コンテキストアウェアネスを用いた スマートデバイスアプリケーションのための低消費電力化

2014SE021 廣瀬 航　2014SE066 宮澤 真康　2010SE888 塗師 惇

指導教員: 沢田 篤史

1　はじめに

近年, スマートデバイスは技術の進歩により高性能化している. それに伴ってユーザのアプリケーションへの要求も多様化したことで, アプリケーションの多機能化が進み, アプリケーションを利用する機会が増加したことで消費する電力も増加してきた. スマートデバイスはユーザが持ち運びながら利用するので, 限られた容量のバッテリで稼働時間を確保することが重要である. そのような背景から, スマートデバイスの低消費電力化が求められている. 画面の輝度を調整したり, 音量を小さくするといったように, OS レベルで消費電力を抑えることもできるが, アプリケーションレベルで消費電力を抑えることで, よりスマートデバイスの消費電力を抑えることが出来る. これについて twitter[1] を例に挙げて説明する. twitter は新しい tweet を取得した時、自動でタイムラインを更新して表示する. twitter が備える省電力化の機能の一つに, この自動で更新する間隔を長めに変更する, という機能がある. この機能により, 更新間隔は長くなるが通信回数が減少することで, 消費電力を抑えることができる. しかし, 最新のタイムラインを取得することに時間がかかるので機能性は低下してしまう. OS レベルで消費電力を抑える機能も増えているが, このようにアプリケーションレベルで消費電力を抑える省電力機能も増えている.

OS レベルで消費電力を抑える場合, きめ細やかな省電力化を実現することが出来ないという問題があるが, アプリケーションレベルではユーザの使用状況に応じた省電力化を行うことが出来る。ここでも,twitter を例にあげて説明する. ユーザがタイムライン上で画像を見る機会があまりないという使用状況の場合, 画像を URL 表示にする省電力化を, コンテキストアウェアネスを用いることでアプリケーションレベルにおいて実現できる. このような省電力化の機能と, それに対応して妥協する特性との関係が明確になっておらず, アプリケーションを開発する際に重視すべき特性を誤って妥協してしまう可能性が存在する. さらに, 省電力を考慮したスマートデバイスアプリケーションを統一的に開発する方法が存在しないので, それぞれの省電力化の機能の変更を柔軟に行うことが困難となっている. さらには, 多様化した省電力化の機能の実現方法を個別に理解する必要があり, アプリケーション開発時の負担になっている.

本研究の目的は, 省電力を考慮したスマートデバイスアプリケーションの開発支援である. そのために, 低消費電力を考慮して, 既存のスマートデバイスアプリケーションのためのアーキテクチャ [2] を再設計する. アーキテクチャの設計にあたり, 省電力機能と妥協する非機能特性との対応関係を明確にするために整理する. これらのことから, アプリケーションに応じて非機能特性を考慮したモジュールを再利用することによる開発を提供できると考える.

本研究の手順として, 我々はスマートデバイスアプリケーションにおいて実装されている省電力機能の調査を行う .AppStore[3] の全てのカテゴリからそれぞれ 1 つずつアプリケーションの省電力機能を調査することで, 現在実装されている省電力機能を擬似的に網羅する. この調査から得た省電力機能群それぞれに対し, アプリケーションは概ね MVC アーキテクチャに従って構築されているとの仮定の下,

- 機能と Model,View,Controller との関わり
- 機能に対し妥協する非機能特性

以上の 2 項目を考察し分類する. 省電力機能と Model, View,Controller との関わりを分類した結果から得られたいくつかのグループを PBR パターンを用いて省電力アスペクトとして定義する. 我々の研究室で提案されているスマートデバイスアプリケーションのためのアーキテクチャにこの省電力アスペクトを組み込むことで低消費電力を考慮したアーキテクチャを構築する. このアーキテクチャの有用性を示すために, 実際に twitter クライアントのアプリケーション作成することで事例検証を行う.

2　背景技術

2.1　コンテキストアウェアネス

コンテキストアウェアネスとは, プログラムから観測することのできる外部環境やシステムの内部状態で, 時間や場所とともに変化し, それがプログラムの様々な実体の実行に影響を与えるものを指す概念のことである [4].

2.2　アスペクト指向技術

アスペクト指向技術とは, 横断的関心事を単一モジュールとすることでモジュール間の独立性を高める技術であり, オブジェクト指向の欠点を補うことができる. 横断的関心事とは, オブジェクト指向など, システムに対する支配的な関心事に基づいて分割された複数のモジュールに横断的に関係があり, 支配的な関心事によるモジュール分割では 1 つのモジュールにまとめにくい関心事のことである. アスペクト指向プログラミングの利点としては保守性が向上すること挙げられる.

2.3 CSA-Isys

CSA-Isys とは, 本研究室で提案されているスマート デバイスアプリケーションのためのアーキテクチャである. これを図 1 に示す.[1]

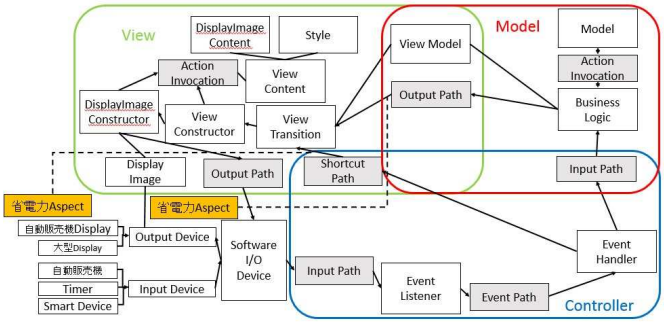

図 1 CSA-Isys

この CSA-Isys は, インタラクティブシステムのアーキテクチャ中心開発において基盤となり, 複数の既存のアーキテクチャを全て説明可能とする, という設計思想に基づいて設計されたアーキテクチャである. また,MVC アーキテクチャとその派生に基づいて設計されている.PBR パターンとは, 我々の研究室で提案されている技術であり, AspectJ のアドバイス記述を抽象化して設計された, 動的再構成を目的とした自己適応のためのアーキテクチャパターンである.CSA-Isys においては, コード記述の規範の提示と大粒度でのコンポーネントの再利用を可能にしている.

3 省電力機能と非機能特性の対応関係

本章ではアプリケーションレベルで実装されている省電力機能の調査を行い, 省電力機能と非機能特性の対応関係を明らかにする. 低消費電力を考慮したスマート デバイスアプリケーションアーキテクチャを構築するので, アプリケーションレベルで制御することができる機能を整理する必要がある.AppleStore で取り扱われているアプリケーションを SNS, ゲーム等, 各カテゴリごとに調査を行うことで, 現在アプリケーションレベルで実装されている省電力化の機能を擬似的に網羅した. また、省電力化の機能と非機能特性の対応関係を整理した. この整理結果を表 1 に示す.

我々は省電力機能を擬似的に網羅しているので, 妥協する非機能特性は, 表 1 より ,

- 機能性
- 使用性
- 機能性，使用性
- 機能性，効率性
- 使用性，効率性
- 機能性，使用性，効率性

表 1 電力機能と非機能特性との対応関係

省電力技術	非機能特性
PUSH通知をoffにする。	機能性
効果音をoffにする。	機能性
動画の自動再生をoffにする。	機能性・使用性
位置情報の取得を停止する。	機能性
画質を低画質に変更する。	機能性
自動スリープをonにする。	使用性・効率性
オフタイマー機能を使う。	機能性・効率性
音質を低音質に変更する。	機能性・使用性
ゲームエフェクトをoffにする。	使用性
昼夜モード切替する。	機能性・使用性・効率性
音声案内設定をoffにする。	機能性・使用性
白黒設定にする。	機能性・使用性
ディスプレイの明るさを暗くする。	機能性・使用性
画像のプレビューをoffにする。	使用性
タイムラインを自動更新から手動更新にする。	機能性・使用性・効率性
タイムラインの画像表示をURL表示に変更する。	機能性・使用性・効率性

の 6 種類に分類できる. また, 省電力機能が妥協する非機能特性を整理したので, 重視する非機能特性を誤って妥協することを防ぐことができる.

4 省電力を考慮したアーキテクチャの再設計

本研究の 3 章で網羅した省電力機能と Model,View, Controller に対しての関わりについて調査する. その調査結果を表 2 に示す.

表 2 電力機能と MVC との関係

省電力技術	MVC
PUSH通知をoffにする。	C
効果音をoffにする。	V
動画の自動再生をoffにする。	V
位置情報の取得を停止する。	MC
画質を低画質に変更する。	V
自動スリープをonにする。	MV
オフタイマー機能を使う。	MV
音質を低音質に変更する。	V
ゲームエフェクトをoffにする。	V
昼夜モード切替する。	V
音声案内設定をoffにする。	V
白黒設定にする。	V
ディスプレイの明るさを暗くする。	V
画像のプレビューをoffにする。	V
タイムラインを自動更新から手動更新にする。	MVC
タイムラインの画像表示をURL表示に変更する。	MVC

この結果から ,Model,View,Controller に対しての関わりは,

- View
- Model
- Model と View
- Model と Controller

以上の 4 つである. これらを省電力アスペクトとして定義する. 本研究室で提案されているスマート デバイスアプリケーションのためのアーキテクチャ CSA-Isys に, 本研究で定義した省電力アスペクトを組込むことで省電力を考慮したアーキテクチャを構築する.

4.1 省電力アスペクトの定義

我々は, この CSA-Isys 上に省電力コンサーンが横断的関心事として存在していると考える. ここで、PBR パター

ンを用いて横断的関心事である省電力コンサーンを省電力アスペクトとして分離することを考える．PBR パターンを用いて定義した省電力アスペクトを図 2 に示す．

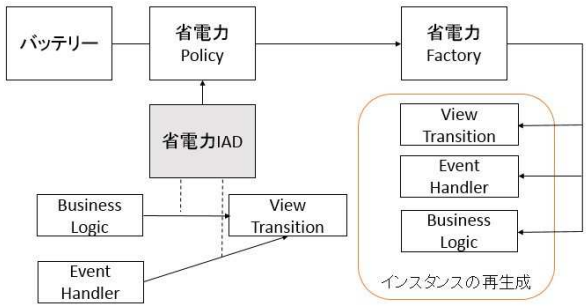

図 2　省電力アスペクト

我々は, 画面遷移をするタイミングでコンテキストを確認し, 省電力機能を働かせようと考えたので, ViewTransition へ送られるメッセージに省電力アスペクトをフックさせた．この考えに基づき, 省電力アスペクトを CSA-Isys に付加したアーキテクチャを図 3 に示す．

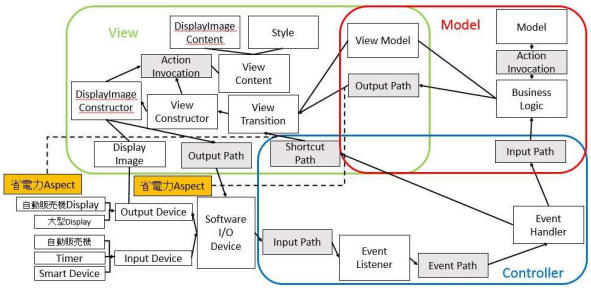

図 3　省電力アスペクトを付加した CSA-Isys

4.2　提案アーキテクチャの妥当性

前節の図 3 に示したアーキテクチャの妥当性を確認する．我々は, このアーキテクチャの妥当性を確認する必要があると考える．事例検証としてこのアーキテクチャを用いてアプリケーション設計を行う．下記に示す選択技術によって, 具象アーキテクチャを構築する．

- インタラクティブシステム:WebApplication
- プログラミング言語:Java
- アプリケーションフレームワーク:Java Servlet
- 外部との通信 Protcol:HTTP
- 具象表現方式:HTML
- 外部イベント 表現形式:Form データ
- 内部イベント 表現形式:Java Event
- モジュール構成法:状態遷移機械としてモデル化
- アーキテクチャスタイル:AM-MVC

構築した具象アーキテクチャを図 4 に示す．

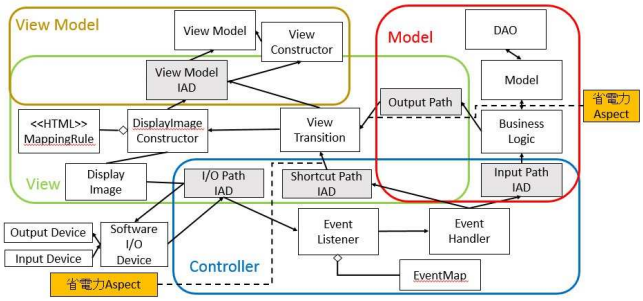

図 4　具象アーキテクチャ

事例検証に用いるアプリケーションは,twitter クライアントである．このアプリケーションが備える機能は,「タイムラインの自動更新」「ツイートの送信」の二つである．また, アプリケーション自身がバッテリー残量を確認することができる．アプリケーションはこのバッテリー残量をコンテキストとしており, バッテリー残量が 50 %を下回ったことをアプリケーションが確認した際に省電力機能が働く．つまり, ユーザが省電力機能の on,off を操作するのではなく, アプリケーションがバッテリー残量を判断して, 省電力化するのである．アプリケーションの省電力機能として, 本研究で調査した省電力機能群の「タイムラインの自動更新を手動更新に変更する」,「タイムラインの写真表示を URL 表示に変更」し, 写真の「画質を低画質に変更する」という二つを備えることにする．この時,「タイムラインの自動更新を手動更新に変更する」という省電力機能を利用した場合の省電力アスペクトを図 5 に示す．

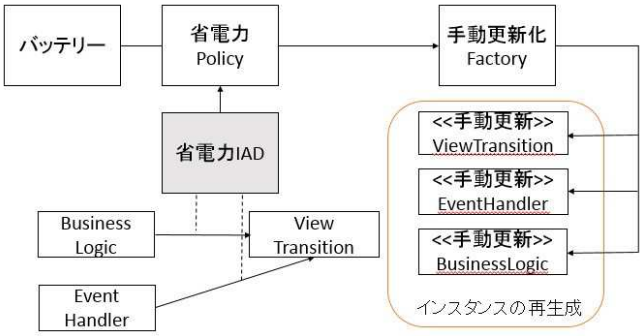

図 5　省電力アスペクトの一例

図 4 の具象アーキテクチャを用いてアプリケーションのクラス図と状態遷移図の設計を行う．クラス図を図 6, イベントハンドラの状態遷移図を図 7, 省電力機能が働いている時の状態遷移図を図 8 に示す．結果として，アーキテクチャに基づき，省電力に関する記述を局所化して実現することができた．

5　考察

アプリケーションレベルの省電力化が OS レベルの省電力化と比較して優れていることは, ユーザの使用状況

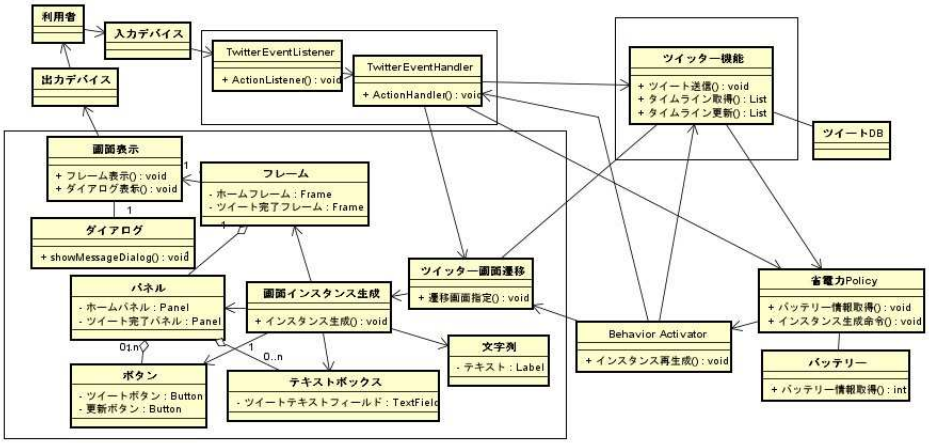

図 6 クラス図

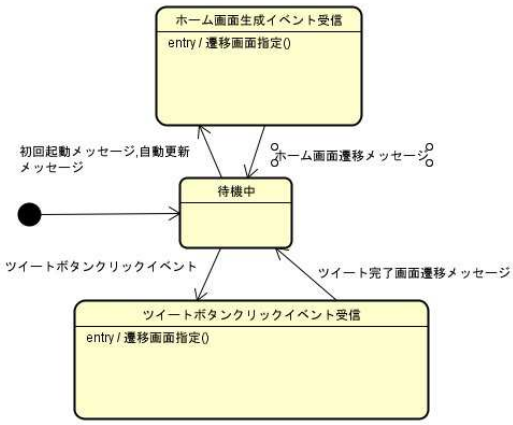

図 7 イベントハンドラの状態遷移図

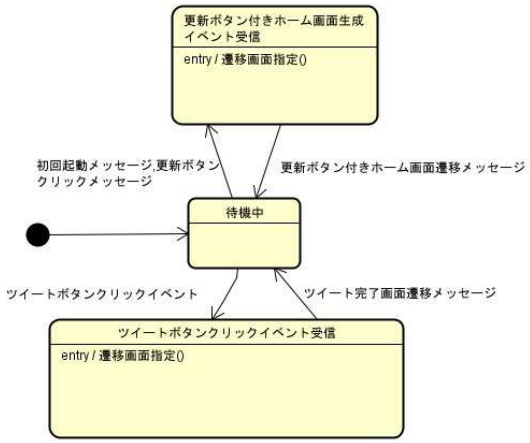

図 8 省電力機能が働いている状態遷移図

に応じたきめ細やかな省電力化を実現できることである．これに対し OS レベルが優れていることは, ユーザの使用状況に左右されずにいつでも省電力できることである．また，本研究では省電力アスペクトを定義し，内部の省電力 Policy がバッテリー残量を判断していた．しかし, ViewTransition がバッテリー残量を判断するというアプローチが考えられる．この場合, 本研究を上回る利点としてプログラムの量が少なくなることが挙げられる．しかし, 欠点として, 省電力アスペクトを定義しないので, 省電力に関するプログラムが散在してしまい, 保守性が低下することが挙げられる．

6 おわりに

本研究では AppStore で取り扱われているアプリケーションをカテゴリごとに調査をすることで既存の消費電力技術を擬似的に網羅した．また, 消費電力技術と Model, View,Controller との関係と, 妥協する非機能特性との関係を調査した．また, 既存の省電力機能を省電力アスペクトとして分離し, 本研究室で提案されているアーキテクチャ CSA-Isys に組込み, スマートデバイスアプリケーションのための低消費電力の提案アーキテクチャを構築した．これにより, アプリケーション毎に重視する非機能特性を妥協しない省電力機能が選択でき, その機能を適用したアーキテクチャが構築できるようになったので, アプリケーション開発時の負担を減らすことが可能となった．さらに, 図 6, 図 7, 図 8 のようにアプリケーション設計を行うことで, このアーキテクチャの妥当性を確認した．今後の課題として, 設計に基づいて実装を行い, どの程度消費電力を抑えることができるかを確認することが挙げられる．

参考文献

[1] https://twitter.com/

[2] 江坂篤侍, 野呂昌満, 沢田篤史," インタラクティブシステムのための共通アーキテクチャの設計" ソフトウェア工学の基礎 (日本ソフトウェア科学会 FOSE2017), pp. 129-134,2017

[3] https://www.apple.com/jp/ios/app-store/

[4] 紙名哲生:“ 文脈指向プログラミングの要素技術と展望” コンピュータソフトウェア,vol.31,no.1pp.3-13,2014.