スマートデバイス向けツール】

- ・Business4Mobile

- **Delphi/400** — DelphiをIBM i (AS/400)に完全対応させたミドルウェア — 国内約700社、全世界約6,000社の導入実績

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

# アジェンダ

- 豊鋼材工業株式会社様について
- iOSモバイルを使った鋼材品質証明
- iOSアプリ開発Tips
- Delphi による iPod Touch以外のモバイル化対応
- まとめ

「iPod Touch による工場内のモバイル化」

# 豊鋼材工業株式会社様について

# 豊鋼材工業株式会社様

創造 ファクトリー  
豊鋼材工業株式会社  
since 1958

商号 豊鋼材工業株式会社  
(本社 福岡県)

設立 昭和33(1958)年6月4日  
事業内容 鉄鋼および各種金属の加工および販売  
資本金 4億5,000万円  
代表者 清水 豊  
取扱量 25万7千トン  
従業員数 188名  
<http://www.yutaka-steel.co.jp/>

# 豊鋼材工業株式会社様

## 事業所所在地

**本社:**

福岡県糟屋郡篠栗町

**工場:**

福岡、苅田、長崎、  
宮崎、鹿児島

**支店・営業所:**

北九州、長崎、熊本、宮崎、  
中国、大分、佐世保、  
鹿児島、沖縄



# 豊鋼材工業株式会社様

## 事業内容

鉄鋼およびその他金属の加工、販売



総販売量

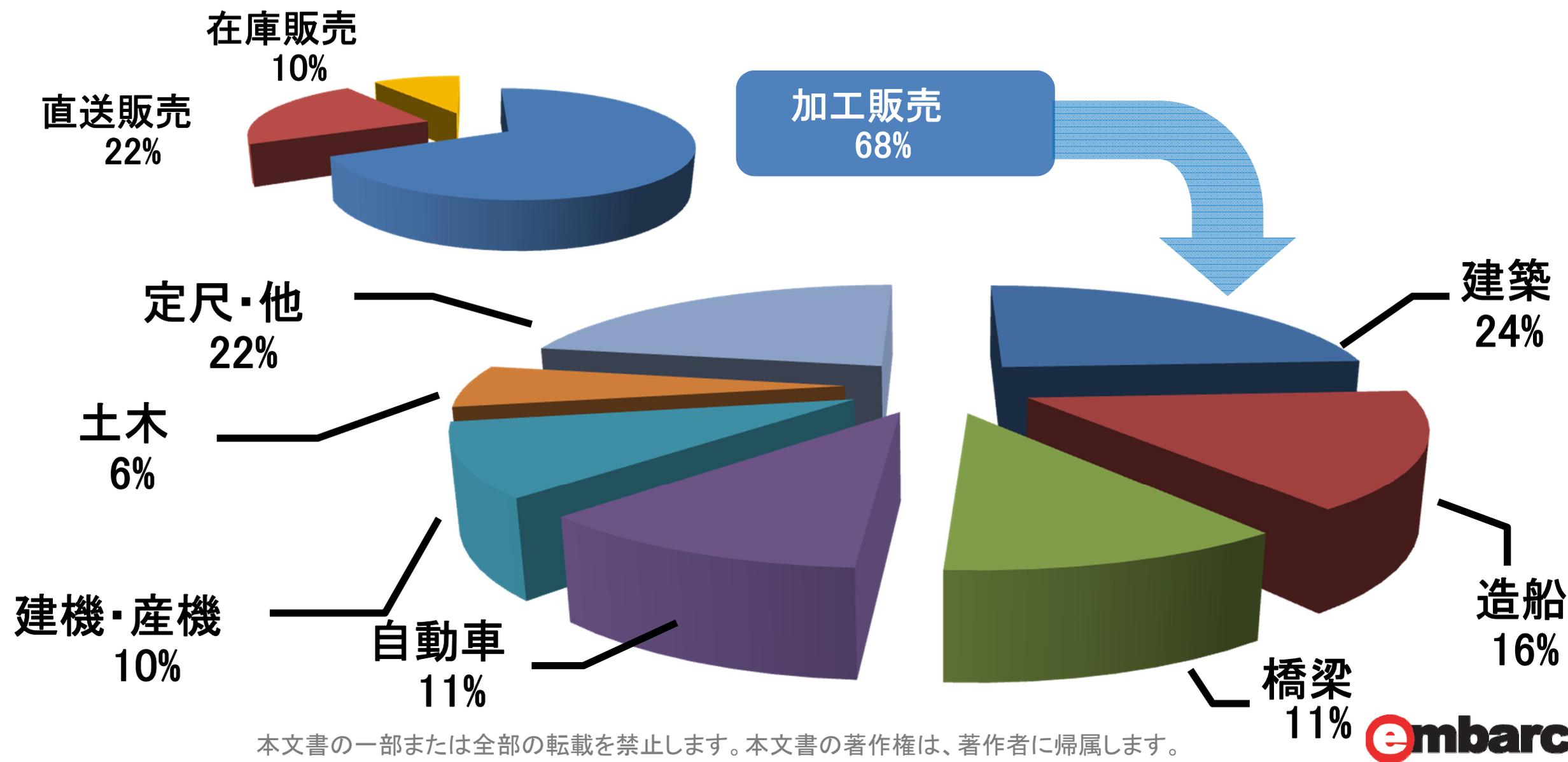
≒ 年間25.7万トン（連結）

鉄鋼およびその他金属の二次加工、販売



# 豊鋼材工業株式会社様

## 用途、販売先



「iPod Touch による工場内のモバイル化」

# iOSモバイルを使った鋼材品質証明



製造総括部長  
兼 製造総括課長  
石井裕昭 様

## 工場内モバイル化のねらい

- 現在、食品に限らずさまざまな産業分野で偽装防止への取り組みが進んでいる

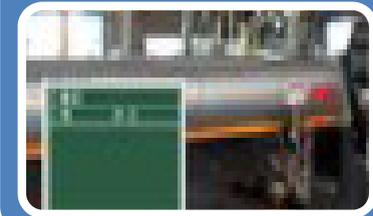
もちろん鋼材分野も例外ではない！

鋼材は見た目では品質判断が難しく、品質は構造物の安全性などに直結し、非常に重要 → 管理強化したい

(お客様から品質証明書類を求められることが増えてきた)

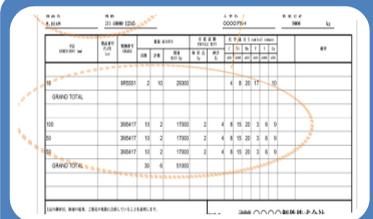
# 工場内モバイル化のねらい

- これまでの品質証明作業



## 加工工程

- 指定タイミングの鋼板状況写真



## 加工終了後

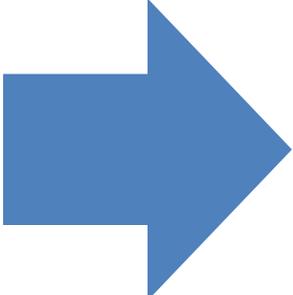
- ミルシート(鋼材メーカーの証明書)の提出

→ 鋼板を加工タイミング毎にデジタルカメラで撮影し、  
お客様のご要望時に写真ファイルを手作業で整理して提供

これらの作業や写真管理を効率化、正確性を高めるために  
モバイル端末を導入することを検討！

# 工場内モバイル化のねらい

- 細かいニーズへ柔軟な対応ができる自社開発に決定



## モバイル 機器

- 低コスト実現の為、工場内Wifiに使用を限定して iPod Touch (第五世代) を導入

## アプリ

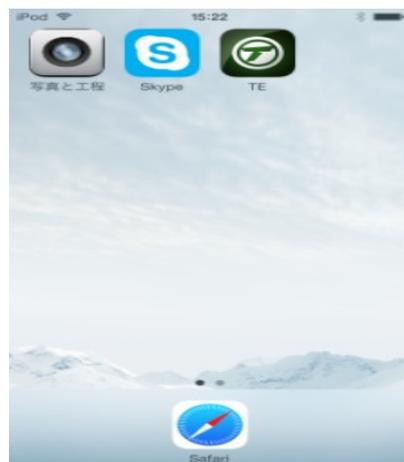
- iPod Touchで動作するネイティブアプリは Delphi/400で開発
- iOSアプリは、通常のAppStore向けではなく、Enterpriseライセンスを使って社内配信



# 開発項目一覧

## 開発検証機能の全体概要

分類	項目	内容、備考
<b>新規Delphi 開発機能</b> 	<ul style="list-style-type: none"> <li>・<u>工程、鋼材ステンシル管理カメラ</u></li> <li>・作業予定ロット、順番公開</li> <li>・<u>生産実績入力</u></li> <li>・稼動日報</li> <li>・出荷準備(梱包)済情報</li> </ul>	<ul style="list-style-type: none"> <li>・新しいDelphi開発環境にて、従来管理できていなかった情報を取得する。</li> <li>・入力業務を簡易化するアプリを開発し、効率化と管理レベル向上を図る。</li> </ul>
<b>既存5250 業務</b> 	生産実績入力、出荷チェック 現品票作成、 ロケーション管理、棚卸し 等	ハンディーターミナル用の各種業務メニューをWaveLink TEで表示し、既存業務を行う。既存機器の1/3以下のコストで機能を実現。
<b>コミュニケーション 機能</b> 	Skype(無料通話)による ユーザー間通話、メッセージ機能	無線LANエリア内での通話を可能化。 またはメッセージ、ビデオ通話など

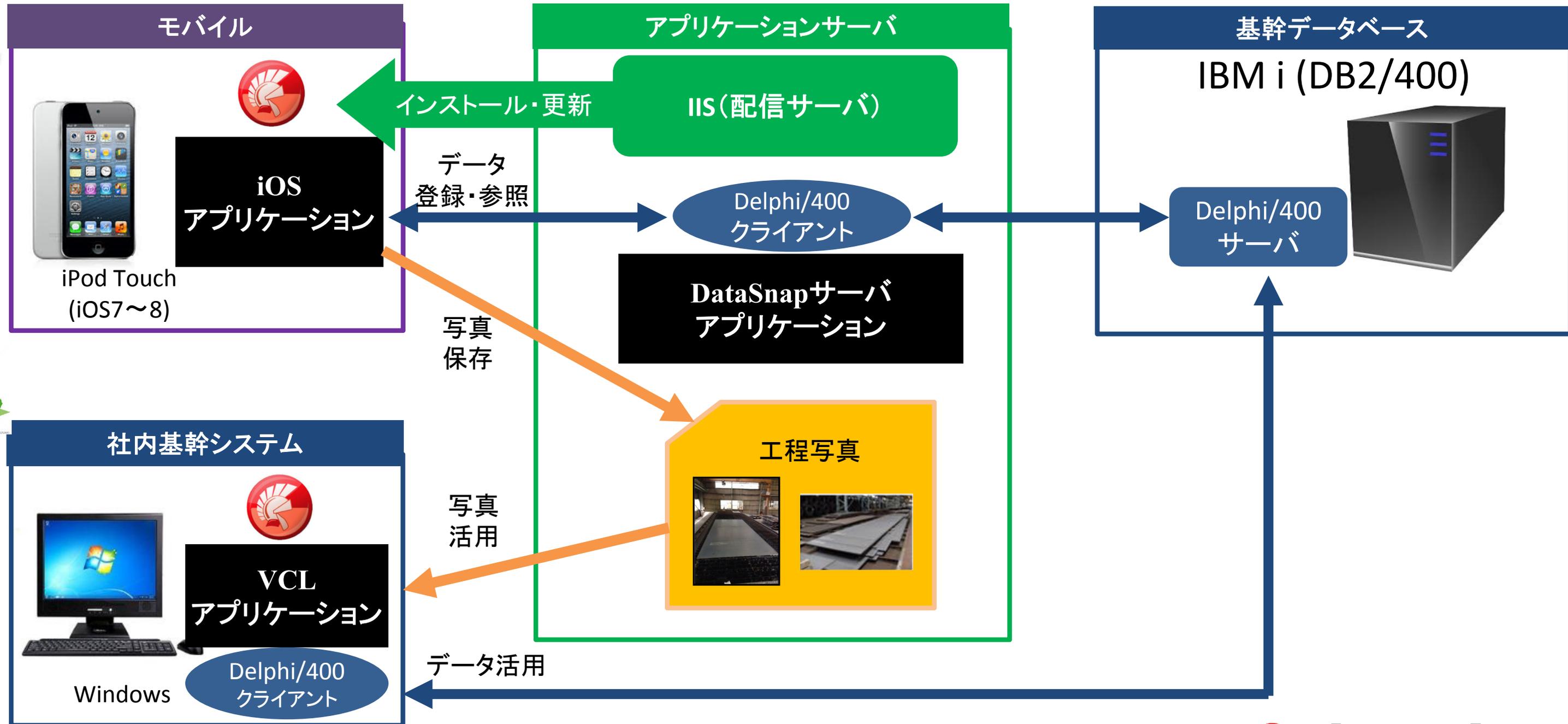


# 目的と経緯

## 工程写真・鋼材写真撮影の機能開発

分類	目的	従来の課題と経緯
関連作業 効率化	<u>1台/機種保有</u> でカメラ探しをなくす	限られた台数の <u>デジカメを共有使用</u> 、無い場合は都度探す必要があった。
	<u>撮影をオペータ作業に完全移行</u> 、撮影待機(スタッフ、オペータ共)、連絡ロス時間をなくす	工場スタッフ撮影時は対象材処理の都度連絡、時間確認必要。待機時間も無駄。
	工場スタッフのカメラ(メモ리카ード)回収、データ仕分作業をなくす	<u>写真データ回収が必要</u> 、回収データの客先仕分けは黒板の工事名を確認必要
	提出書類と写真の照合半自動化	板番、ロット番号等との対応も写真画像を一枚毎に <u>確認する手間</u> を要する。

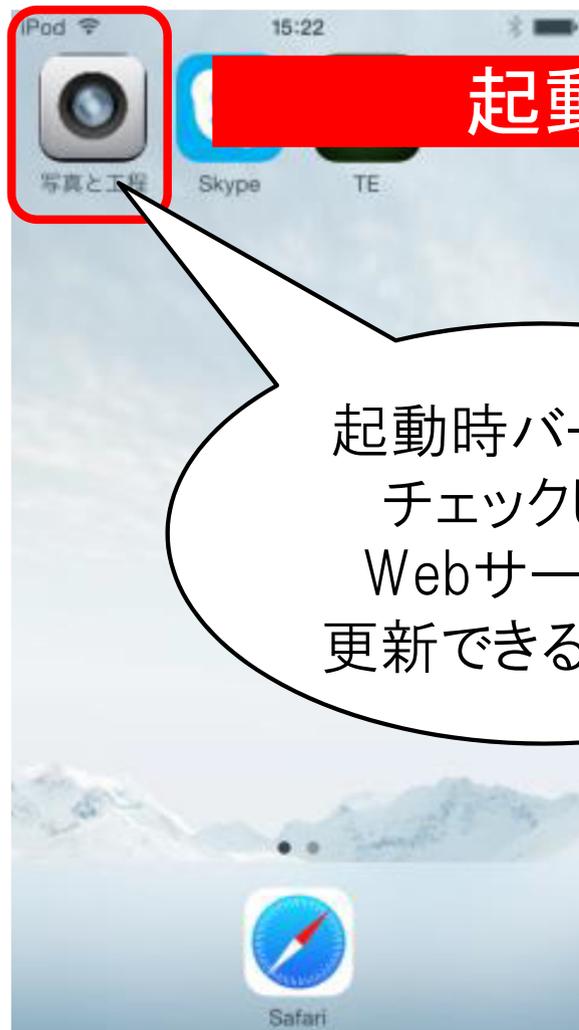
# システム構成



# iPod Touchを使った業務機能(一例)

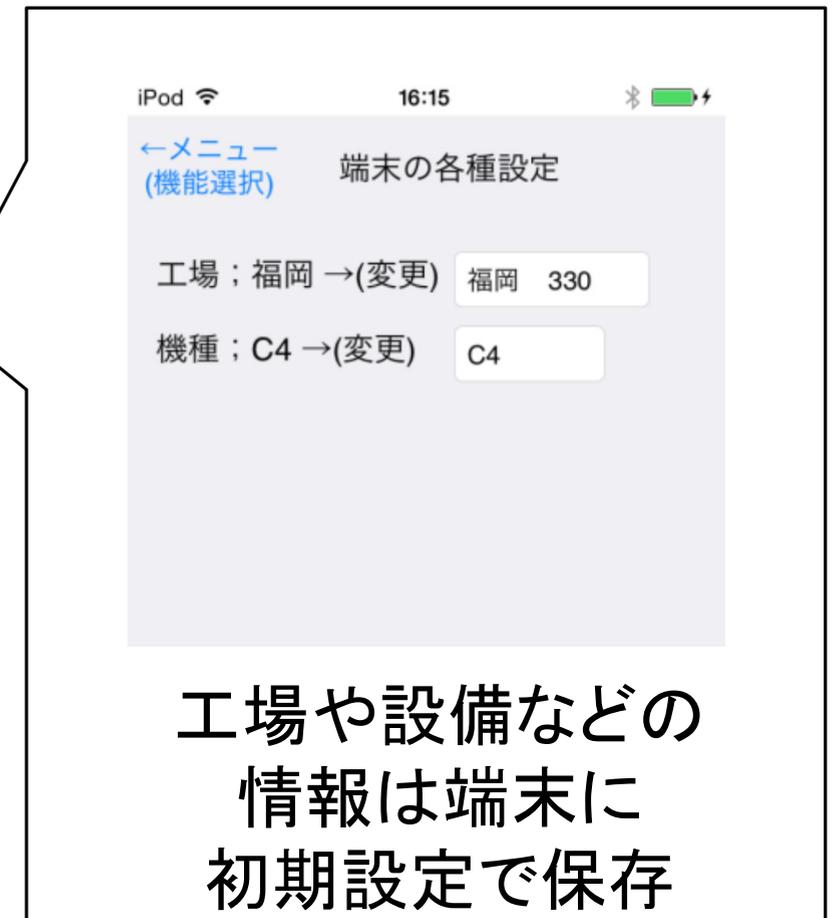


- 指示されたロット番号情報と共に写真を撮るだけでデータや写真を基幹システム上で管理できる

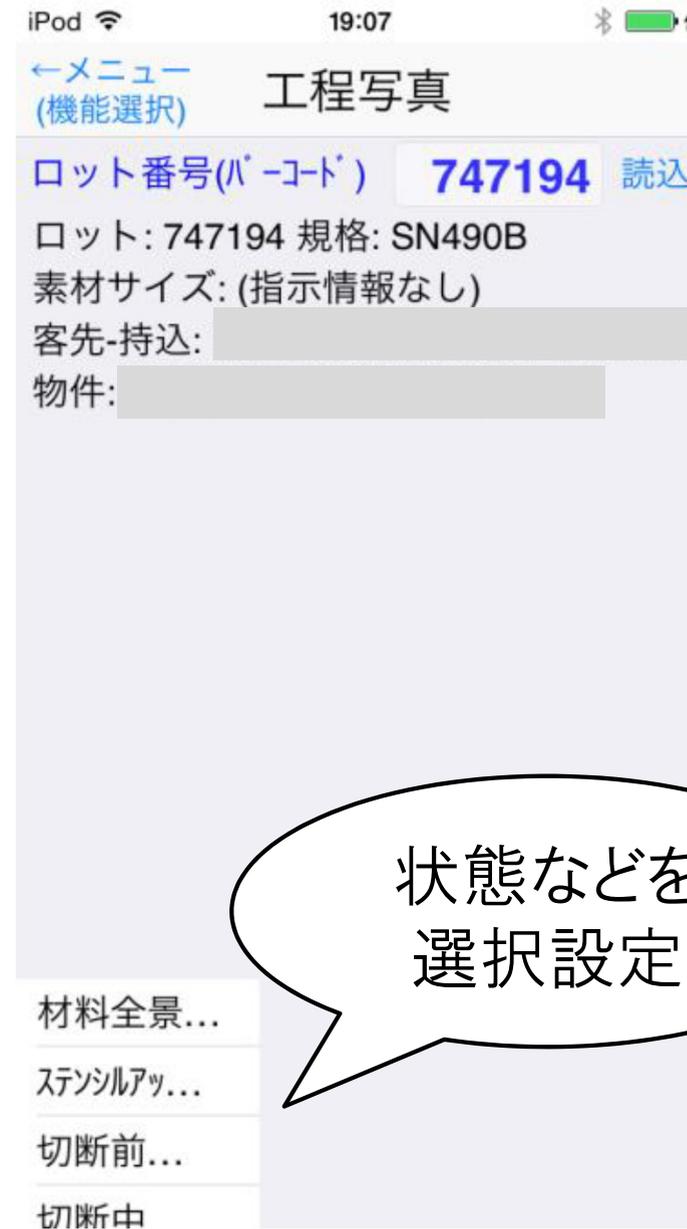


起動

起動時バージョン  
チェックして、  
Webサーバより  
更新できる仕組み



# iPod Touchを使った業務機能(一例)

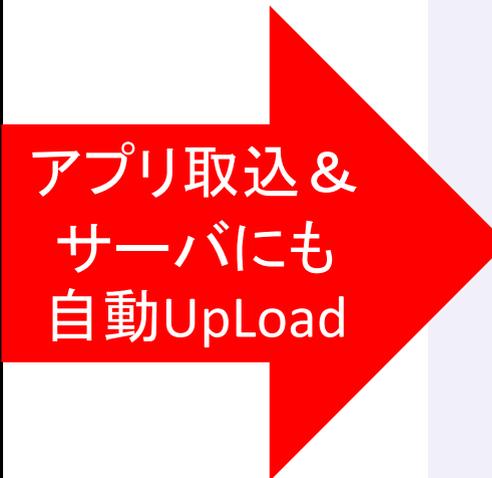


状態などを  
選択設定



作業指示書の  
QRコードから  
ロット情報を読み取る  
(読取音なども制御)

# iPod Touchを使った業務機能(一例)



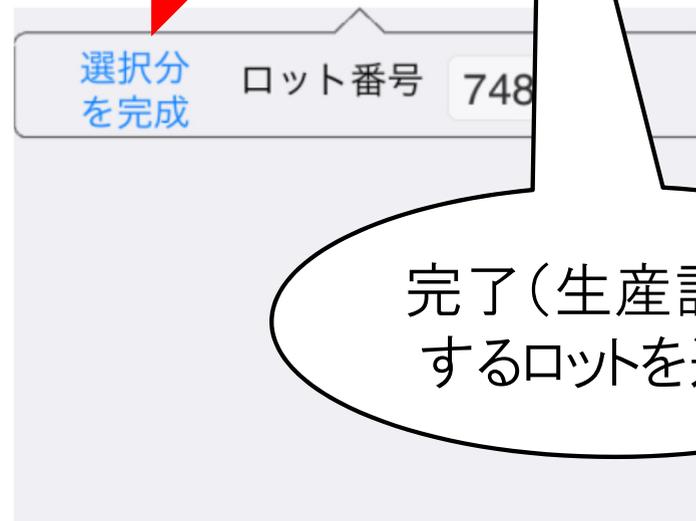
# iPod Touchを使った業務機能(一例)



**起動**



**完了**



完了(生産計上)するロットを選択



スライドで残鋼板の形状選択



残鋼板の寸法や山番などを入力して完成

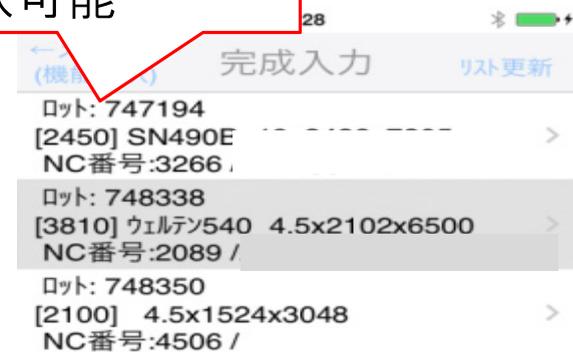


# iPod Touchを使った業務機能(一例)

## 特徴

- ① 残材の発生が無い場合はタッチ操作のみで完結
- ② 残材発生時は発生した形状パターンを図で選択可能
- ③ 形状の位置関係どおりに寸法入力可能

リストから対象  
選択可能



残材無ければ  
タッチ

残材有無選択



形状パターンを選択



寸法等を入力



残材入力後に  
タッチ



# iPod Touchからのデータを基幹システム活用(一例)



- 営業担当者が社内システムでIBM i の管理データを読み込み、お客様毎、案件毎に写真やデータにアクセス・自動整理できる

The screenshot displays a web application interface for material management. It includes several sections:

- ロット・素材実績** (Lot/Material Performance): A table listing production lots, materials, and quantities.
- 工程写真明細** (Process Photo Details): A table showing production steps, dates, and photo links.
- 素材写真明細** (Material Photo Details): A table showing material specifications and photo links.
- 一括処理** (Batch Processing): A section for managing photo uploads and corrections.
- 表示** (Display): A section for adjusting the view and zoom of the photos.

The main area shows a large photo of a material surface with a grid pattern, and a smaller photo of a material surface with a grid pattern. The interface is in Japanese and includes various navigation and control elements.

自動整理された  
写真データをお客様の  
ご要望時にメール等で  
すぐに提供可能！  
(正確化・迅速化を実現)

# iPod Touchからのデータを基幹システム活用(一例)

Y-VIS(MP) -- 新生産・在庫管理メニュー  
 素材在庫関係(M) 生産・出荷状況(P) 各種資料 | 旧業務メニュー | 特定お客様メニュー | インターネット、西面共有 | 帳票印刷

履歴明細 | 各種生産実績集計 | 写真整理

ロット・素材実績

生産日	ロット	素材品種	規格	厚	板番
▶ 150107	747469	他メーカー	SM490A	28	7545622
150107	747470	他メーカー	SM490A	32	7545502
150107	747482	NSSMC建材	SM490A	19	28120101
150108	747587	Y発生不定尺	SM490A	12	66238001
10109	747613	NSSMC建材	SM490A	16	34732202

表示ロット (1/5) ↑ ↓ ⚡ ⌂ Excel出力 印刷

表示ロット: 747469 (1/5) | ロット番号: 747469 | 直近撮影日: 150106 | 撮影枚数: 4

素材品種: 他メーカー | 規格: SM490A | 厚: 28 | 板番: 7545622 | 摘要(物件): 新博多ビル01節柱仕口単品BH

1/4 3KA-747469-150106172645.JPG | 2/4 3KS-747469-150106172715.JPG | 3/4 3SA-7545622-150106172844.JPG | 4/4 3SS-7545622-150106172413.JPG

一括処理 (対象外はチェックを外す)  
 ファイル名:  そのまま  簡略(ロット+連番)  ロット+状況+連番  
 保存先:  そのまま  別の場所に移動  別の場所のコピー  
 実行

工程写真明細 (表示ロット: 2/2)

工場	区	ロット	状況	撮影日	時刻	保存先
▶ 330	K	747469	材料全景	150106	172647	¥¥FLUE
330	K	747469	ステンシルアップ*	150106	172717	¥¥FLUE

素材写真明細 (表示ロット: 2/2)

工場	区	板番	品番	撮影日	時刻	保存先
▶ 330	S	7545622	8	0	150106	172347 ¥¥FLUE
330	S	7545622	8	0	150106	172416 ¥¥FLUE

表示:  スクロールバー  ドラッグ  
 パンウィンドウ:  表示  
 S C M C L  Xボタン  
 保存先を開く

ファイル名: 3KA-747469-150106172645.JPG | 表示モード:  スケール  ストレッチ  実寸  幅  高さ | 縮尺: 0.5

左の選択行のロットの  
写真一覧

写真撮影の  
ロット一覧

上記選択ロットでの  
写真明細

上記ロットの使用鋼材の  
写真明細

上の写真一覧で選択分  
の拡大写真

# iPod Touchからのデータを基幹システム活用(一例)

## 抽出した写真の要否選択

表示シート (1/5)    ロット番号 747469    撮影枚数= 4 枚  
直近撮影日 150106

素材品種	規格	厚	板番	摘要(物件)
他メーカー	SM490A	28	7545622	新博多ビル01節柱仕口単品BH

チェック	ファイル名	修正	説明
<input checked="" type="checkbox"/>	3KA-747469-150106172645.JPG	<input type="checkbox"/>	材料全景
<input checked="" type="checkbox"/>	3KS-747469-150106172715.JPG	<input type="checkbox"/>	ステンシルアップ
<input checked="" type="checkbox"/>	3SA-7545622-150106172344.JPG	<input type="checkbox"/>	材料全景
<input checked="" type="checkbox"/>	3SS-7545622-150106172413.JPG	<input type="checkbox"/>	ステンシルアップ

不要であれば  
チェックを外す

工程写真

鋼材写真

## 写真ファイル名の自動整理機能

一括処理  
(対象外はチェックを外す)

ファイル名  
 そのまま  
 簡略(ロット+連番)  
 ロット+状況+連番

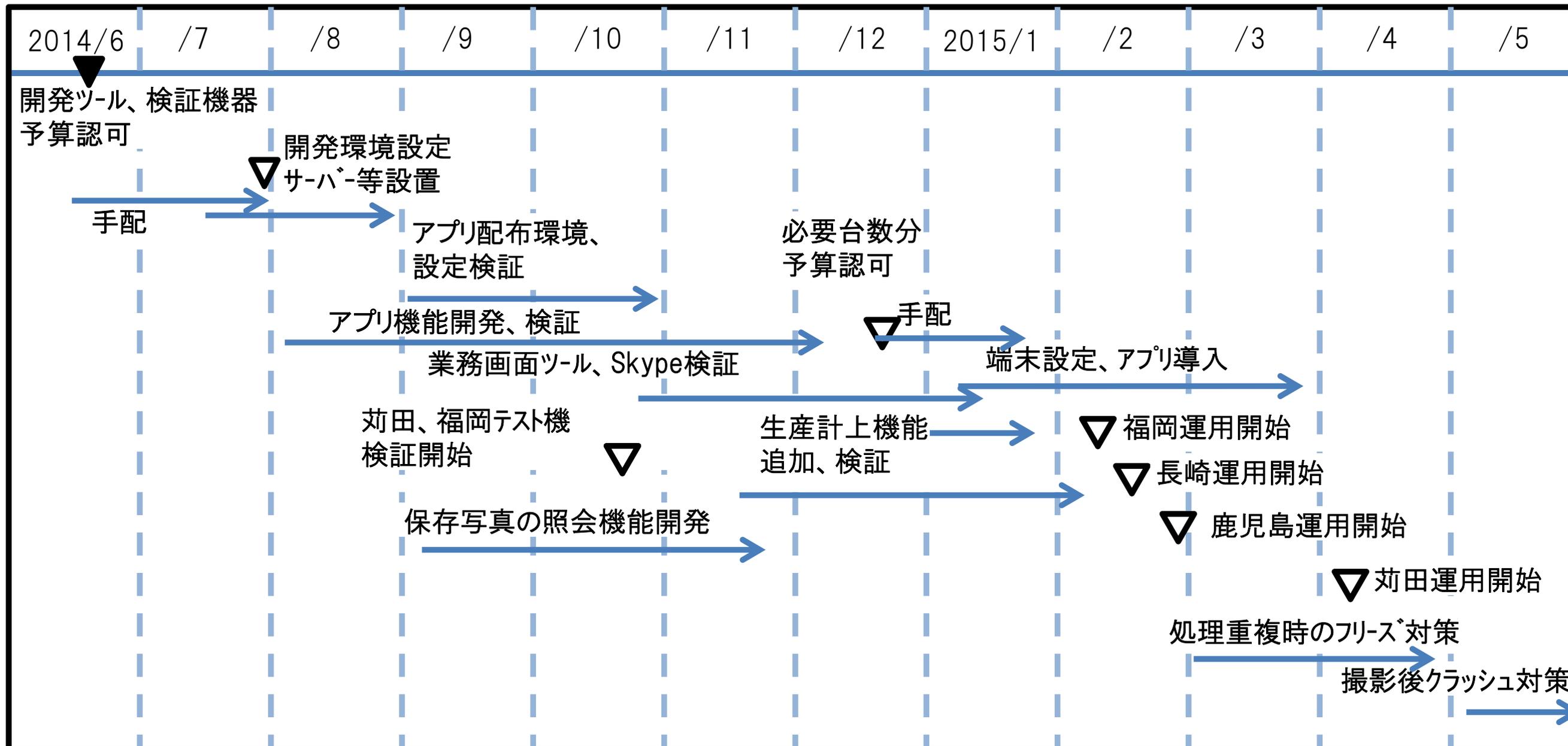
保存先  
 そのまま  
 別の場所に移動  
 別の場所にコピー

実行

それぞれ指定

上記指定の後  
クリック

# 立上げ、稼働スケジュール



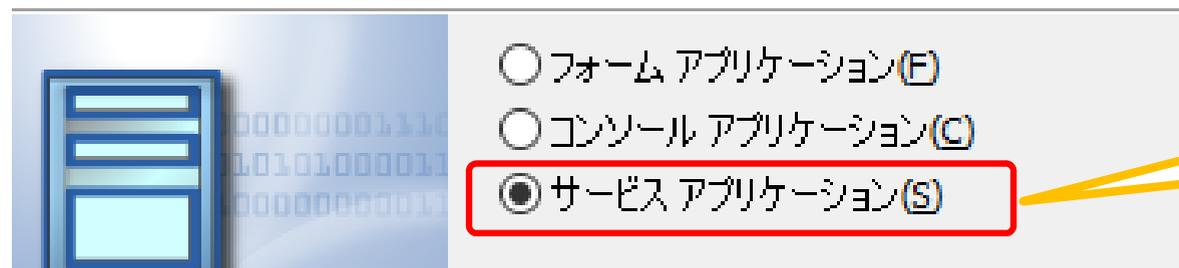
「iPod Touch による工場内のモバイル化」

# iOSアプリ開発Tips

- ①DataSnapアプリケーションの形式
- ②写真の撮影
- ③撮影写真の転送
- ④写真のExif情報
- ⑤バーコード、QRコードの読み込み
- ⑥オフライン処理
- ⑦更新アプリの配布

# ①DataSnapアプリケーションの形式

- DBへの処理はDataSnapサーバ経由で実装  
基本的な実装方法は第29回の  
「こんなに簡単！ Delphi によるiOS/Android業務アプリケーション開発！」を参考にしてください。  
<http://edn.embarcadero.com/jp/article/images/44161/t2.pdf>



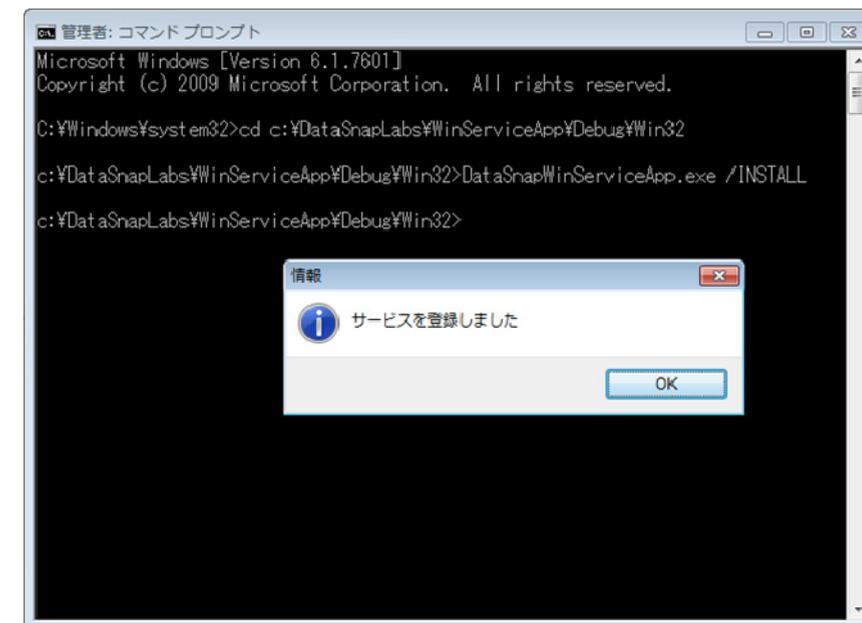
開発時はフォームアプリ形式でも構いませんが、  
運用時はサービスアプリ形式が便利(ログアウト可)

作成したDataSnapアプリケーションは、管理者権限で Windows のコマンドプロンプトを開き、以下のコマンドを実行してサービスを登録できます。

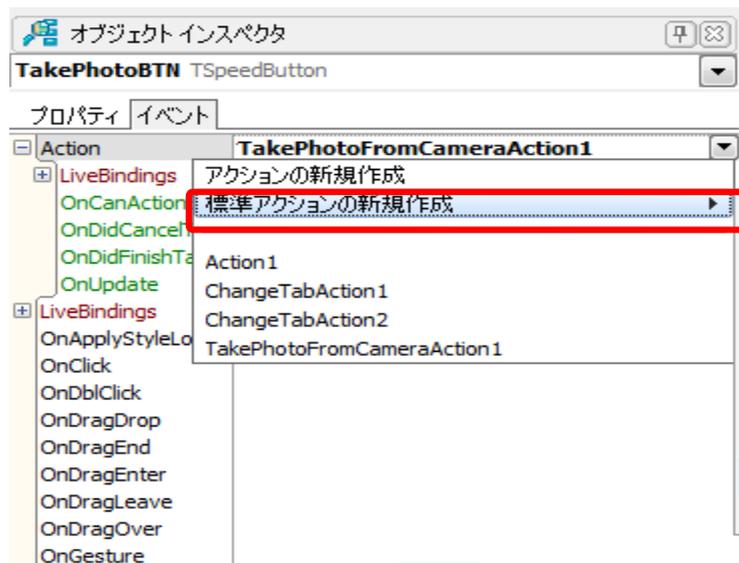
[インストールコマンド]  
ファイル名 /INSTALL

[アンインストールコマンド]  
ファイル名 /UNINSTALL

コントロールパネルからサービスを確認すると、“ServerContainer1” (デフォルト)等の作成したサービスが表示されますので、右クリックから「サービスの開始」でアプリケーションを開始します。

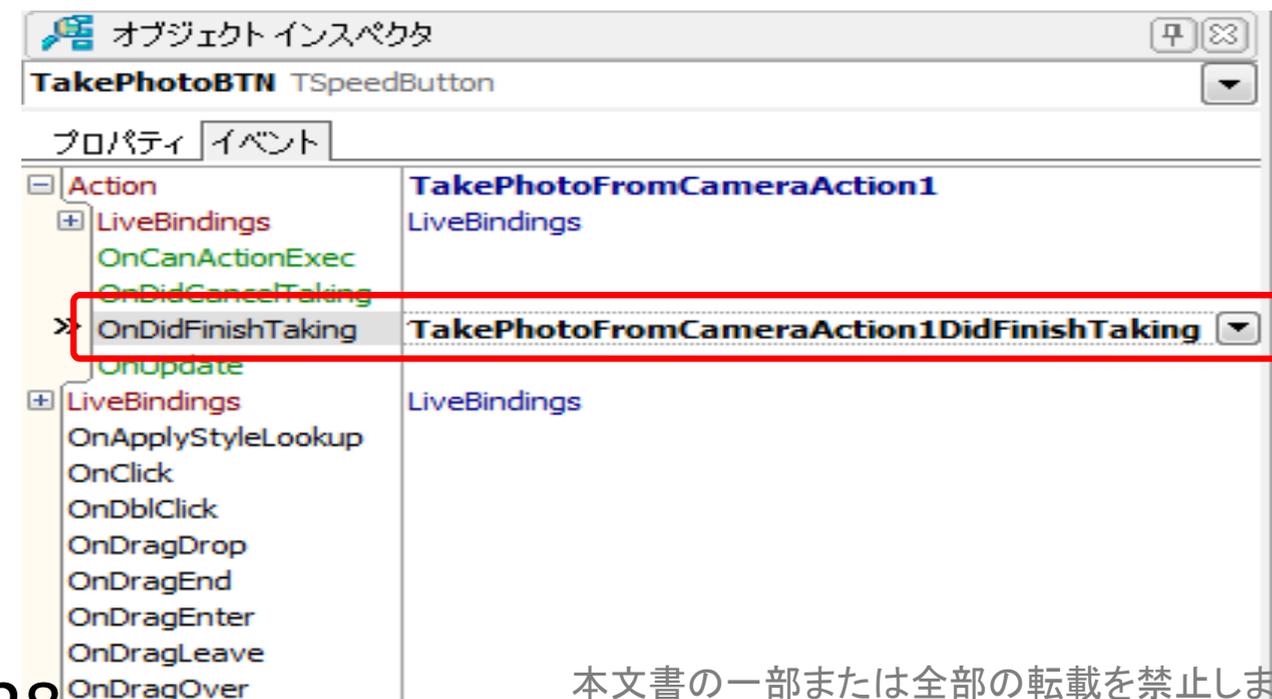


## ②写真の撮影



ボタンのAction  
 →標準アクションの新規作成  
 →メディアライブラリ  
 →TTakePhotoFromCameraAction

でボタンクリックでカメラ起動・写真撮影可能になる



撮影後の取得画像の処理は  
 OnDidFinishTakingイベントでコード記述

(HELPより)  
 デバイスのローカルライブラリから写真を取得するプロセスが完了した際に発生します。  
 OnDidFinishTaking イベントハンドラを記述すると、取得した写真を処理することができます。



## ②写真の撮影

```

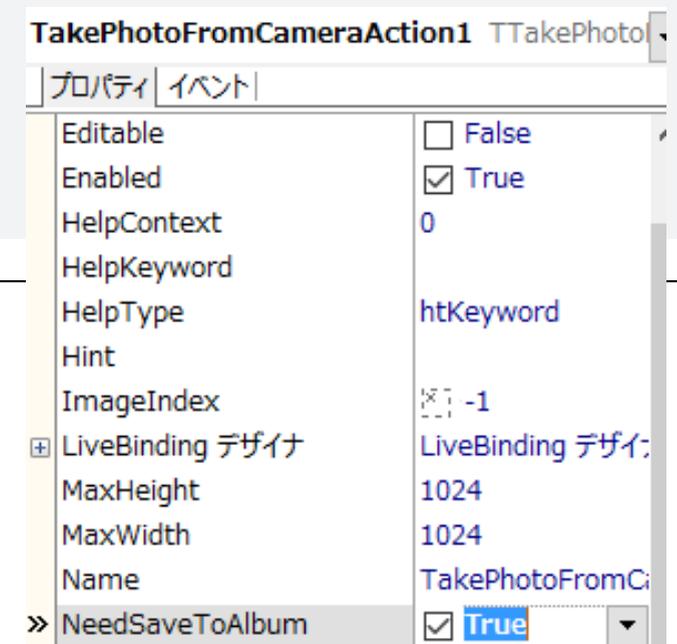
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
var
  FileName : String;
begin
  //MultiIMGというTimage(画面表示用)に取得画像を表示
  MultiIMG.Bitmap.Assign(Image);
  //ここでIMG.jpgというファイル名で表示されている画像を保存
  FileName := TPath.Combine(TPath.GetDocumentsPath, 'IMG.jpg');
  MultiIMG.Bitmap.SaveToFile(FileName);

  //IMG.jpgをサーバーに転送する場合、処理を記述

end;

```

アルバムに保存しておく場合は、TTakePhotoFromCameraActionのNeedSaveToAlbumプロパティをTrueで設定しておく。  
 ※XE7以前の場合は、usesにiOS関連のユニットを追加してUIImageWriteToSavedPhotosAlbumを使用して独自に実装を行う。



## ②写真の撮影

- 撮影実績枚数(2015年5月分)

工場	工程写真					鋼材写真			合計
	全景	切断前	切断中	ステンシル	小計	全景	ステンシル	小計	
福岡	140			89	229	121	114	235	464
苅田	200	4	164	410	778	270	223	493	1,271
長崎	12		6	11	29	53	51	104	133
宮崎					0	3	3	6	6
鹿児島	15		25	32	72	117	72	189	261
合計	367	4	195	542	1,108	564	463	1,027	2,135

ファイルサイズ 平均 1.3MB/個 → 約 3GB/月 → 約40GB/年  
 サーバーのDドライブ容量 700GBなので 15年以上は使用可能 と推定

当然 バックアップは必要 → USBのHDDへ定期自動バックアップ

## ③撮影写真の転送

- 撮影した写真をTIdFTPコンポーネントで転送  
(DataSnapで転送する機能を用意しても可能)

```
IdFTP1.Host := '999.999.999.99'; //保存先ファイルサーバのIPアドレス
IdFTP1.Username := 'ユーザー名'; //FTPで許可されたユーザー
IdFTP1.Password := 'パスワード'; //パスワード
IdFTP1.Port := 21; // ポート番号
IdFTP1.Passive := True; // パッシブモード
IdFTP1.Connect; // サーバーに接続する
//保存するフォルダに移動(任意で設定してください)
DestDir := '任意のサブフォルダ名(実在)'; //親はFTPで設定しているルートフォルダ
IdFTP1.ChangeDir(DestDir);

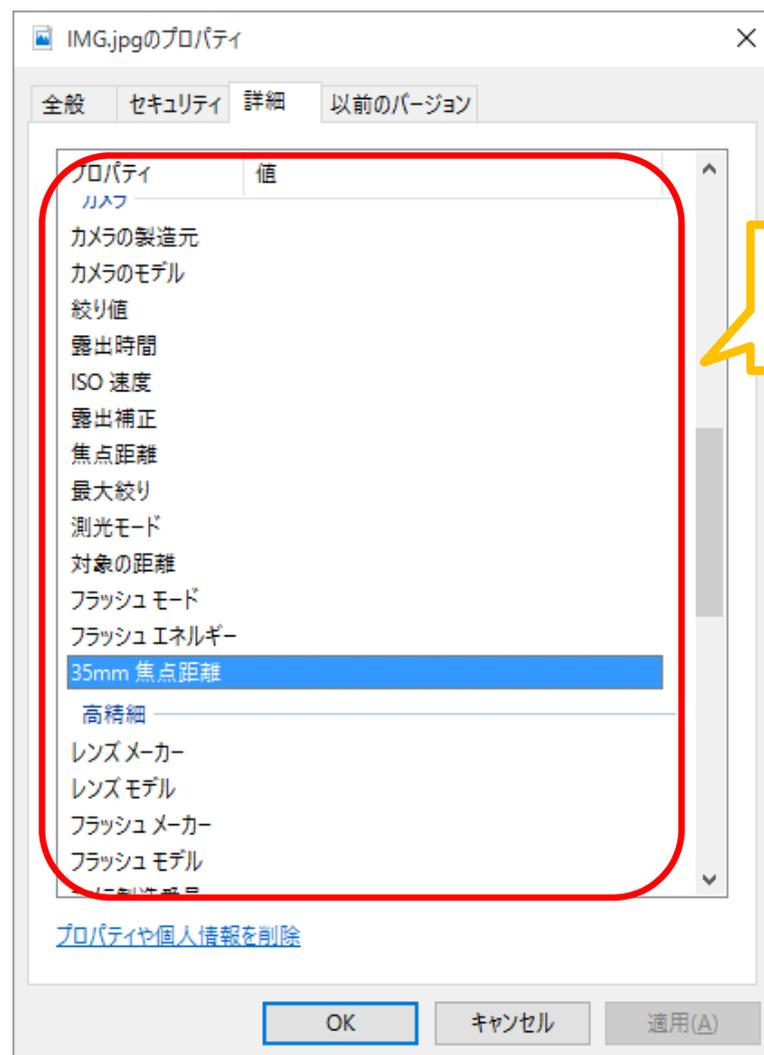
//アップロードする画像のファイル名(画像のファイル名は撮影後に命名したもの、下記は例)
SourceFile := TPath.Combine(TPath.GetDocumentsPath, 'IMG.jpg');
//アップロードするファイルのサーバー上のファイル名
DestFile := '任意の保存ファイル名(後で特定できること)'+'.JPG';
//ファイルをアップロードする
IdFTP1.Put(SourceFile, DestFile);
//切断
IdFTP1.Disconnect;
```

VCLと処理は同様

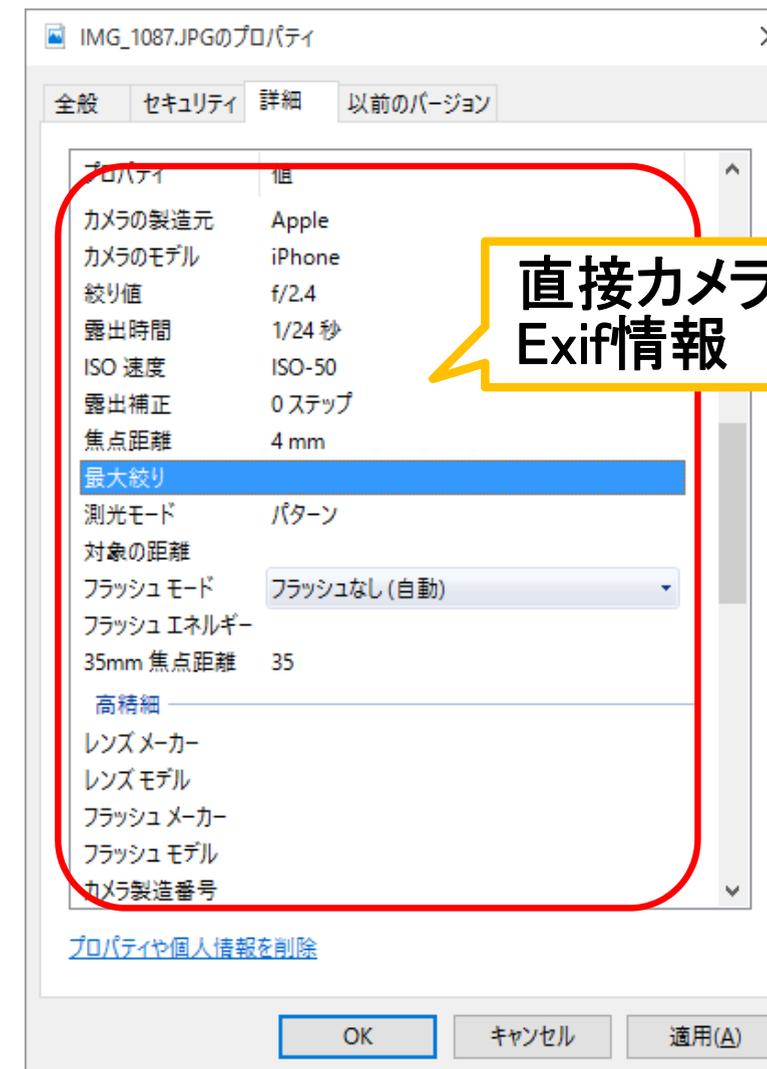
- 予め保存先PCでFTPサーバーの設定しておく
- IdFTPコンポーネントは配置しておく

## ④写真のExif情報

- 撮影した写真を確認するとjpgで保存をしているがプログラム上ではTBitmapで扱っているためExif情報が不足



Exif情報が存在しない

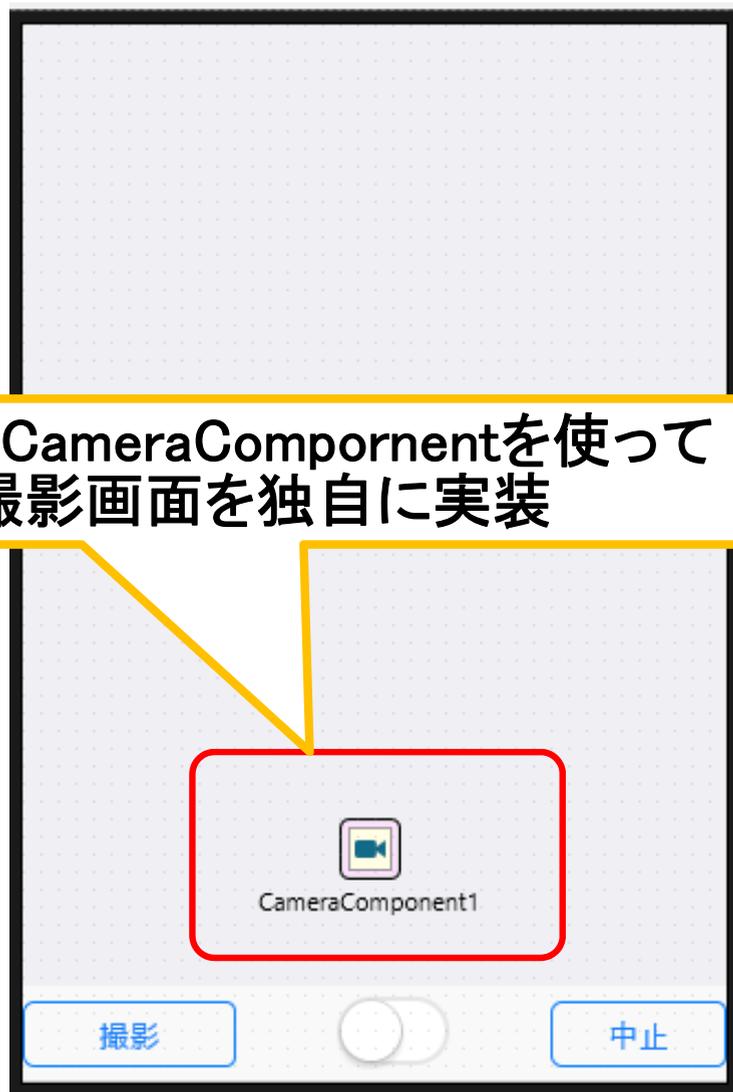


直接カメラで撮影した写真のExif情報

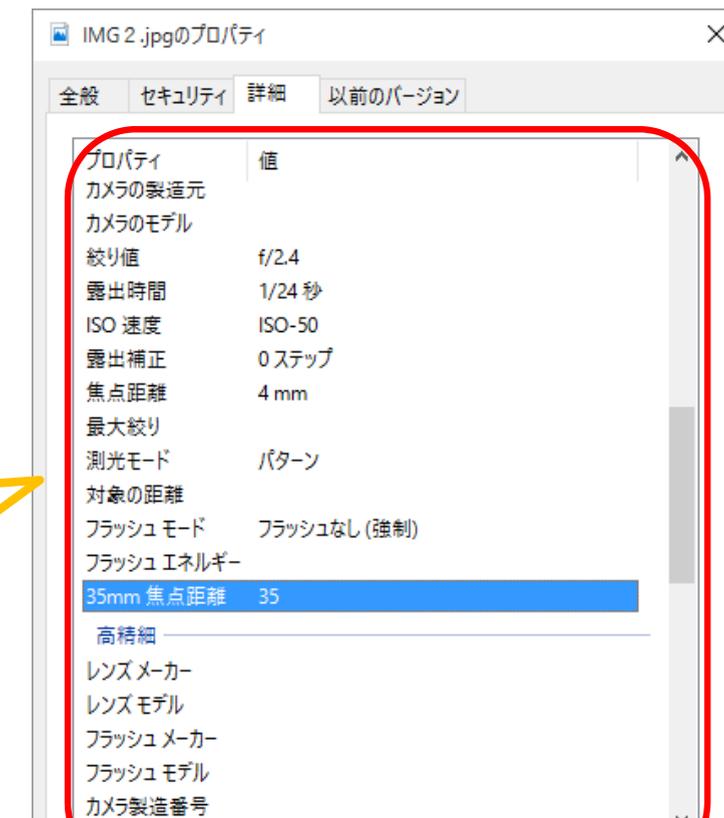
## ④写真のExif情報

- Actionからの撮影ではなく、撮影用のフォームを作成し、AssetsLibraryを利用して写真を保存する

TCameraComponentを使って  
撮影画面を独自に実装



Exif情報が保存できている  
(まだ不足情報もあるので  
さらに精度を上げることが課題)



AssetsLibraryを組み込んだ実装は下記のソースなどが参考になります。  
<https://github.com/ChristenBlom/SquareCamFmx/blob/master/uFMain.pas>

## ④写真のExif情報

```
procedure TFMain.Button1Click(Sender: TObject);
begin
  if (CameraComponent1.HasFlash and Switch1.IsChecked) then
    CameraComponent1.FlashMode := FMX.Media.TFlashMode.fmFlashOn;
  TakePicture; //撮影処理
end;
```

Flash等も必要であれば制御する(ここではTSwitchでON/OFF)

```
//TakePictureで呼ばれるStillImageCapturedあたりで保存処理等はアプリにあわせて変更
procedure TFMain.StillImageCaptured(const ImageDataSampleBuffer: CMSampleBufferRef;
const Error: NSError);

....中略
if ImageDataSampleBuffer <> nil then
begin
  JpegData :=
TAVCaptureStillImageOutput.OCClass.jpegStillImageNSDataRepresentation(ImageDataSampleBu
ffer);
  NSourceFile := NSSTR(保存ファイル名);
  JpegData.writeToFile(NSourceFile, True);
end;
```

## ⑤バーコード、QRコードの読み込み

- バーコード、QRコードの読み込みはTMS社のFreeコンポーネントであるTTMSFMXZBarReaderを利用

参考: TTMSFMXZBarReader

<http://www.tmssoftware.com/site/freetools.asp>

Androidの場合はTKRBarCodeSannerを追加で組み込めば同様に可能(Delphiのバージョンによって多少の修正は必要)

<http://www.file-upload.net/download-8601754/TKRBarCodeSanner.zip.html>

基本的な使い方例)

①TMSFMXZBarReaderコンポーネントをフォームに配置。

②バーコードリーダーの起動(Button、EditのOnClick等)

```
TMSFMXZBarReader1.Show; //これでカメラ起動
```

③TMSFMXZBarReaderのGetResultイベントで値をセット

```
//読み取れたらこのイベントが自動的に動く
```

```
procedure TForm1.TMSFMXZBarReader1GetResult(Sender: TObject; AResult: string);
begin
    Edit1.text := AResult;
end;
```

基本は  
これだけ

読み取りができたなら  
TMSFMXZBarReader  
では緑の枠表示



## ⑤ バーコード、QRコードの読み込み

連続使用時に読み込みが完了しない場合もあるため、コンポーネントを動的に生成したほうが安定する

動的な使用例)

```
procedure TForm1. Edit1Click(Sender: TObject);
begin
  try
    //TMSFMXZBarReaderを生成
    TMSFMXZBarReader := TTMSFMXZBarReader.Create(Self);
    //作成済みのイベントを設定
    TMSFMXZBarReader.OnGetResult := TMSFMXZBarReaderGetResult;
    //TMSFMXZBarReaderを実行
    TMSFMXZBarReader.Show;
  finally
    TMSFMXZBarReader.Free;
  end;
end;
```

## ⑤ バーコード、QRコードの読み込み

TTimerを使った繰り返し連続読み取り例)

```
procedure TForm1. Edit1Click(Sender: TObject);  
begin  
    Timer1.Enabled:=false;  
    TMSFMXZBarReader1.Show;  
end;
```

読取りカメラ起動

キャンセルすれば  
このイベント自体発生しないので終了

//読取り成功時の処理

```
procedure TForm1.TMSFMXZBarReader1GetResult(Sender: TObject; AResult: string);  
begin  
    if Edit1.text='' then  
        Edit1.text :=AResult;  
    else  
        Edit1.text :=Edit1.text +','+AResult;  
    //Timerを開始  
    Timer1.Enabled:=true;  
end;
```

読取りできた値を追加  
(この例はカンマ区切り)

再読取のタイマー起動

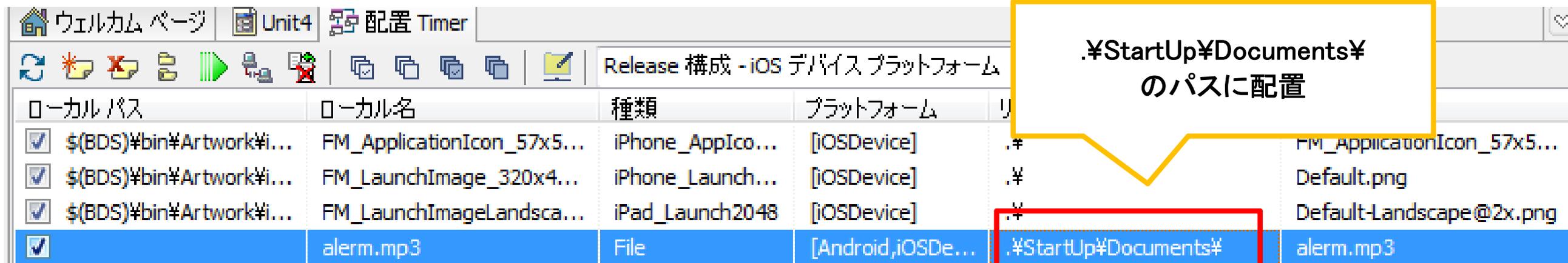
//Timerは読取り操作を再度呼出し

```
procedure TForm2. Timer1Timer(Sender: TObject);  
begin  
    Edit1Click(Sender);  
end;
```

## ⑤ バーコード、QRコードの読み込み

- iPod Touchはバイブレーション機能がないため、処理完了を音で通知

音源ファイルは配置マネージャで組み込む



読み取り完了を音で通知する例)

```
MediaPlayer1.FileName := TPath.Combine(TPath.GetDocumentsPath, 'alarm.mp3');

if MediaPlayer1.State=TMediaState.Playing then
begin
    MediaPlayer1.Stop;
    sleep(100);
end;
MediaPlayer1.CurrentTime := 0;
MediaPlayer1.Play;
```

## ⑥オフライン処理

- オフライン時はデバイスローカルにデータを一時保存

オフライン時にデータをローカル保存例 (ClientDataSetを前提にcdsファイルで一時的な保持)

```
procedure TForm1.Button1Click(Sender: TObject);
var
  sFolder: string;
begin
  //パス設定
  sFolder := TPath.GetDocumentsPath;

  //パスが存在しなければディレクトリを作成
  if not DirectoryExists(sFolder) then
    Mkdir(sFolder);

  //保存ファイル名の指定(pFileNameはグローバル変数)
  pFileName := TPath.Combine(sFolder, 'XXXXX.cds');

  //デバイスのファイルを保存
  ClientDataSet1.SaveToFile(pFileName, dfBinary);
end;
```

読み込み時はClientDataSetのFileNameに設定してOpenできる。  
(一時的な保持でなければcdsでは色々限界があるので、SQLite等を使う)

デバイス内の指定パスを使用

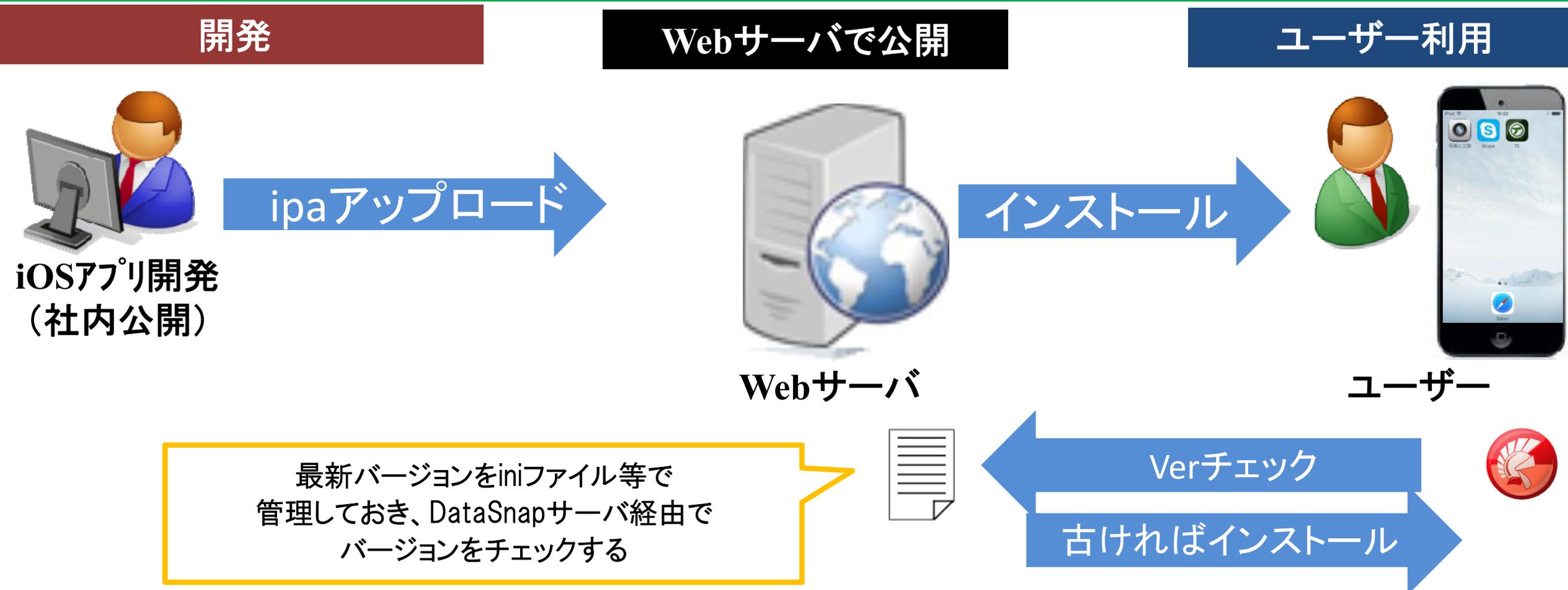


※XE5以前では永続パスでも  
アプリ更新時に刷新されて  
しまうので注意

## ⑦更新アプリの配布

- アプリのバージョンをチェックして更新する仕組み

社内公開でのiOSアプリ配布 (AppStoreは使用しない)



## ⑦更新アプリの配布

### 実現方法

- ・アプリ操作の何らかのタイミングで自らのバージョンの更新有無を確認  
(今回はメニュー(Tab)切替え時に設定)
- ・更新有無は使用アプリのバージョンとサーバーのOriginal版のバージョン情報比較  
(取得方法は次ページ)
- ・更新ありの場合は更新版ダウンロード(インストール)用WEBページを表示
  - ・ダウンロード対象はコンパイルして生成されるアプリ名.ipa(関連の圧縮ファイル)
  - ・アプリ名.ipaはWEBサーバーのルートフォルダにコピーしておく



インストール



Webサーバの配布・更新ページ  
(Safariのバージョン等でリンクが  
動作しない場合はhtmlで考慮)

## ⑦更新アプリの配布

- アプリでバージョンをチェックしてWebサーバの更新ページを起動

iOSアプリ側バージョンチェック例)

```
//自分のバージョン取得
AppKey := (NSSTR('CFBundleVersion') as ILocalObject).GetObjectID;
AppBundle := TNSBundle.Wrap(TNSBundle.OCClass.mainBundle);
BuildStr := TNSString.Wrap(AppBundle.infoDictionary.objectForKey(AppKey));
NowVer:=StrToIntDef(StringReplace(UTF8ToString(BuildStr.UTF8String), '.', '', [RFReplaceAll]),
                    9999999);

//DataSnapサーバーアプリの関数で同パスのiniファイルよりオリジナルバージョン情報取得
SQLConnection1.Open; //直接のバージョン取得出来ないのでiniファイルに記述した情報を読む
Temp:=TServerMethods1Client.Create(SQLConnection1.DBXConnection);
// DataSnapService.exeのGetAppliver関数に引数(アプリ名)を渡して取得する
Oriver:=StrToIntDef(StringReplace(Temp.GetAppliver(アプリ名), '.', '', [RFReplaceAll]), 0);

//判定結果で更新ページをブラウザで開く
URL:='http:XXXXXXXXXXXXXXXXXXXXX' //サーバーのアドレス/更新ページのhtmファイル';
if Oriver>NowVer then
OpenURL(URL, False); //更新ページを開く→Safari起動
```

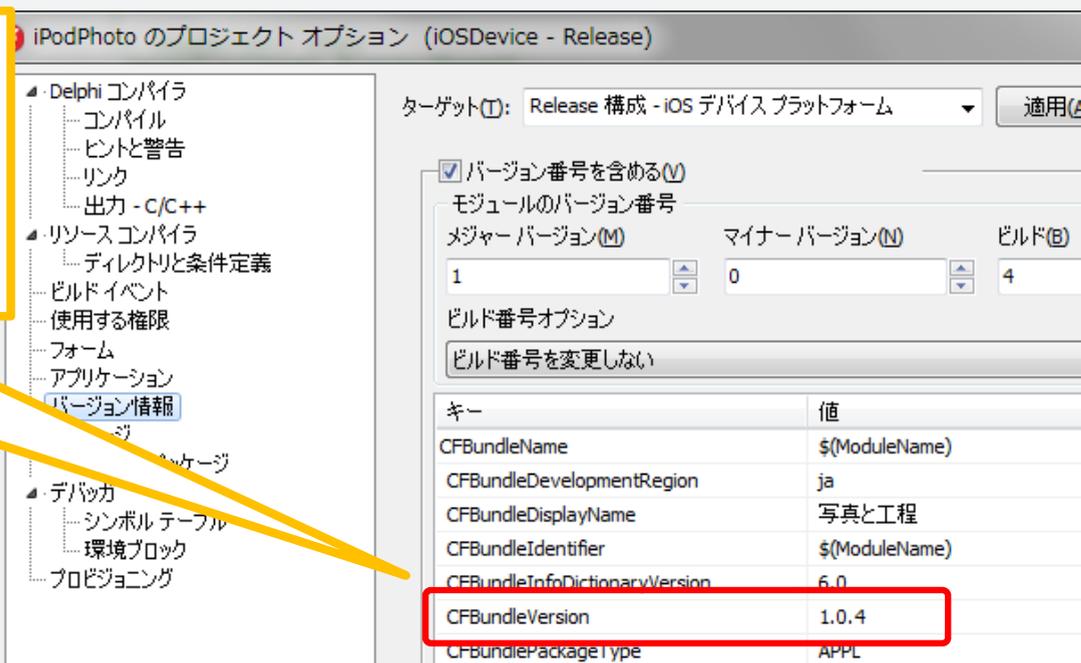
## ⑦更新アプリの配布

- アプリでバージョンをチェックしてWebサーバの更新ページを起動

DataSnapサーバ側のバージョン取得例)

```
function TServerMethods1.GetAppLIVER(AppName:string):string ;
var
  ini:TIniFile;
begin
  ini:=TIniFile.Create(ChangeFileExt(Application.ExeName, '.INI' ) );
  try //INIファイルから文字列を読み込む
  Result:=ini.ReadString('APPLIVER',AppName,'0.0.0.0');
  finally
    ini.Free;
  end;
end;
```

[プロジェクト|オプション]のバージョン  
情報で設定している  
CFBundleVersionをサーバ上の  
Iniファイルで管理しておく

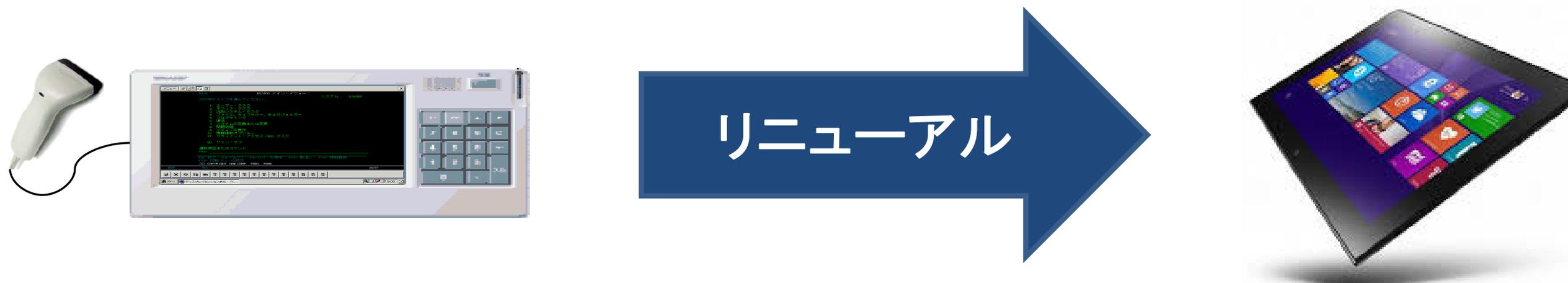


「iPod Touch による工場内のモバイル化」

# Delphi による iPod Touch以外のモバイル化対応

# iPod Touch以外のモバイル機器導入

- WindowsCE (CUIアプリ) + バーコード機器などの老朽化したシステムをWindowsタブレット向けにビジュアル化して刷新



- Windows8.1Proタブレット型 (ThinkPad10 Lenovo) を採用
- アプリケーションは他のシステム同様に **Delphi** で開発



工場内容断機に配置され、NCデータを加工ラインに送信する中継機として機能する。  
+ 使用鋼材チェック機能を追加

# iPod Touch以外のモバイル機器導入

- 生産指示書を見ながら目視チェックしていた為、チェックミスが排除できなかった課題を解決！

## ①作業指示書

- QRコードをスキャン

## ②鋼材

- QRコードをスキャン

## ③IBM i (DB2/400)

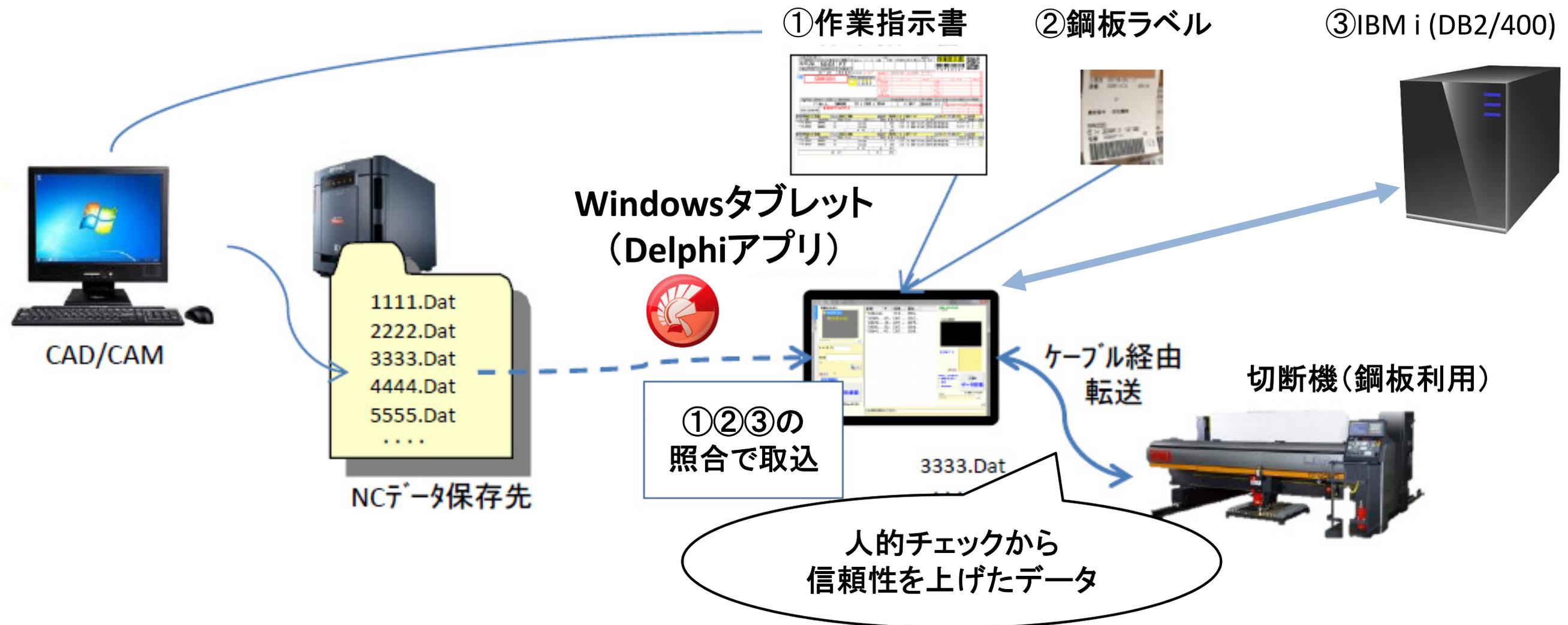
- ①②をデータ照合 → 使用鋼材のチェック



タブレットの機能を活かして  
系統的に信頼性アップ

# Windowsタブレットを使った業務機能(一例)

- Windowsタブレットでシステムのなチェック



# Windowsタブレットを使った業務機能(一例)

- Windowsタブレットでシステムのチェック

名前	サイズ	更新日時
SUBmoto		2015/01/13 10:16
O2089.dat	43.7 KB	2012/02/01 21:06
O3258.dat	15.9 KB	2014/01/16 9:13
O3266.dat	29.3 KB	2014/02/04 14:26
O5641.dat	97.7 KB	2014/08/19 11:16

①作業指示書から  
QRコードスキャン

②鋼板ラベルから  
QRコード、バーコード  
スキャン

③IBM i でデータ照合

鋼板切断の為  
切断機へNCデータ転送

# Delphiを活用したモバイル対応業務の拡大



- 今後のモバイル対応予定

紙やハンディーターミナル等で行っている業務を中心にモバイル対応化を拡大していく

- 鋼材準備業務
- オペレータ日報業務
- 出荷業務

「iPod Touch による工場内のモバイル化」

# まとめ

# まとめ

- Delphiを使ったマルチデバイス開発・モバイル化

社内基幹システム (IBM i + WindowsPC) をはじめ、新しく導入した様々なモバイル機器 (iPod Touch、Windowsタブレット) の全てにDelphiのマルチデバイス開発で対応できている

- 自社用の細かいニーズに対応したモバイル化をスタートできた
- WindowsPC向けに開発してきたDelphiスキルを活かした (モバイル化は実質3人月で完成)
- 紙ベースやハンディ等の専用機器業務は、Delphiによるモバイルシステム化で効率アップできる可能性がある

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

# 31ST EMBARCADERO DEVELOPER CAMP

第31回 エンバカデロ・デベロッパーキャンプ

Thank you!

embarcadero®

