一般論文

Mirandaのモバイル対応開発

トータルソリューション事業所　技術第1部　開発課
萬慶 久夫、中島 龍二

1．まえがき

IoT時代の到来とともに、スマートフォンやタブレットなどのモバイル端末を活用したソリューションへのニーズが高まっている。これを機に、生産現場の情報共有ソリューションとして、当社製のFA向けデータ収集ソフトウェア「Miranda」のモバイル対応開発（Android端末対応）を行った。

本稿では、モバイル対応開発の課題と解決策について紹介する。

2．Mirandaの紹介

2.1　製品概要

FAの生産現場において、製造プロセスの最適化を実現するためにはデータの収集が不可欠である。

Mirandaは、図1のようにMELSEC(注1)からデータ収集し製造プロセスの「見える化」を支援するソフトウェアである。

2006年の販売開始以降、ライン立ち上げ時の調整作業、稼働中の運転状態の記録、チョコ停(注2)の原因解析など、あらゆるシーンのデータ収集に活用できるソフトウェアとして鉄鋼・自動車・食品・紙パルプといった様々な分野で導入され、販売本数は2018年度末時点で累計1300本を超えている。

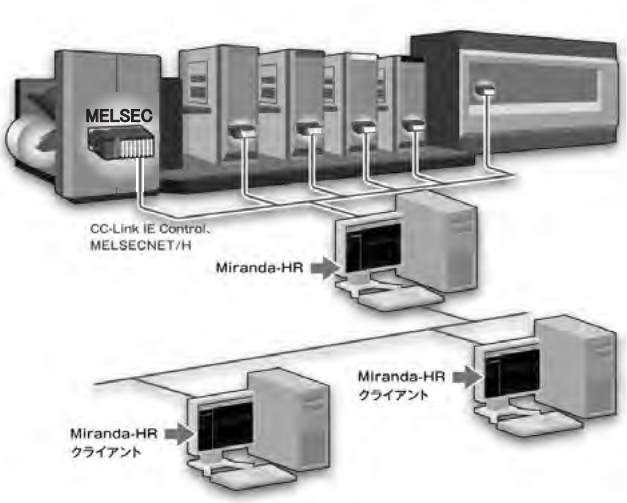

図１．Mirandaのシステム構成例

2.2　特長

(1)各種MELSECから簡単にデータ収集が可能

MirandaはMELSECの各種シリーズ(iQ-R、Q、QnA、A、FX、Lシリーズ)と豊富な接続経路(シリアル、USB、Ethernetなど)で接続可能である。

MELSECへユニットを増設することなくデータ収集できるため、MELSECとパソコンをケーブルで接続するだけで簡単に導入可能である。

(2)多彩なグラフレパートリー

収集したデータは、図2のようにトレンドグラフ、棒グラフ、パレート図など様々なグラフ形式で表示可能である。

用途に適したグラフ表示により、チョコ停発生時の原因解析や設備異常の傾向確認など、生産設備の改善に向けた調査を迅速に行うことができる。

図２．Mirandaのグラフ表示

2.3　今後の開発構想

Mirandaは2006年の販売開始以降、データ収集機能やグラフ表示機能など、データ収集システムの基本となる機能の拡充に注力し、開発を進めてきた。

近年、生産現場では更なる生産性向上と生産情報の集約・活用が求められている。今後は、基本機能の更なる拡充に加え、以下に示す使用性/拡張性/情報共有に向けた開発を進めることが重要である。

・使用性：解析時間短縮を目的とした使いやすさ向上
・拡張性：データベース/AI/他システムとの連携
・情報共有：IoT/クラウド/モバイルへの対応

本稿では、情報共有を目的としたモバイル対応開発について紹介する。

（注1）三菱電機株式会社製のシーケンサ
（注2）生産設備の何らかのトラブルにより、短時間の設備停止が繰り返し発生する事象。

3．Mirandaのモバイル対応開発

3.1　開発概要

Mirandaは、MELSEC内のデータを定周期で収集し、パソコンのデータベースに格納する。この収集データをモバイル端末で表示できるようモバイルアプリケーションの開発を行った。パソコンとモバイル端末の接続については、既納顧客への導入の容易性を優先し、回線工事や回線使用料が必要な公衆回線は使用せず、Wi-Fi接続による無線LANを利用するものとした。

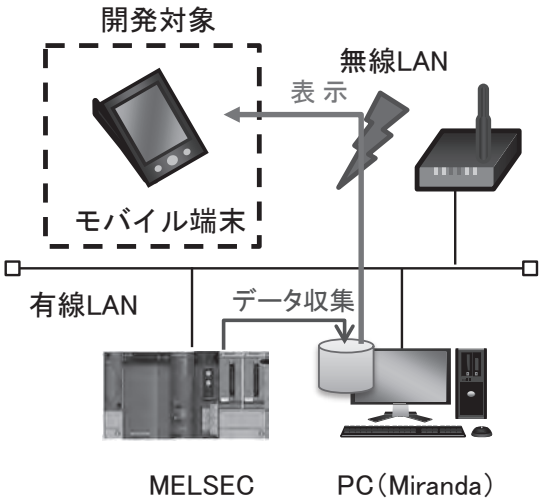

図3．システム構成

3.2　開発に採用した技術

モバイル対応開発では、モジュール性(注3)を高めることを目的にアーキテクチャパターンMVVM(Model-View-ViewModel)(注4)を採用した。また、ユーザーインタフェースの一部にモバイルUI特有のPUSH通知を採用した。モバイル対応開発に用いた特有の各機能について説明する。

3.2.1　アーキテクチャパターン

近年、モバイル端末では、Android(注5)、iOS(注6)、UWS(注7)といったプラットフォームの多様化が進んでいる。また、ソフトウェアは高機能化し、ソースコードの規模が大きくなっており、プラットフォームに合わせて新規実装すると、多大なコストを要する。コストを抑えつつ、新しいプラットフォームへ対応するためには、流用率を高める必要がある。流用率を高めるためには、モジュール性を考慮した設計が必要となる。一般的には、モジュール性を高めるため、MVC(Model-View-Controller)(注8)などのアーキテクチャパターンが使われる。モバイル対応開発に当たり、MVCよりもモジュール性が高くなるMVVMを採用した。

MVVMは、図4に示すとおり、Model、View、ViewModeの三層に分かれる。Modelには、データとビジネスロジックを、Viewには画面を、ViewModel(注9)には、画面の状態、及び、画面の状態を制御するロジックを配置する。双方向データバインディング(注10)の仕組みにより、ViewModelからViewの操作を不要とし、モジュール性を高めている。

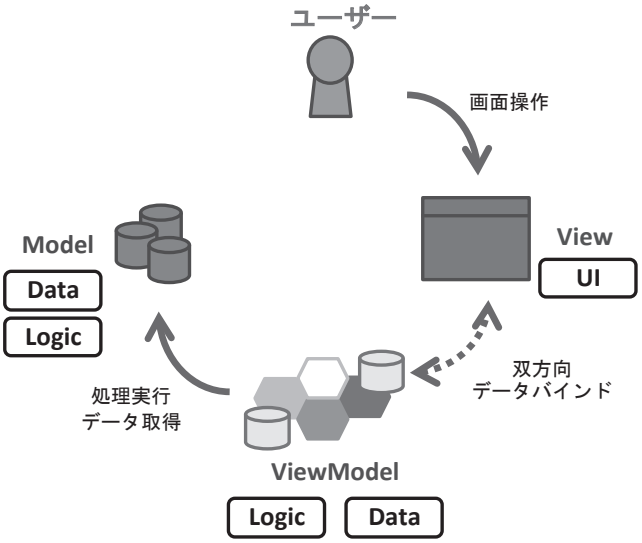

図4．MVVMパターン

3.2.2　ユーザーインタフェース

ユーザーインタフェース開発で用いた手法、実装した機能について説明する。

(1)マテリアルデザイン

ユーザーインタフェースの設計には、マテリアルデザインを取り入れた。マテリアルデザインとは、Googleが提唱するデザインガイドラインであり、要素の重なりを物理的にとらえ、平面(XとY軸)の中に、明確な奥行き(Z軸)の概念を持つ(画用紙に紙を重ねて貼り付けるイメージ)。PC向けに作成したデザインは、スマートフォンやタブレットなど、操作するデバイスが変わると、レイアウトが崩れたり、操作性を悪くなることがある。マテリアルデザインを取り入れることにより、直感的で統一した操作性を保つことができ、ユーザーがストレスを感じずに様々なデバイスを操作することができる。例えば、

(注3) 各モジュールの独立性。モジュール間のインタフェースが定義されており、相互の関連性が少ないほどモジュール性が高いと呼ぶ。
(注4) ソフトウェアアーキテクチャパターンのひとつ。MVCの派生パターンであり、ModelとViewの間を繋ぐViewModelという概念が導入された。
(注5) Google社のオペレーティングシステム。
(注6) アップル社のオペレーティングシステム。
(注7) Windows 10にて導入された、様々なデバイス上で動作するアプリケーションを単一のフレームワーク上に統合する仕組み。
(注8) ソフトウェアアーキテクチャパターンのひとつ。
(注9) ユーザーインタフェースに対する処理を除いたシステム固有の処理。画面の抽象化。
(注10) UIの操作・更新を抽象化することでデータの変更が自動的にUI変更へ反映されるようにする手法。

図5のように、画面構成要素を重ねることで、浮き出ている感と奥行きを表現している。

図５．マテリアルデザインを取り入れた画面

(2)プラットフォーム共通UI

プラットフォーム共通のUIとして、Xamarin.Formsを採用した。Xamarin.Formsは、モバイルアプリケーション開発におけるクロスプラットフォームなUIライブラリである。Xamarin.Formsを使用すれば、同一のUIコードを、Android、UWP、iOSのいずれの環境でも動作させることができる。だたし、ラジオボタンとチェックボックスは、iOSに存在しないUIであり、Xamarin.Formsでは提供されていないため、図6、図7に示すとおり、ラジオボタンとチェックボックスを独自に実装した。これにより、将来のiOS対応においても、画面が共通化されていることで、トータルで実装コストを低減できる。

チェックボックス

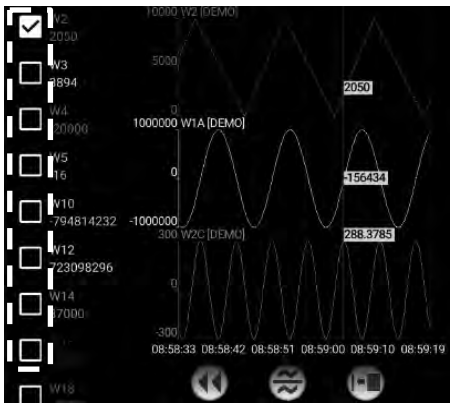

図６．チェックボックス

ラジオボタン

図７．ラジオボタン

(3)PUSH通知

Mirandaには検知した各種異常を通知する機能があり、センサの上下限値異常を知らせるために重要な機能のひとつである。モバイルに対応したMiranda(以降、モバイル版Miranda)には、異常通知を明示的に知らせることができるPUSH通知機能を採用した。PUSH通知の一般的な実装方法としては、Google Cloud Messaging(注11)などの外部サービスの利用が挙げられるが、Mirandaはローカル LAN環境で利用されているため、外部サービスを利用できない。そこで、外部サービスに依存しないMiranda用の通知サービスを独自に開発し、そのサービスをモバイル端末に常駐させることにより、図8に示すとおり、モバイル端末でセンサの上下限値異常を受け取る仕組みを実現した。

図８．PUSH通知

3.3　開発における課題と解決策

Mirandaのトレンドグラフ表示機能のソースコードはPC版では210 KL相当の規模であり、ソースコードをモバイル版Mirandaに移植した場合、製作・試験に多大な時間を要する。開発では、実装コストの削減のために処理部分のプログラムロジックをPC版と共用し、UIのみを新規開発する方針とした。

(注11) Google社が提供するモバイルアプリとサーバーアプリケーション間のメッセージングを容易にするサービス。

3.3.1　ソースコードの共用化

ソースコードを共有するに当たり、Windowsプラットフォームとモバイルプラットフォームの各ライブラリに互換性がない。そこで、図9に示すとおり、モバイルアプリケーションのプロジェクトへソースコードをリンク追加することにより、Windowsアプリケーションとモバイルアプリケーションでソースコードを共用した。Windowsアプリケーションとモバイルアプリケーションで共用することで、ソースコードの二重管理をなくし、PC版とソースコードの整合性を確保することができた。

図９．ソースコードリンク追加

3.3.2　モバイル版Mirandaへの移植における課題

前項のようにソースコードをリンク追加することで、モバイルアプリケーションのプロジェクトから、Mirandaのソースコードを参照できるようになる。しかし、Windows用UIのソースコードもリンクされるため、図10に示すとおり、.NET Frameworkの名前空間やクラスがモバイル向け.NET Frameworkに存在せず、ビルドエラーが発生した。

図10．ビルドエラー

リンク追加により発生したモバイル版Mirandaのビルドエラーの原因は、モバイル向けの.NET Frameworkが、Windows向け.NET Frameworkの一部の名前空間（UIや描画など、OSに依存する名前空間）をサポートしていないためであった。サポートされていない.NET Frameworkの名前空間及びクラスを、図11のとおり自作することでビルドエラーを解決した。具体的には、System.Windows.FormsなどUIに関するクラスは空実装とし、System.Drawing.Drawing2Dなどグラフ描画に必要な名前空間は、モバイル端末固有のAPIを使用して自作した。

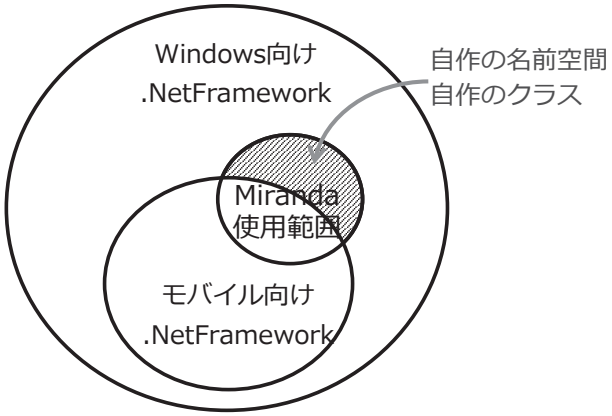

図11．.NET Framework使用範囲

3.3.3　MVVMの拡張

モバイルアプリケーションのプロジェクト構成は、図12に示すとおり、プラットフォーム（Android、iOS、UWP）共通のプロジェクト（以降、PCL側と表現）と、プラットフォーム固有であるAndroidのプロジェクト（以降、Native側と表現）から構成されるが、実装において次の制約がある。

・PCL側は使用できる.NET Frameworkが制限されており、流用元のMirandaクラスはNative側へ配置が必要。
・PCL側から、Native側を参照できない。

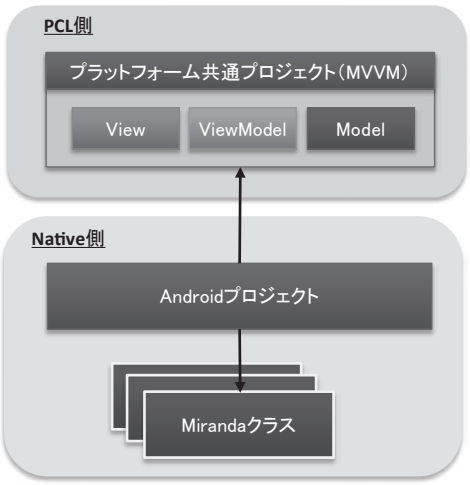

図12. プロジェクト構成

上記制約により、図13のとおり、MVVMのModel からMirandaクラスを参照できないため、クラス内のデータやビジネスロジックを使用できないという問題が発生した。

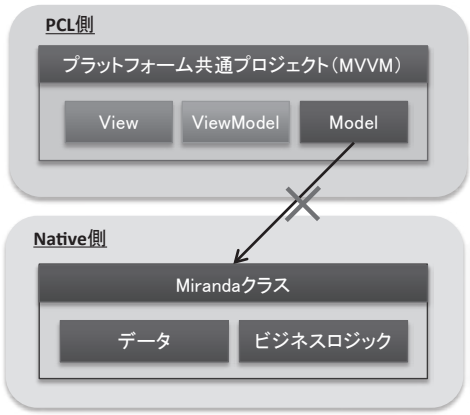

図13. PCL側からNative側を参照

上記問題を解決するため、図14のとおり、NativeModelという仕組みを開発し、MVVMのModel からNativeModelを経由し、Mirandaクラスを参照するように対処した。

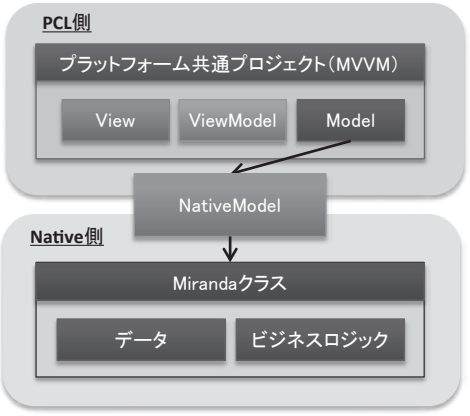

図14. NativeModel

NativeModelの仕組みを図15に示す。NativeModelは、アクセサ、Mirandaクラスラッパー抽象クラス、Mirandaクラスラッパー実装クラスで構成される。Mirandaクラスラッパーは、Mirandaクラスをラップするクラスである。Mirandaクラスラッパーは、自動的に、Dependency Serviceと呼ばれるDIコンテナへ設定(Inject)される。アクセサは、Dependency ServiceからMirandaクラスラッパーのインスタンスを取得し、Mirandaクラスラッパー経由で、Mirandaクラスを参照することができる。NativeModelは上記の仕組みをカプセル化し、外から見ると、PCL側、Native側を横断するModelのような振る舞いとなる。

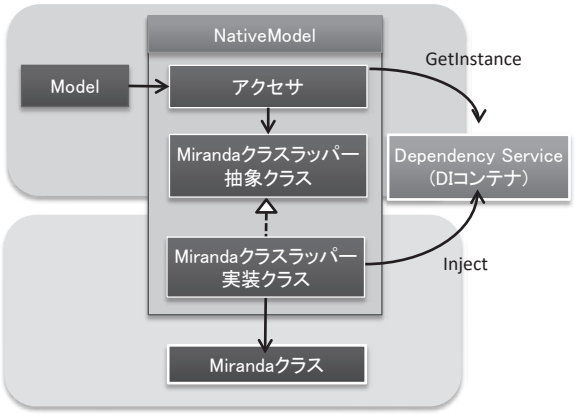

図15. NativeModelの詳細

DIコンテナとは、アプリケーションにDI(Dependency Injection: 依存性注入)機能を提供するフレームワークである。Dependency Injectionとは、デザインパターンの一種である。このデザインパターンを使用したイメージを図16に示す。オブジェクトBに変更が発生すると、通常の実装ではオブジェクトAの変更も必要となる。Dependency Injectionを適用した実装では、オブジェクトBの抽象クラスのAPIが変更されない限り、オブジェクトBに変更が発生しても、オブジェクトAの変更は不要となる。つまり、オブジェクトAB間で結合度が緩くなりモジュール性が高まる。

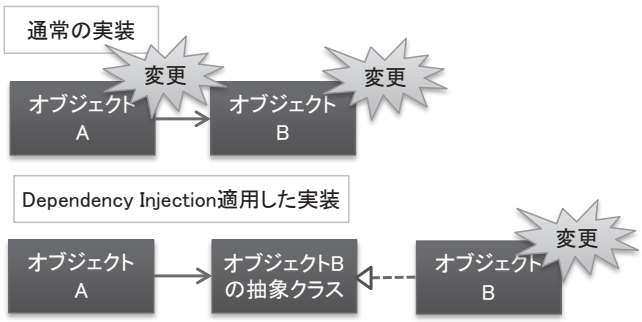

図16. 疎結合

3.4　開発の成果

ソースコードの共用やMVVMの拡張など前項で挙げた手法により、PC版Mirandaのソースコードを86%流用し、実装コストを抑えつつ、マテリアルデザインやPUSH通知など、モバイルアプリケーションの特色を活用したモバイル版Mirandaを開発することができた。また、Xamarin.Formsを用いることで、将来的に他のプラットフォームにも展開できるよう、保守性を確保することができた。

4．むすび

本開発では、Mirandaの一部機能をモバイルアプリケーションとして移植し、モバイル化の実現性を確認できた。今後は、2.3項で挙げた使用性、拡張性においても機能拡充を進め、製品の魅力向上を図ることで、販売規模を拡大していく。

最後に、本開発ならびに検証を通じて技術支援をいただいた社内外の関係各位に深く感謝申し上げる。

執筆者紹介

萬慶　久夫　　マンケイ　ヒサオ
2004年入社。主に自主ビジネス向けFAパッケージのソフトウェア開発に従事。現在、トータルソリューション事業所技術第1部開発課。

中島　龍二　　ナカシマ リュウジ
1996年入社。主に自主ビジネス向けFAパッケージのソフトウェア開発に従事。現在、トータルソリューション事業所技術第1部開発課グループリーダ。