低消費電力化を考慮したスマートデバイスアプリケーションの構築支援に関する研究

2012SE012 江口有香　2012SE215 三小田創太
2012SE241 鈴木勇気
指導教員：沢田篤史

1　はじめに

近年，スマートフォンやタブレット端末などのスマートデバイスが急速に普及し，スマートデバイスの用途が多様化している．その結果，スマートデバイスの高性能化が求められ消費電力が増加してきた．一般に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り，したがって，スマートデバイスの低消費電力化が求められている．

低消費電力化のための方法には，画面のバックスクリーンの輝度の設定，点灯時間の設定などがあるが，それらはオペレーティングシステムにより制御されることが多い．オペレーティングシステムによる制御を利用すれば，アプリケーションの開発者は低消費電力制御について考慮しなくて良いが，その反面アプリケーションの動作や処理状態を反映した緻密な制御ができなくなるという問題がある．

本研究の目的はスマートデバイスのための低消費電力化を考慮したアプリケーション構築支援技術の実現である．この技術を用いることでオペレーティングシステムが提供する消費電力制御機能を用いることなく，個々のアプリケーション独自の制御を実現することが容易になる．その結果，処理状態を緻密に反映した低消費電力制御機能をもつアプリケーションの開発が容易になるとともに，アプリケーションに特化した消費電力制御機能を適切にモジュール化し，デバイスを跨いで再利用することが可能となる．

スマートデバイスのための低消費電力化を考慮したアプリケーション開発論を提案するために，本研究ではスマートデバイスのオペレーティングシステムにおける低消費電力制御機能の調査に基づいて，アプリケーションレベルで制御可能な消費電力制御の仕様を記述する方式を提案する．さらに，この仕様記述に基づき消費電力制御を行うためのソフトウェアアーキテクチャの構築を行う．アーキテクチャはアスペクト指向の考え方に基づき，消費電力制御に関するコンポーネントをアスペクトとして分離し，再利用可能性に考慮して設計する．

2　スマートデバイスにおける消費電力制御機能とその課題

スマートデバイスの急速な普及によりユーザーの期待がスマートデバイスに集まり，より高いパフォーマンスを要求されるようになった．ユーザの要求により，スマートデバイスの機能は，開発者によって高性能化された．スマートデバイスの高性能化によって，バッテリ持続性が必要とされるようになり，スマートデバイスの消費電力の低減は最も重要な課題の一つとなっている．

2.1　スマートデバイスにおける消費電力化のアプローチ

本研究は，ソフトウェアで低消費電力化を実現するためのアプローチは三点挙げる．

- ディスプレイによる制御
- 通信機能による制御
- CPU の制御

一つ目は，電力消費に最も影響がある「ディスプレイ」である．輝度の調節やスリープ状態に入るまでの時間の設定などが挙げられる．

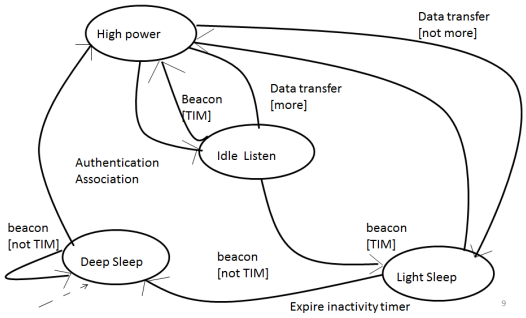

図 1　Wi-Fi STA 振る舞い [2]

二つ目は，Wi-Fi や LTE，Bluetooth などの「通信機能」である．Wi-Fi の電力消費の振る舞いを図 1 に示す．これは電力削減モードで作動する Wi-Fi のステーション (STA) を状態遷移図で表現したものである．この図は 4 つの電力状態とそれらの間の遷移からなる．

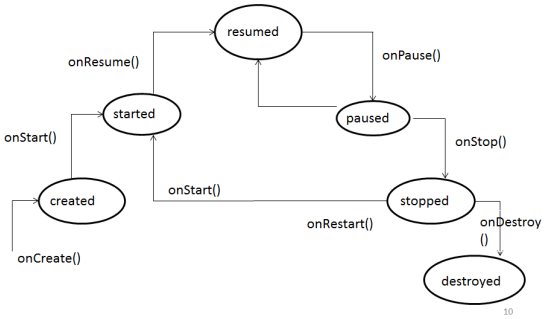

図 2　Activity のライフサイクル [2]

三つ目は,「CPU 使用率」である．AndroidOS における Activity(タスク) の状態画面遷移においての CPU の働きを表したものを，図 2 に示す．これは, アプリ実行開始時 onCreate() から始まり，ユーザーが操作することによってメソッドが呼ばれ, 終了時には onDestroy() が起動される．このパワーセーブ方法はある時間間隔，ユーザがスクリーンを操作しない時，電力管理サブシステムが動作して，システムを自動的にスリープにさせる. これによって，電池の消耗を軽減できる.

2.2 アプリケーションレベルでの低消費電力化の必要性

アプリケーションレベルで低消費電力化を制御する方法は十分に体系化されておらず, オペレーティングシステムに依存した消費電力制御が大半である. オペレーティングシステムに依存した状態ではアプリケーションの状態を緻密に反映した消費電力制御の実現できない．アプリケーションの状態を緻密に反映した消費電力制御が可能になれば個々のアプリケーションに対応した低消費電力化が可能である. 本研究ではオペレーティングシステムが提供する消費電力制御機能を用いることなく，個々のアプリケーションごとで制御を可能にすることでアプリケーションの状態を緻密に反映した消費電力制御の実現ができると考えた.

3 消費電力制御のための仕様記述方式

アプリケーションレベルで低消費電力制御を実現するにあたり，アプリケーションが制御する消費電力を整理する．アプリケーションの状態ごとに使用する機能使用しない機能に優先度を定める. デバイスのバッテリ残量 [5] に応じて，優先度の低い順に制限を設ける制御方法を考察する.

3.1 優先度表と制御表

本研究ではアプリケーションレベルで低消費電力化を実現するため，アプリケーションレベルで制御できる機能を整理する必要がある．Android の電力消費状態 [1] 調査した結果，電力消費が大きい機能が以下の機能である.

- ディスプレイ
- 通信機能

優先度の設計するにあたり，機能毎に制限していく優先度の定義を定める必要がある. 優先度の定義をもとに実際に AngryBirds を例にして優先度表を作成する．本研究では電力消費が大きい機能でもある,「ディスプレイ」と「通信機能」に着目し，優先度表を定めることで低消費電力化の実現が可能であると考える．ディスプレイではアプリケーション起動時の画面の最大輝度の設定, 画面の輝度を下げるまでの時間である輝度調節, 画面の初期状態の ON/OFF の設定を行なう. また通信機能では通信速度の制御を行なう.

3.2 優先度表

図 3 は本研究で提案する優先度表を作成するにあたり使用する優先度の設定をしたものである．優先度の設定は開発者が行なうものとしている.

優先度	1	2	3
機能	アプリケーションを動作させる上で持続させることが不可能になるのを防ぐもの。	アプリケーションが動作するために必要ではあるが必須ではないもの。	アプリケーションが動作するのに確実に必要であるもの。

図 3 機能ごと優先度設定

図 4 は AngryBirds というゲームアプリの消費電力を調査し，状態毎に分けたものである.

機能 / 状態	通信	ディスプレイ
ロード	21.1	19.8
選択画面	10.5	18.4
ゲームプレイ	0	27.3
データセーブ	13.3	20.2
選択画面	4.0	11.4
ゲームプレイ	0	16.0
合計	84.9 [mW]	113.1 [mW]

図 4 Angry Birds の電力消費量比較

本研究で扱う優先度は三段階に設定でき, 数字が大きいものほど優先度が高いものとしている．図 3 で設定した優先度を使用して優先度表を作成する．優先度表の書き方の例として，AngryBirds の消費電力状態を考慮した優先度表を作成したものを図 5 に図示する.

機能	通信	ディスプレイ		
	速度	最大輝度	輝度調節	画面 OFF
優先度	2	2	2	3

図 5 AngryBirds 優先度

3.3 制御表

バッテリ残量と優先度表を基にアプリケーションの消費電力機能の制御方法を示す制御表を作成する. バッテリの残量によって消費電力モードを設定する．以下に消費電力モードの例として，デバイスの電力消費状態におけるモードを設定したものを記述する.

- バッテリ残量:高

  ハイパワーモード

- バッテリ残量:中
  ノーマルモード
- バッテリ残量:低
  省エネモード

制御表では定義した優先度表の優先度を基に制御を行なう．制御に関して，ハイパワーモードでは，整理した機能の優先度に対して制限をせず制御を行なう．ノーマルモードでは，整理した機能から優先度表の優先度 1 の機能に対して制限を行なう．省エネモードでは整理した機能から優先度表の優先度 1，2 の制限を行なう. 作成する制御表の枠組みを図 6 に示す.

モード / 機能	速度	最大輝度	輝度調節	画面OFF
ハイパワーモード	全ての機能に対して制限をしない。			
ノーマルモード	優先度1の機能に対して制限を行なう。			
省エネモード	優先度1と2の機能に対して制限を行なう。			

図 6　機能をモード別に制御させた制御表

図 6 で設定した制御方法の枠組みを用いて, 制御表の作成する. 制御表の書き方の例として実際に AngryBirds の制御表を作成したものを図 7 に示す.

機能 / モード	速度	最大輝度	輝度調節	画面OFF
ハイパワーモード	OS	レベル	での	制御
ノーマルモード	IdleListen状態		60秒以上	
省エネモード	lightSleep状態	30%以下	30秒以上	

図 7　Angry Birds 制御表

3.4　優先度表と制御表の関係

特定のアプリケーションに対して, 開発者が機能毎に設定した優先度を整理した表を優先度表とする．制御表は優先度表を基に作成され，制御表は優先度表を基に作成することで優先度を活かした制御が可能になると考える．制御表の利用により, アプリケーションレベルで低消費電力を可能にする.

4　低消費電力化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

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共通アーキテクチャに，本研究で提案した仕様記述方式を適用し，アーキテクチャの拡張をする.

4.1　スマートデバイスアプリケーションのための共通アーキテクチャ

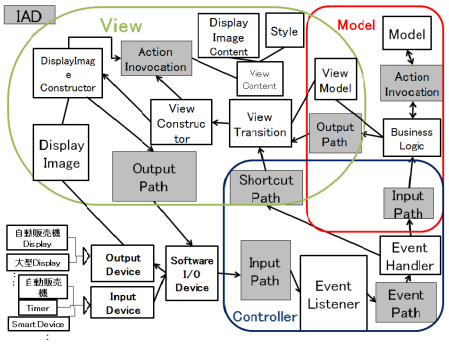

図 8　共通アーキテクチャ [1]

図 8 に示されているのは, 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共通アーキテクチャである．以下に共通アーキテクチャのコンポーネントを示す.

- Controller EventListener：イベントを内部形に変換し，アプリケーション使用者からのイベントを受け取り EventHandler に通知．EventHandler：自身の状態と EventListener からのイベントによって，Model または View にイベントを通知.
- Model Business Logic：Model や View 更新のロジックを状態遷移機会として抽象化したもの.
- Model：Database．ViewTransition：画面遷移を抽象化したもの．ViewContent：View のデータ構造を定義．ViewConstructor：Model を参照し，ViewContent のインスタンス化を行う．DisplayImageContent：ViewConstructor で生成した

開発において，開発者は Model，View，Controller に関連して次の 6 項目を定義する必要がある．アプリケーションのモデル，アプリケーションの状態遷移画面レイアウト (内容，役割，見栄え)，画面遷移イベント，コントローラの状態遷移実装は，この 6 項目を入力として，特定の開発技術に依存した一連の開発ステップ群によって行なわれる.

4.2　アスペクト指向アーキテクチャの設計

優先度表と制御表を用いたアプリケーションレベルでの低消費電力化の実現する際に必要となる構造を提案する．構成要素とその責務を以下に示す. 図 7 は各構成要素間の関係を示すアーキテクチャである.

- FunctionPolicy
  機能ごとの優先度を定義したもの
- Scheduler
  FunctionPolicy から受け取った情報と,Obsever から受け取ったバッテリ残量のモードを基に制御方法を Model に指示

- Observer
  バッテリ残量を監視と電力消費モードが切り替わったときに Scheduler に通知
- Function Scheduler と Function Policy の優先度についての共通部分をアスペクト指向で抽出したもの

優先度に関する FunctionPolicy と Scheduler と Function を Function アスペクトとする. 開発者は Function アスペクトに以下を記述するだけで, 機能ごとの特徴を考慮した低消費電力化を実現できる.

- 各モードの電力を消費する機能を制限をかけていく優先度

アプリケーションレベルで低消費電力化を実現するためのアスペクト指向アーキテクチャを提案する. アプリケーションレベルでの低消費電力化の実現に必要なコンポーネントをを共通アーキテクチャに組み込んだものを図 9 に示す. Function アスペクトは Controller 部分に関係し, Observer は MVC のどこにも関係がないと考察する.

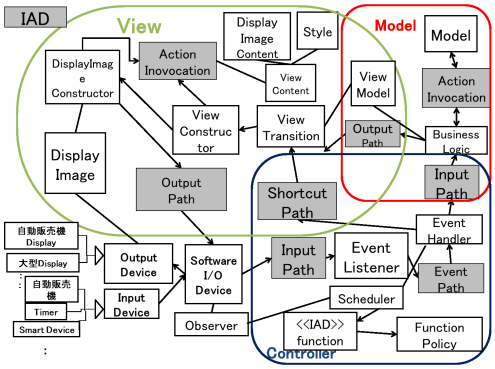

図 9 アスペクト指向アーキテクチャの適用

5 考察

本研究では，共通アーキテクチャから低消費電力化に関する横断的関心事を分離したアスペクト指向アーキテクチャを提案した．スマートデバイスの低消費電力化を実現する方法として，アプリケーションの開発者がアプリケーションの機能の一つとして，画面の輝度や通信状態において逐一低消費電力化を考慮してアプリケーションの開発をすることが挙げられる．この方法で開発されたアプリケーションでは，そのアプリケーションを利用する全てのユーザに, 低消費電力化を実現したアプリケーションを提供することができる．しかし，この方法ではアプリケーション内で表示される画面や， 通信状態に応じて低消費電力化を考慮する必要があるので，開発に要するコストがかかり，効率が悪い．

本研究で提案したアスペクト指向アーキテクチャを基にスマートデバイスアプリケーションの開発を行なうことで，消費電力を要する画面の輝度や通信状態を考慮することなく，低消費電力化を実現することができる．

6 おわりに

本研究では，スマートデバイスのための低消費電力化を考慮したアプリケーション開発方法論の提案を目的とした．その際，オペレーティングシステムにおいてのスマートデバイスの仕様・消費電力についての調査を行ない，「ディスプレイ」,「通信機能」についてアプリケーションレベルでの低消費電力化をするための機能の抽出を行なった．調査を基に，本研究はアプリケーション開発時に使用する，アプリケーションに対するスマートデバイスの機能の優先度を表としてまとめ，バッテリー残量による機能の制御方法を表した制御表としてまとめた．提案した優先度表と制御表を用い，本研究はアプリケーションアーキテクチャを設計した．アプリケーション開発者が，本研究で提案する開発方法論を適用することによって，他デバイスでも再利用可能なアプリケーションを開発する支援となる．提案したアプリケーションレベルの仕様記述制御モジュールに関して有用性の確認をした後，その仕様記述制御モジュールをオペレーティングシステムに対して実現することも可能であると考える．今後の課題として，提案したアーキテクチャをアプリケーションの状態を取得し，コントロールできるように更に拡張する必要がある. また，提案した仕様記述方式を用いたアプリケーションレベルでの低消費電力化について，特定のアプリケーションを用いて，妥当性を検証する必要がある．

参考文献

[1] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, Houjung Cha, “ AppScope:Application Energy Metering Framework for Android Smartphones using Kernel Activity Monitoring, ” Dept.of Computer Science, vol14, pp8-13

[2] 中島震. “ スマートフォン・アプリ電力消費のモデルベース解析, ”情報処理学会研究報告, Vol.2013, No.4, pp.1-8, 2013

[3] 江坂篤侍，野呂昌満，沢田篤史. “ インタラクティブソフトウェアの共通アーキテクチャの提案, ” ソフトウェア工学研究会報告，2015-SE-187(32), pp1-8，2015

[4] 鷲崎弘宜，坂本一憲，大杉直樹，権藤克彦，服部哲，久保淳人，小林隆志，大月美佳，丸山勝久，榊原彰, “ デザインパターンへのソフトウェア工学的取り組み, ” 情報処理会ソフトウェア工学研究会パターンワーキンググループ，Vol27，No.2，pp.1-3

[5] 高田伸彦, 吉田一誠, 鈴木雅実, 柳澤良一. “ Android OS のバージョンアップに影響を受けにくいアプリケーションの開発方法, ” 教育システム情報学会研究報告 27(5),pp141-146,2013