最優秀賞

iPod Touchの業務利用開発と検証
―多目的端末としてどこまでできるか?

石井 裕昭 様
豊鋼材工業株式会社
製造総括部 部長

豊鋼材工業株式会社
http：//www.yutaka-steel.co.jp/

鋼板加工のトータルコーディネーターとして、レベラー・スリット・ 溶断・開先・穴明け・プレス・ベンディング・溶接・マシニング・ショット・プライマー設備を有機的に活用し、幅広い産業分野のお客様に商品・サービスを提供している。九州・沖縄エリアでは業界トップシェア。伊藤忠丸紅鉄鋼・新日鐵住金系の会社である。

開発の経緯

豊鋼材工業は2005年より、新生産管理システムとして工場内のハンディターミナルやバーコードプリンタなどのハードウェアの整備と、Delphi/400導入による工程管理、トレーサビリティの強化、業務効率化、見える化などを推進してきた。

その一方、工場の現場サイドから情報を利用・参照したり、簡単に情報を入力して作業効率を高めるための端末に関しては、使用環境や適正な端末形態、開発ツール、価格等の制約より整備が進んでいなかった。

このような中で、写真を扱う業務で写真データの一貫管理のニーズが発生し、さらに他の業務用途でも機能拡張のニーズがあることから、Delphi/400の最新バージョン化と開発環境の整備により、従来の業務プロセスを大きく改善できるのではないかと考えた。

なお、従来から使用している生産管理システムは「Delphi/400 Ver2007」で継続開発しており、開発環境が「Delphi/400 XE5」(当時の最新バージョン）に変わる場合、Unicode対応のための既存ソースの改修や、サードパーティのコンポーネントの対応などでコンバージョンに要する工数が膨大になると判断し、既存の開発ライセンスを継続し、並行運用にすることとした。今後は、用途、利用端末により開発・運用環境を使い分けていく予定である。

工場で使用する端末については、業務の内容や用途によって適切なOS、サイズ等は異なるものの、まずは写真撮影用途を意識してiPod Touchでの機能検証から事前スタディを開始した。

iOSのiPod Touchを端末として選定した理由は、端末メーカー、モデルによる挙動の違いを意識しなくてよく、端末を随時追加する際に、ほぼ同じモデルを提供できるのと、電話機能がなく月々の通話料が発生しないからであった。

iPod Touchで開発・検証した機能

事前スタディで、iPod Touchで実現したい機能の実装可否と、業務用途として実際に耐え得るかの確認を行った。期待している機能は、以下の通りである。

① Delphi/400で開発する新たな用途、およびGUI化済みの既存業務機能
②従来よりハンディターミナルの5250エミュレータ用に作られ、運用されている機能（5250エミュレータは株式会社ヒットの「WaveLink TE」を利用）
③通話、メッセージなどのコミュニケーション機能（Skype利用）

①で新たに開発するのは、工程写真・鋼材写真の撮影・保管管理機能、作業予定・順番情報公開機能、生産完了実績入力機能、作業時間（日報）報告機能などである。一部はすでにネイティブアプリとして開発・運用済みであるが、今後も継続的にニーズ掘り起こし、開発を進め

表1　iPod Touchでの開発･検証内容

分類	項目	内容、備考
新規Delphi開発機能	・工程、鋼材ステンシル管理用カメラ ・作業予定ロット、順番公開 ・生産実績入力 ・稼動日報 ・出荷準備(梱包)済現品情報、など	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可能とする。時間を要していた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。
既存ホスト業務	・生産実績入力(完成メニュー) ・出荷チェック ・現品票作成 ・ロケーション管理 ・棚卸し　など	従来ハンディーターミナル用に作られている各種業務メニューをWaveLinkというアプリで表示し、既存の業務を行う。 バーコード読取専用機ではないため、読取のレスポンス等に応じて適用業務は選定する。 既存機器の約1/3以下のコストで機能を実現したい。
コミュニケーション機能	・Skype(無料通話)による iPod等の登録ユーザー間通話 メッセージ機能など	無料アプリSkypeにより無線LANエリア内での Skype同士の通話を可能化。 またはメッセージ、ビデオ通話など




表2　iPod Touchでの写真撮影機能開発の目的

分類	目的	従来の課題と経緯
品質保証体制の強化(トレーサビリティ)	▪ 品質証明としての写真情報を管理維持する基盤づくり ▪ 鋼材の保有･使用履歴の客観的証拠として原則全鋼板のステンシル写真を保管 → (鋼材制限の緩和=滞留材削減) ▪ 生産履歴･鋼材識別情報と写真の関連性の維持とサーバー内一元管理	要求時のみ個々のカメラで撮影された工程写真データが工場スタッフ、営業担当のPCの任意のフォルダーに保存され、フォルダー、ファイル名等で整理。保管ルール取決め無し。 品質証明書類として鋼材のステンシル写真要求増。使用時にステンシル部分無いと使用制限され滞留在庫の一因。またステンシル有無の管理も必要。 事後に特定の生産履歴、又は特定の鋼材について要求されても、有無の確認、検索は困難。
関連作業の効率化	▪ 1台／機種の保有で機種間のカメラ探し(オペレータ)をなくす ▪ 工場スタッフによる撮影をオペレータ作業に完全移行し、撮影待機(スタッフ、オペレータ共)、連絡のロス時間等をなくす ▪ 工場スタッフのカメラ(メモリカード)回収、データ仕分け作業をなくす ▪ 営業担当者の提出書類と写真内容の照合作業を半自動化	限られた台数のデジカメを共有使用で、手元に無い場合は都度探す必要があった。 工場スタッフが撮影時は対象材処理の都度、連絡を取る、予定時間を確認など必要。双方の待機時間の無駄。 写真データ回収を意識する必要があり、回収データを客先毎に仕分けには黒板の工事名を全て確認必要。 板番、ロット番号等との関連も写真の画像を一枚毎に開いて確認するしかなく、手間を要する。

る。

②は、バーコード、QR コードの読み取りを iPod Touch のカメラ機能で行うことによる、操作性、レスポンス、読み取り可否等の確認である。実用に耐え得るかがポイントとなる。これが有効であれば、既存の高価なハンディターミナルの代替機として低コストでの導入が可能となる。

③は、WiFi 環境の工場内や事務所間で無料通話が行えれば利便性が高く、構内電話が不要になる可能性もある。

以上のポイントを【表 1】にまとめる。

Delphi/400 XE5による写真撮影機能の開発

従来の課題とニーズ

構造物等で使用される鋼材は、使用箇所などの条件を考慮して、さまざまな材質で注文を受ける。鋼材の材質は、外見では判断できないため、工場における在庫管理、識別管理と、確実な使用実績の報告により、使用鋼材の証明書を提出する。通常は、鉄鋼メーカーが発行するミルシート（鋼材検査証明書）から該当分を提出することが多い。最近は、一時期の鋼材偽装問題も影響してか、写真での証明提出を求められるケースが多くなっている。

品質保証体制強化を含めた開発の目的を【表 2】にまとめる。また、この場合、写真撮影から提出までのプロセスで、次のような問題があり、業務効率化の妨げとなっていた。

・デジタルカメラの台数制約による共有化→カメラ探し時間
・スタッフによる撮影データの回収
・スタッフが撮影する場合の、タイミングの連絡、待ち時間の発生など
・回収した撮影データ（複数のお客様分が混在）から特定分のみを抽出する手間（個々の画像確認し選別）
・撮影データと生産実績データ（基幹情報）の関連がなく、事後の検索が困難
・お客様からの要求がなく撮影していない鋼材は、事後の提出が不可能

上記の問題を解決するために、iPod 端末は 1 つの作業機器に対して 1 台を割り当て、写真撮影に関するアプリは、次のような仕様で開発することにした。

・撮影した写真データは、ネットワーク上のサーバーに自動的に保存する。
・写真データのファイル名は、検索の容易性を考慮し、キー情報を含むユニークな値とする。
・キーとなる情報は、鋼材の識別番号もしくは作業ロット番号とし、撮影前に QR コードまたはバーコードのいずれかで読み取る。
・IBM i のデータベースに、写真のファイル名、保存先フォルダー名、撮影日時、鋼材番号、作業ロット番号などを持つ写真データ抽出用のレコードを、写真 1 枚ごとに書き込む。
・保存された写真データを特定し、抽出・整理する機能は、IBM i の生産実績データベース、鋼材データベース等と関連させて既存の生産管理システムに追加する（操作は既存の PC）。

以上の処理フローを【図 1】に、iPod Touch での画面操作の流れを【写真 1】～【写真 2】に示す。

これらの仕様を実現するための iPod アプリ開発は、ミガロ．のサポートを受けながら主に次のポイントを押さえて進めた。

システム機器の構成は【図 2】の通りである。

QRコード、バーコード読み取り

Delphi/400 XE5 には、QR コード、バーコード読み取り用のコンポーネントは含まれていないため、当社では TMS 社のフリーコンポーネント「TTMSFMXZBarReader」を登録して開発を進めた。

読み取り処理の実装自体は比較的容易である。使用したコンポーネントでのコード記述例を【ソース 1】に示す。

このコードからわかるように読み取り画面（カメラ）は Show メソッドで起動し、読み取りが成功すると自動的にコンポーネントの GetResult イベントが動く。GetResult には読み取られた文字が引数で渡されるので、この文字列を利用するようにコード記述するだけである。ただし、一次元バーコード読み取りの場合、桁数にもよるが、ある程度の大きさが必要となり、当社では鋼材ラベルのバーコードは約 10％ほど幅を拡大した。QR コードでは一次元バーコードよりレスポンスはかなり改善されるが、こちらもサイズによっては読み取り不能となる。

有料アプリなどではストレスなく読める場合もあるため、改良の可能性はあると思われるが、現状では手がかりがない。ハンディターミナル相当の業務を iPod 等で代替するには、特に QR コード読み取りの操作性とレスポンスが最大のポイントであり、頻度にもよるが一定の読み取りレベルに到達できれば、その汎用性からもさまざまな業務で置き換えが進む可能性がある。

IBM iの情報抽出と表示

DataSnap サーバー経由での取得となるが、抽出条件（SQL 文）は、ClientDataSet の CommandText に記述する。TQuery ではオブジェクトインスペクターの SQL プロパティで Lines への記述であったので長文でも問題ないが、CommandText プロパティは複数行表示不能のため、プログラムで行を分けてコードを記述した方がよい。行分け時は、＋でつなぐテキスト間のスペースを忘れないようにする。なお、抽出したデータを表示する DBGrid のようなコンポーネントがないので、各項目の意味がわかるように Label にタイトルとレコードの値を組み合わせて表示した。

また、方法は省略するが、ListView コンポーネントで、LiveBinding デザイナーによって ClientDataSet と関連づけて表示することもできる。

オブジェクトインスペクタのプロパティでの SQL 記述の比較を【図 3】に示す。　また、コードでの記述方法の比較を【ソース 2】に示す。

写真データのサーバー保存

撮影された iPod の写真データを、FTP プロトコルを用い、あらかじめ準備されたサーバー上のフォルダに jpg 形

図1　写真撮影、保管、利用のフロー

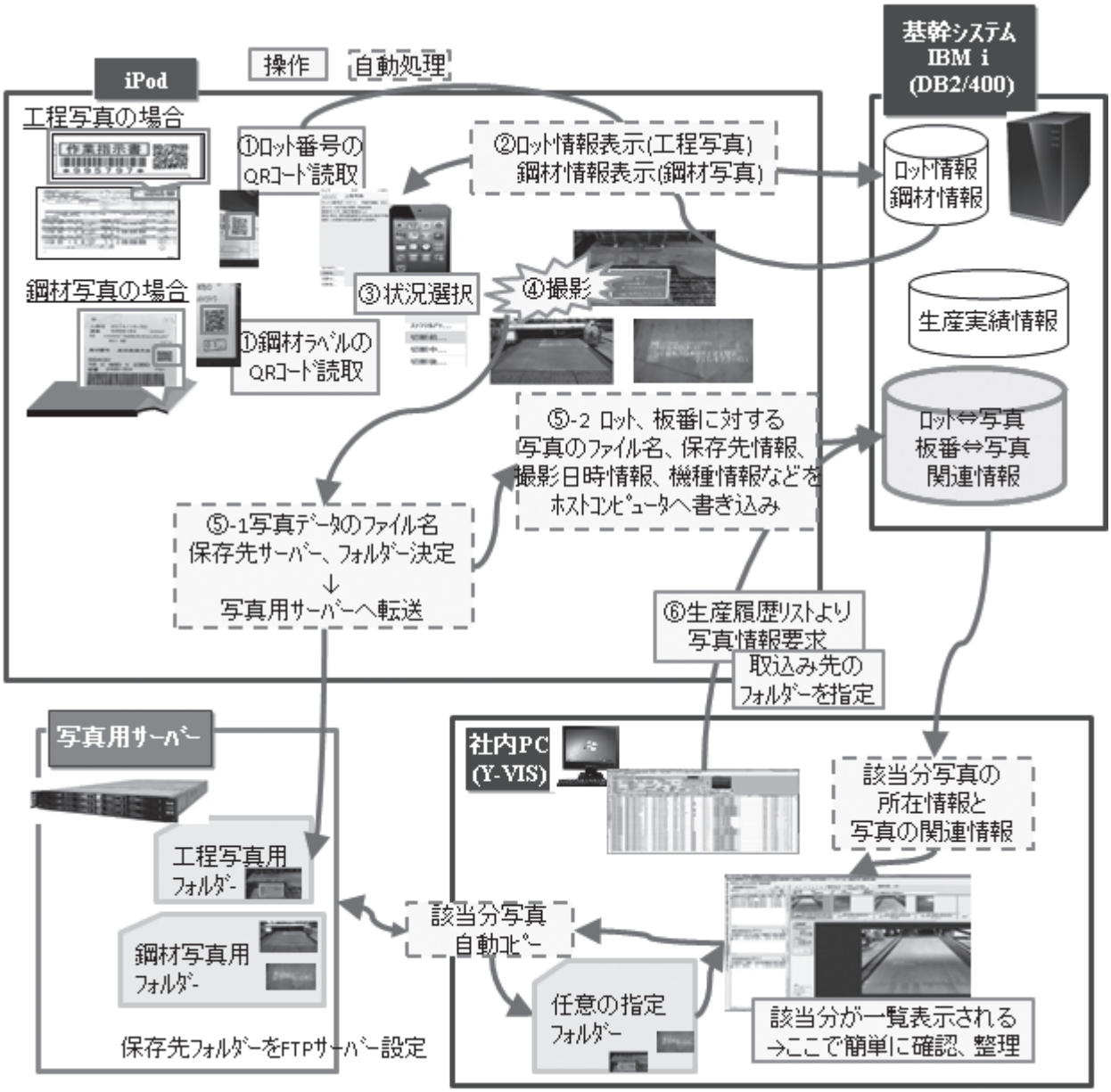


写真1　カメラ起動までの操作画面

式で保存している。具体的には、IdFTPコンポーネントを配置して保存先のサーバー（hostプロパティでIPアドレス指定）にConnectメソッドで接続し、指定のフォルダーへChangeDirメソッドで移動の上、Putメソッドでファイルを保存できる。なお、移動元ファイルは、Camera撮影のActionで撮影後に実行されるOnDidFinishTakingイベントで得られるBitmapのイメージをファイルに割り付けたものである。

IBM iのデータベースへの情報書き込み

これもClientDataSetのCommandTextに記述するSQL文（Insert、Updateなど）で、ClientDataSet.Executeで実行する。

写真アルバムへの保存はオプションで

当初、トラブルを想定して端末の写真アルバムにも撮影データを保存していた。しかし、コードの不備でメモリが消費されアプリのクラッシュが頻発したため、メモリ解放するようにコードを改修するとともに、アルバムへの保存は初期設定でのオプションとした。【ソース3】にこの部分の処理を示す。

保存された写真データを特定し、抽出･整理する機能

これは、従来の生産管理システム（Y-VIS）への機能追加で対応した。生産実績を、特定の条件で絞り込む機能は従来から使われているが、新たに追加された写真の保存実績のデータベースとのリレーションにより、該当する写真ファイルのフルパスを取得し、任意のフォルダにファイルコピーできる。また、抽出されたファイルをImageコンポーネントにそれぞれ割り付けて表示しているが、この部分では、特に目新しい機能は使用していない。なお、画面上で選択した写真はNewtone社製品のImageKitを使い、任意の拡大、パンウインドウ表示などの機能を実装した。操作画面のイメージを【図4】～【図6】に示す。

iOSアプリ使用のポイント、注意点

iOSでIBM iに接続し、機能を利用するにはDataSnapサーバーのDataSnapアプリを介する必要があり、DataSnapアプリを常時起動させておく必要がある。

当社ではこれを、フォームアプリケーションではなくサービスアプリケーションとしているが、何らかの原因でフリーズまたは終了しているケースがあったため、タスクスケジューラで定期的にbatファイルによる再起動等を行っている。

サービスの再起動は、下記の2行で実行される。

```
net stop "ServerContainer1"
net start "ServerContainer1"
```

また、複数の端末がDataSnap経由でアクセスする場合は、セッションの輻輳により処理が停止することがあるので、DataSnapサーバーアプリ側のTSQLConnectionのパラメータで、マルチセッションの使用可否を決める［Decimal Separator］をYに設定することが重要であり、またアクセス前後での接続､切断を漏れなく行う必要がある。【ソース4】にDataSnapサーバーとの接続、切断の処理方法を記述した。

アプリ更新の通知と作業による確実な更新処理の実装も、端末を配布した後の運用では重要である。

当社では、アプリ内の選択機能の切り替え時に、サーバーで管理する最新版バージョンと自アプリのバージョン情報を比較し、バージョンが上がっている場合は更新用のWebページが開き、更新を促す。更新が試されないと毎回、リンクページが開くことになる【図7】。なお、サーバーでの最新版確認は、DataSnapアプリに実装した関数で取得している。

開発用のiPod端末で検証されたアプリはアプリケーションサーバーに保存し、各iPod端末のブラウザのリンクよりインストールあるいは更新するが、iOS 7.1以降は社内配布用Webサーバーの SSL証明書をあらかじめ各端末に導入しておく必要がある。この手続きはMac、サーバー等で行うが、詳細な手順は、ネット情報またはミガロ．のサポートなどに確認が必要である。また開発ライセンス、証明書等には期限があるので定期的な更新を要する。

開発ライセンス登録～インストール、配布に必要な手続き、設定と各構成機器の関連を【図8】にまとめるが、高いセキュリティと厳密なライセンス管理のために、かなり煩雑な手順を踏む必要がある。

システム立上げ時に発生した問題点

システム立上げ時に発生した主なトラブルについて、【表3】にまとめた。

QRコードと1次元バーコードの併用への対応

ラベルは、既存のバーコードリーダーに対応するものや、QRコードを表記しないものが混在しているため、バーコード読み取り時の処理には工夫が必要である。通常、QRコードは1次元バーコードよりも多くの文字情報を保有できるので、さまざまな運用を想定して記載する情報を決定している。

同一の帳票に印字されたQRコードと1次元バーコードを同じ用途で使用する場合、1次元バーコードの情報は、QRコードでは先頭に配置する方がよい。実際にQRコードと1次元バーコードを併用している例を【図9】に示す。

iOSで5250エミュレータ使用

当社では従来、棚卸しをはじめとしてさまざまな用途でハンディターミナルを利用してきたが、機能追加、改修時の容易さ、リアルタイム性等の理由から、ハンディターミナルに5250エミュレータを搭載し、IBM iに直結する方式で運用してきた。

今回の検討にあたっても、iPodに5250エミュレータを搭載しIBM i上の従来資産を活用して業務が行えるのであれば、積極的に取り組みたいと考えていた。

調査の結果、iPodで利用可能な5250エミュレータとして株式会社ヒットの「WaveLink TE」があることがわかり、

写真2　写真撮影から保存までの画面

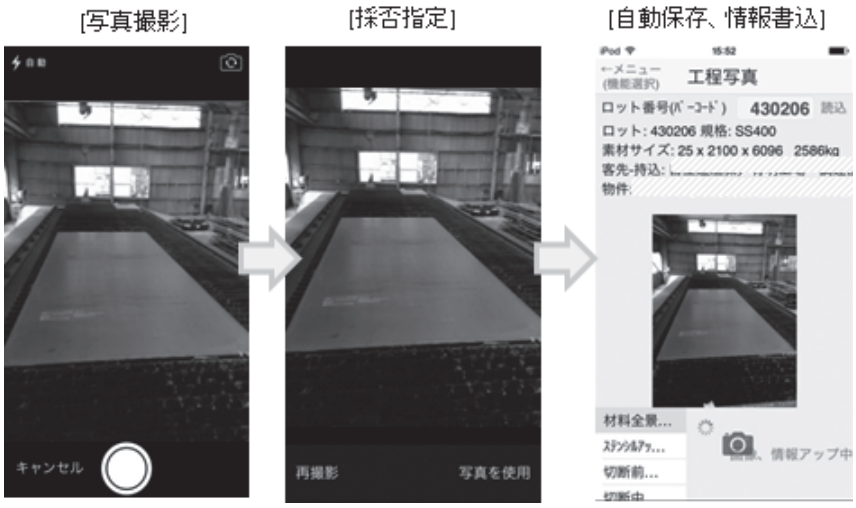
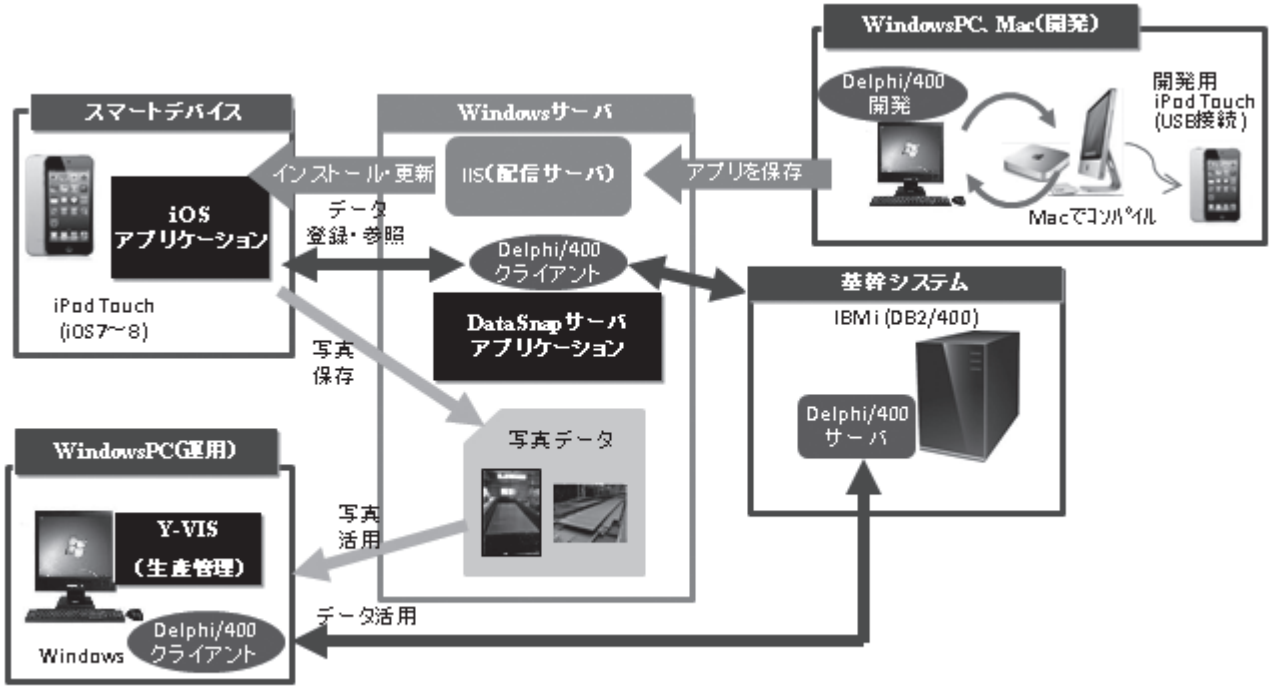


図2　iOSデバイスに関連するシステム機器構成



ソフトキーを数字キー主体の現場仕様にカスタマイズできれば、業務で利用可能と判断した。

ただし、ハンディターミナル用の既存フォームと処理コードは1次元バーコード用に作られているため、iPodでQRコードを読み取ると、保有する情報量の違いにより、入力フィールドの桁数を超えた文字が、次のフィールドまたは同じフィールドに書き込まれてしまう。

「WaveLink TE」ではこのような場合、オーバーフロー分を切り捨てる、または分割して次のフィールドに書き込む、といった設定をオプションで選択できるので、柔軟な対応が可能であった。【図10】

また、ハンディターミナルの場合、QRコード等のスキャンはボタン操作であり、照射部分がレーザー光などで確認できるが、「WaveLink TE」では、画面のソフトキーでカメラを起動し（【図11】）、表示部分で対象を確認、ピント合わせで読み取り、という手順を踏むため、最初は慣れを要した。iPodは、ハンディターミナルと比べて、携帯性、バッテリーの継続時間、コストなどで有利なため、各用途での運用可否を見極めていく予定である。

5250エミュレータの業務機能のGUI化

従来の5250エミュレータ画面と、同じ機能をDelphi/400 XE5で開発したアプリ操作画面との比較を【図12】に示した。この機能は、製品の生産後に残った鋼材の寸法、形状を登録する処理であるが、Delphi/400 XE5のGUIアプリの方が、入力フィールドの間違い、寸法位置の勘違いなどのミスが起こりにくく、直感的な操作が可能である。この例のように、従来の5250画面操作で処理しにくい機能については、Delphi/400 XE5で作り直しを検討したい。

コミュニケーション機能（Skype）検証

今回の目的は業務アプリの開発であり、通話は主目的でないため、毎月の通話料発生を避けるためにiPod Touchを選択したが、無料のアプリ「Skype for iPhone」を使って通話を含めたコミュニケーションが行えないか検証した。

検証の結果は下記の通りである。

◎通話機能

Skypeによる工場内での通話は、着信の認識、通話音量、音声出力位置などの点から、現状のままでは実用的でないことがわかった。具体的には、呼出音が工場内ではほとんど聞こえず、バイブレーションの機能もないため、着信に気づかないことが多い。ただし着信履歴は画面上に残るため、事後の折り返し電話は可能である。なお、iPodのマイク、スピーカの位置は通常のスマートフォンと異なるため通話時は注意が必要で、上下、表裏それぞれ逆の方がよい（【図13】）。またマイク付きイヤホン、またはヘッドセットの利用が推奨されている。

◎メッセージ機能

緊急を要さない場合については、メッセージ機能は有効である。使い方はスマートフォンなどのショートメールやLINE等と同様で、簡単である（【図14】）。写真もカメラ連動でその都度送付できるので、情報を視覚的に伝えたい場合に便利であり、さらに必要であればビデオ通話も可能なので、状況をより具体的に伝えることができる。

今後の狙い

今回、Delphi/400 XE5へのバージョンアップと新しい開発環境の整備にあたって、予算申請の際に課題として挙げた、日報（業務時間報告）、鋼材の置場管理、作業予定開示などの機能については、参照させる情報量に応じて、iPodまたはタブレット端末（iPad、Windows Tabletなど）を視野に入れて開発し、業務の効率化、新しい価値の創造につなげていきたいと考えている。

M

ソース1　バーコード読み取りりのコード記述例

```
//TMS社のFreeコンポーネントであるTTMSFMXZBarReaderを使用

// TMS社FreeTool　一番下のページ
 //http://www.tmssoftware.com/site/freetools.asp

①TMSFMXZBarReaderコンポーネントをフォームに配置します。

②バーコードリーダーの起動（Editの Onclickイベント等）
  TMSFMXZBarReader1.Show;

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;


2回目のバーコード読込が完了しない現象に対して、動的にTTMSFMXZBarReaderを生成するよう変更した。

//グローバル変数としてTMSFMXZBarReader を宣言
 private
  { private宣言 }
  TMSFMXZBarReader :TTMSFMXZBarReader;

//バーコード撮影時にTMSFMXZBarReader を生成
procedure TForm2. Edit1Click(Sender: TObject);
begin

//TMSFMXZBarReader が存在すれば解放
 if Assigned(TMSFMXZBarReader1) then
 begin
  TMSFMXZBarReader1.Free;
 end;

 //TMSFMXZBarReaderを生成
 TMSFMXZBarReader1 := TTMSFMXZBarReader.Create(Self);
 //作成済みのイベントを設定
 TMSFMXZBarReader1.OnGetResult := TMSFMXZBarReader1GetResult;
 //TMSFMXZBarReaderを実行
 TMSFMXZBarReader1.Show;
end;
```

図3　SQLのコードの静的設定の比較

ソース2　SQLのコード記述の違い

ClientDataSetのCommandTextの場合

```
TestCDS.CommandText:=
 'SELECT  L.LO2001,L.LO2027,L.LO2017,L.LO2018,H.JU1012,TRIM(K.KAK06K)||''~''||TRIM(H.JU1017) AS KAK,Z.SOZ007,Z.SOZ008,J.SJU005'
 +' FROM WRKLIB/LOTF2P L'
 +' LEFT JOIN DTALIB/JVUHADP H ON L.LO2005>0 AND H.JU1099=LEFT(L.LO2004,1) AND H.JU1003=CAST(RIGHT(L.LO2004,5) AS NUMERIC(5)) '
 +' LEFT JOIN DTALIB/JVUMEIP M ON L.LO2005>0 AND M.JU2002=LEFT(L.LO2004,1) AND M.JU2003=CAST(RIGHT(L.LO2004,5) AS NUMERIC(5)) AND M.JU2004=L.LO2005 '
 +' LEFT JOIN MSTLIB/KYAKUP K ON K.KAK004=H.JU1013 AND K.KAK005=H.JU1016 '
 +' LEFT JOIN WRKLIB/SOZSUP J ON J.SJU001=L.LO2001 LEFT JOIN MSTLIB/SOZUKEP Z ON Z.SOZ016>0 AND Z.SOZ092=J.SJU003 '
 +' WHERE L.LO2001='+Trim(LotNumEdit.text)+' AND L.LO2005>0 ';
```

Queryのの場合

```
 TestQuery.SQL.add('SELECT  L.LO2001,L.LO2027,L.LO2017,L.LO2018,H.JU1012,TRIM(K.KAK06K)||''~''||TRIM(H.JU1017) AS KAK,Z.SOZ007,Z.SOZ008,J.SJU005');
 TestQuery.SQL.add('FROM WRKLIB/LOTF2P L');
 TestQuery.SQL.add('LEFT JOIN DTALIB/JVUHADP H ON L.LO2005>0 AND H.JU1099=LEFT(L.LO2004,1) AND H.JU1003=CAST(RIGHT(L.LO2004,5) AS NUMERIC(5)) ');
 TestQuery.SQL.add('LEFT JOIN DTALIB/JVUMEIP M ON L.LO2005>0 AND M.JU2002=LEFT(L.LO2004,1) AND M.JU2003=CAST(RIGHT(L.LO2004,5) AS NUMERIC(5)) AND M.JU2004=L.LO2005 ');
 TestQuery.SQL.add('LEFT JOIN MSTLIB/KYAKUP K ON K.KAK004=H.JU1013 AND K.KAK005=H.JU1016 ');
 TestQuery.SQL.add('LEFT JOIN WRKLIB/SOZSUP J ON J.SJU001=L.LO2001 LEFT JOIN MSTLIB/SOZUKEP Z ON Z.SOZ016>0 AND Z.SOZ092=J.SJU003 ');
 TestQuery.SQL.add('WHERE L.LO2001=' + Trim(LotNumEdit.text) + ' AND L.LO2005>0 ');
```

ソース3　写真アルバム保存時のコード

```
if PhotoAlbumYNSwitch.IsCheckedthen
begin
CRImage:=BitmapToUIImage(image); //CRImage:UIImage→varで宣言
UIImageWriteToSavedPhotosAlbum(
  (CRImage asILocalObject).GetObjectID,nil,nil,nil);
 CRImage.release;//これが無いとすぐにクラッシュ
end;
```

図4　抽出した写真の一覧表示画面

図5　抽出した写真の要否選択

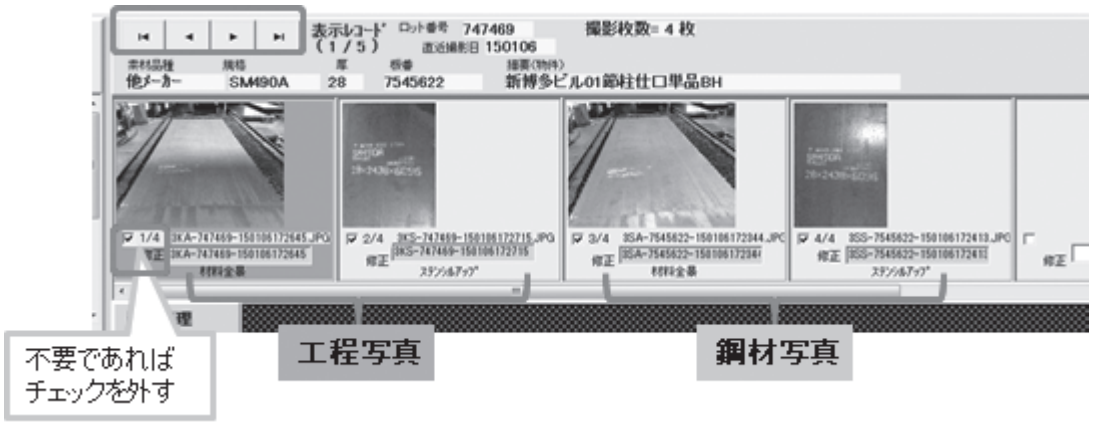


図6　写真のファイル名の自動整理機能

ソース4　DataSnapサーバー接続・切断のコード

```
 functionTForm2.SvrConnectChk:boolean;
begin
  try
   SQLConnection1.Connected := False;
   SQLConnection1.Connected := True;
   result:=true;
 except
   showmessage('スナップサーバー接続できません、連絡のこと');
   result:=false;
   abort;
 end;
end;

function TForm2.SvrDisConnect:boolean;
begin
 try
   SQLConnection1.Connected := False;
   result:=true;
 except
   result:=false;
   abort;
 end;
end;
```

図7　アプリ更新のリンクページ

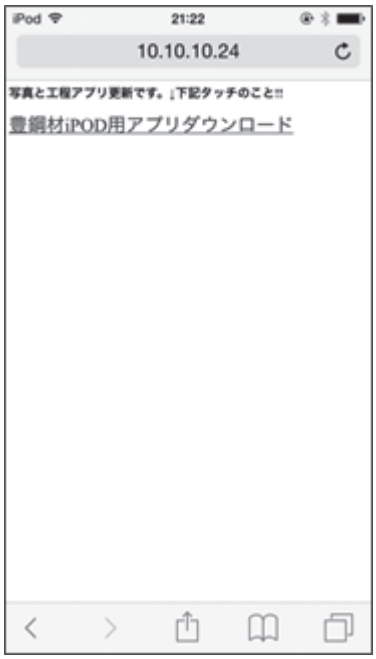


図8　開発ライセンス登録～配布に必要な設定等のイメージ

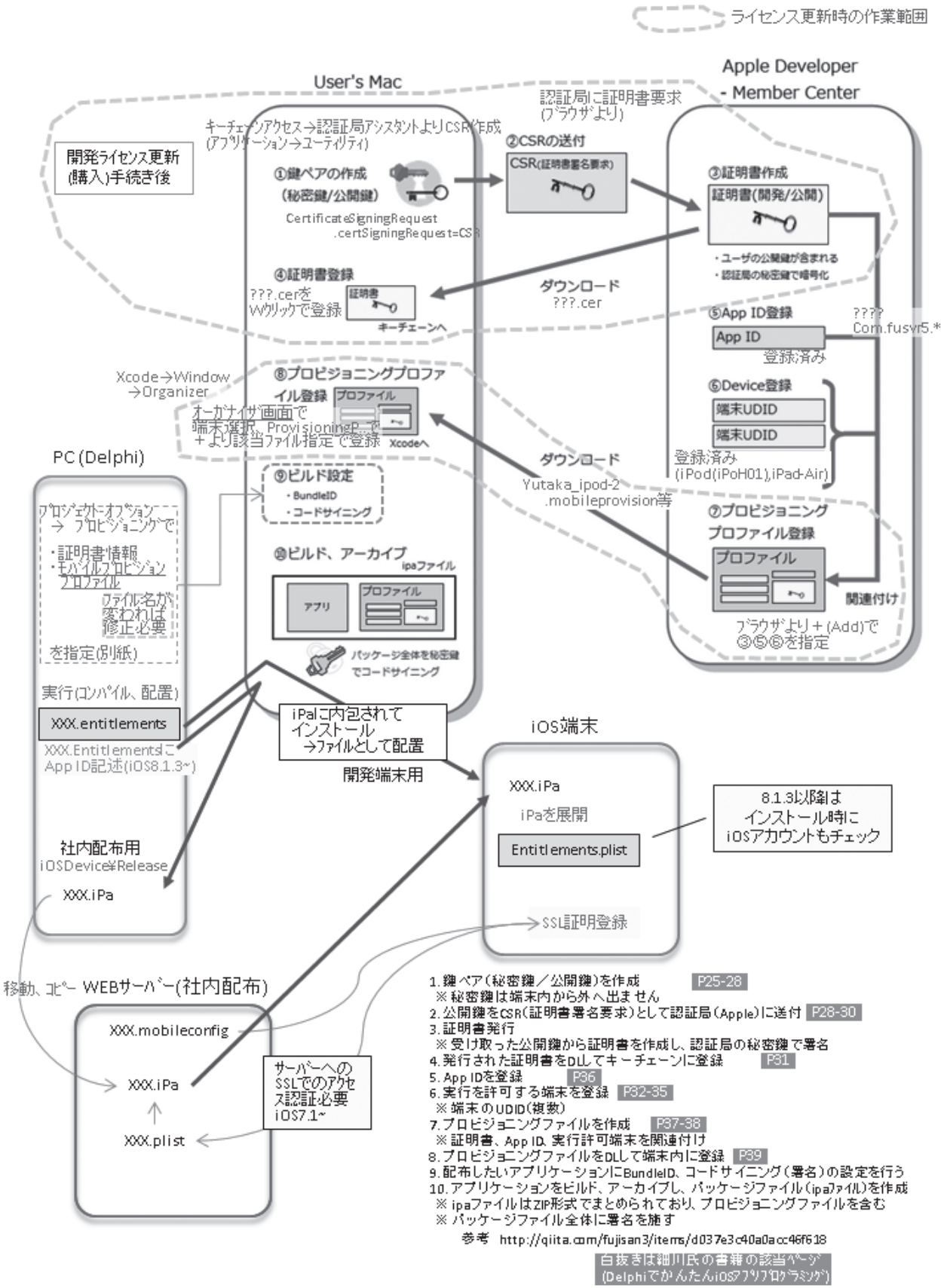


表3　立上げ時の主なトラブル

項目	内容
ｻｰﾊﾞｰｱｸｾｽのﾀｲﾐﾝｸﾞで固まる 一つの端末で固まると以降は別端末でも固まってしまう (DataSnapｻｰﾊﾞｰ機能の再起動が必要)	【原因】 DataSnapｻｰﾊﾞｰ側のﾏﾙﾁﾀｽｸ処理対応が設定されていない 【対応】 ①DataSnapｻｰﾊﾞｰｱﾌﾟﾘ側のTSQLConnectionのﾊﾟﾗﾒｰﾀ[Decimal Separator]を Y に設定 →以降良好 ②iPodからもDataSnapｻｰﾊﾞｰの再起動可能とするｱﾌﾟﾘを追加開発
撮影後にｱﾌﾟﾘが突然終了してしまう	【原因】撮影した写真ﾃﾞｰﾀをiPod内のｱﾙﾊﾞﾑに自動保存する処理でﾒﾓﾘｰを消費していた 【対応】該当部分でのﾒﾓﾘｰをｸﾘｱするよう修正 → 以降良好
ｻｰﾊﾞｰへのﾃﾞｰﾀ保存に時間がかかる	【原因】 無線LANの通信が遅い可能性 【対応】 無線LANの中継機を追加　→ 以降良好
DataSnapｻｰﾊﾞｰのｻｰﾋﾞｽが停止、ﾌﾘｰｽﾞ	【原因】 不明 【対応】 定期的な再起動(ｽｹｼﾞｭｰﾗで自動処理) → 様子見中
アプリの更新、ｲﾝｽﾄｰﾙが出来ない	【原因】 iOS8.1.3以降の仕様変更 【対応】 XXX.EntitlementsにAppIDを記述　→ 以降良好
アプリの更新、ｲﾝｽﾄｰﾙが出来ない (更新ﾍﾟｰｼﾞのﾘﾝｸﾀｯﾁで処理が開始せず)	【原因】 不明 【対応】 iOSを立上げ直して再トライ → 不能時は再トライ

図9　QRコード、一次元バーコードを併用での情報例

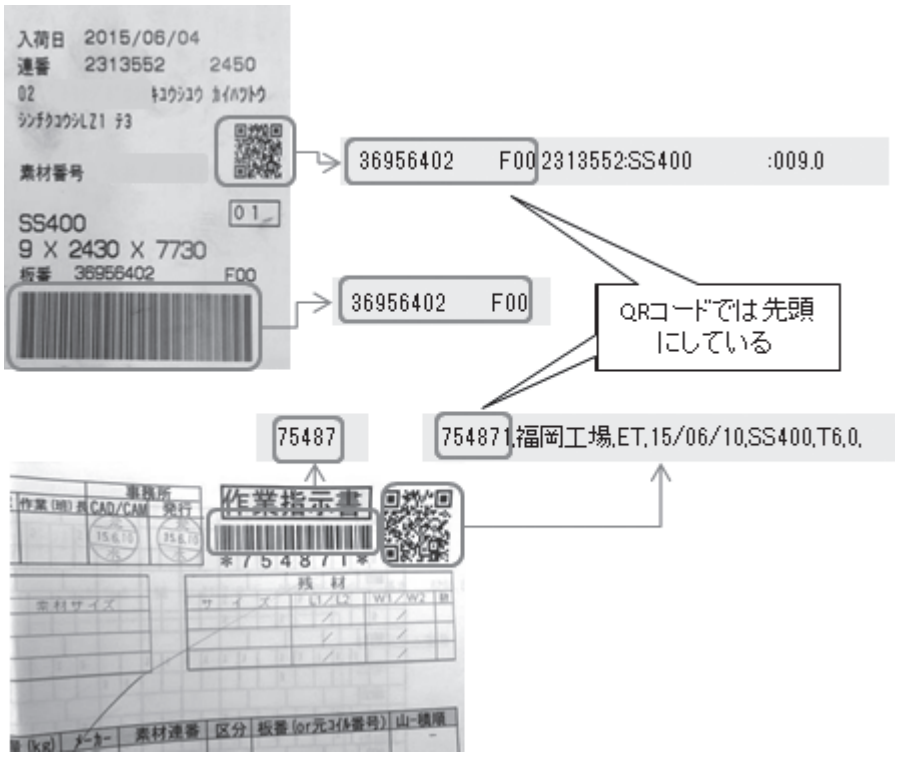


図10　WaveLink TE（5250エミュレータ）の文字数オーバー時の設定

図11　WaveLink TEでのバーコード読取起動

図12　同等機能での5250画面とGUI画面比較

図13　iPod Touchのマイク、スピーカーの位置

http://akiueo.blogspot.jp/2012/09/ipod-touchiphone5.html より

図14　iPod Touchのメッセージ機能使用例