組込みシステムのアスペクト指向アーキテクチャに関する研究
―自律走行ロボットを事例として―

2011SE161 三島真亜久　2011SE169 水野高宏　2011SE241 塩野真
指導教員：沢田篤史

1　はじめに

近年，携帯電話，デジタル家電など我々の身の回りのシステムにはコンピュータが搭載され，組込みソフトウェアによって多様な機能やサービスが実現されている．組込みシステムの応用領域の拡大に伴ない，バッテリ駆動の組込みシステムが広く普及している．そのようなシステムでは，バッテリ寿命の延長が重要で，非機能要求として省電力化が大きい．一般に，組込みシステムにおける省電力化を実現する方法として以下のものが挙げられる．

- 消費電力の少ないハードウェアを利用し消費電力を削減
- オペレーティングシステム (以下，OS) がハードウェアの省電力機能を利用した消費電力の削減
- アプリケーションレベルでハードウェアを制御し消費電力を削減

これらのうち，アプリケーションレベルでハードウェアを制御し消費電力を削減する方法は十分に体系化されていない．個別のアプリケーション論理を考慮するとともに，アプリケーションが満たすべき他の品質特性とのトレードオフについて考慮しなくてはならないからである．

本研究の目的は，アプリケーションのレベルで省電力化を考慮したソフトウェアの開発支援である．この目的を達成するために，我々はアプリケーションレベルで省電力化を可能とするアスペクト指向アーキテクチャの提案をおこなう．アーキテクチャを設計するにあたり，他の非機能要求と省電力化のトレードオフを考慮する．アスペクト指向技術を適用することで省電力化を意識したアプリケーションを開発する際の保守性，再利用性の向上，設計者の開発労力を削減できる．

本研究では，アプリケーションレベルでの省電力化について明確に定義し，その実現方法について考察する．アプリケーションレベルでの省電力化を実現する技術として，リアルタイムデザインパターン [1] と OS による省電力化技術について調査した．その結果から，アプリケーションレベルで省電力化を実現するための構造を定義する．

省電力化に関する関心事は，組込みソフトウェアの複数のコンポーネントに横断することから，省電力化に関する関心事をアスペクトとして定義した．さらに，省電力化を実現する際にトレードオフの関係となる非機能要求として効率性，機能性，信頼性について整理した．この結果から，省電力を考慮したアスペクト指向アーキテクチャを提案する．提案するアーキテクチャを自律走行ロボットのアプリケーション開発に適用し，その有効性を検証する．

2　背景技術

2.1　E-AoSAS++

E-AoSAS++[4] は本研究室で提案されている組込みシステムのためのアスペクト指向アーキテクチャスタイルである．E-AoSAS++ では，システムを並行に動作する状態遷移機械 (以下，CSTM) の集合として定義している．E-AoSAS++ では，CSTM に共通するグローバルコンサーンとして，状態遷移及び並行処理についての論理をアスペクトとして記述する．さらに特定の CSTM のローカルコンサーンとして耐故障性，例外処理，実時間性を取り扱う．E-AoSAS++ に基づくアーキテクチャ記述には UML を用いる．E-AoSAS++ におけるアスペクトの表現にはステレオタイプを用いて UML の意味を拡張して表わす．

2.2　リアルタイムデザインパターン

デザインパターン [2] とはソフトウェアの設計の知識を蓄積し，名前をつけ再利用しやすいように特定の規約に従ってパターンカタログ化をしたものである．リアルタイムデザインパターンは，リアルタイム性を実現する際に起こりうる様々な問題を解消することを目的としたデザインパータン集である．本研究では以下の 2 つのデザインパターンを省電力化に関連するものとしてとり上げる．

2.2.1　Static Priority Pattern

Static Priority Pattern は，設計時に優先度を設定する際に，推奨される構造を示したパターンである．優先度に基づき Scheduler がタスクスケジュールをおこなう．

2.2.2　Highest Locker Pattern

Highest Locker Pattern は，優先度逆転の問題を解消する際に推奨される構造を示したパターンである．共有リソースが priority Ceiling を保持する．priority Ceiling より高い優先度を持つタスクのみが，共有リソースにアクセスできる．StaticPriorityPattern と同様に Scheduler を持つ．

2.3　OS による省電力化技術

アプリケーションレベルで省電力化を実現するにあたり，現在 OS で実現されている省電力化技術を調査した．Dynamic Energy Performance Scaling(以下，DEPS)[5] は，消費エネルギーと性能のトレードオフを最適化するフレームワークである．従来は個別に適用されてきた以下の 3 つの技術を，同時かつ統一的に適用する，リアルタイム OS を中心としたソフトウェア技術である．

- Dynamic Voltage Frequency Scaling(以下，DVFS)
  動的に電源電圧と動作周波数を最適に制御する．
- Dynamic Power Management(以下，DPM)
  DVFS の機能を利用し DPM が保持する Policy に基づき，コンポーネントの ON/OFF を制御する技術である．
- Dynamic Hardware Reconfiguration(以下，DHR)
  動的にハードウェアの構成を再構成する．

3 アプリケーションレベルでの省電力化を実現するアスペクト指向アーキテクチャ

本研究で実現する省電力化は，アプリケーション論理の中で，不必要なコンポーネントの動作を OFF にすることで電力の消費を抑えることと定義する．そこで，目標とするアプリケーションレベルでの省電力化の実現方法を考えた．目指すアプリケーションレベルでの省電力化は，省電力化と他の品質特性とのトレードオフを考慮し，電力を消費するハードウェアコンポーネントの ON/OFF を制御するものである．具体的な実現方法としては，制約として時間を指定し，一定時間が経過したらコンポーネントの ON/OFF を切り替える方法がある．他に，モードを用意し，バッテリ残量によってモードを変更する．モードごとに指定した制約に基づき，各ハードウェアコンポーネントの ON/OFF を切り替える方法などが挙げられる．本研究では，複数のバッテリ残量によるモードを用意し，制約には優先度を利用する．省電力化と共に考慮する品質特性ごとに，ハードウェアコンポーネントを制御する優先度を表として記述したものを複数用意する．バッテリ残量によるモードの切り替えで，参照する優先度表を動的に切り替えることで，考慮する品質特性に対して不必要なハードウェアコンポーネントの動作を OFF にする．考慮する品質特性が変更されても柔軟に対応するには，制約に優先度表を利用する方法が適していると考えた．

3.1 リアルタイムデザインパターン

本研究におけるアプリケーションレベルの省電力化を実現するにあたり，優先度を利用してハードウェアコンポーネントの ON/OFF をおこなう．優先度を利用する際の推奨される構造として，リアルタイムデザインパターンがある．リアルタイムデザインパターンの調査より，優先度に基づきコンポーネントの ON/OFF をおこなうスケジュール情報が含まれている Task Control Block(以下，TCB) と，Scheduler によるタスクスケジューリングを構成する必要がある．

3.2 OS による省電力化技術

アプリケーションレベルで省電力化を実現するにあたり，現在 OS で実現されている省電力化技術の調査をおこなった．DPM は，制約が記述されている Policy を保持している．Policy に基づき DVFS の機能を利用し省電力化を図る．OS による省電力化方法の調査より，制約を記述した Policy が必要になる．そこで，Policy と，アプリケーションレベルの省電力化方法として考えた，品質特性ごとに定めた優先度表を対応づける．

3.3 アプリケーションレベルでの省電力化を実現する仕組み

リアルタイムデザインパターンの調査と，OS による省電力化技術の調査により，アプリケーションレベルでの省電力化を実現する際に必要となる仕組みを提案する．構成要素とその責務を以下に示す．図 1 は各構成要素間の関係を示すアーキテクチャである．

- Priority Policy
  品質特性ごとの優先度を定めたもの
- Task Control Block(TCB)
  Policy から受け取った優先度をもとに，状態ごとの各ハードウェアコンポーネントを OFF にする順番を決定
- Scheduler
  TCB から受け取った情報と，Observer から受け取った情報をもとに，モードの変更と各スケジュールを Controller に指示
- Observer
  バッテリー残量を監視
- Controller
  Scheduler からの指示をもとに各コンポーネントの制御を Body に指示

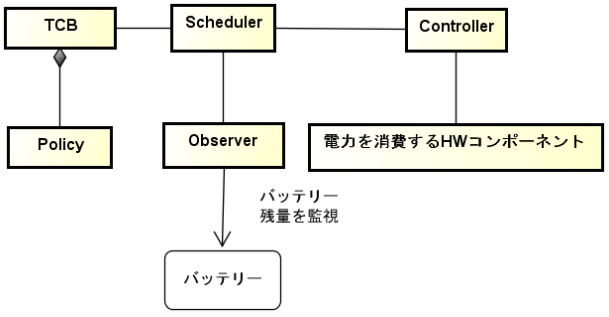

図 1 アプリケーションレベルの省電力化に必要な構造

3.4 アスペクト指向技術の適用

アプリケーションレベルで省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．3.3 で考えた，アプリケーションレベルで省電力化の実現に必要なアーキテクチャの，各モジュールに横断する関心事の特定をおこなう．Observer から通知されたバッテリ残量によるモード変更と，TCB の持つ優先度の情報を利用し Scheduler が各ハードウェアコンポーネントの ON/OFF のスケジュールをおこなう．それにより，モードごとに優先度を利用し

た制御に関する関心事が，複数のモジュールに横断する．優先度に関する横断的関心事の範囲を図 2 に示す．

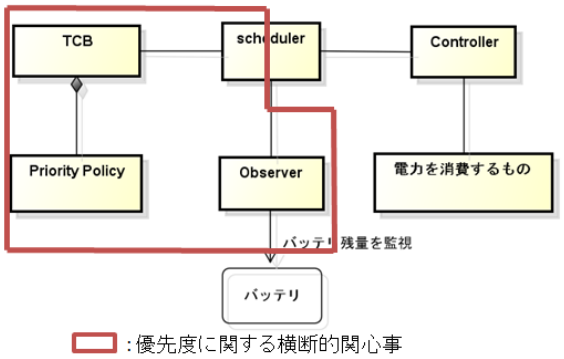

図 2　横断的関心事の特定

優先度に関する横断的関心事を Priority アスペクトとして分離する．本研究で提案する省電力化を考慮したアスペクト指向アーキテクチャを図 3 に示す．

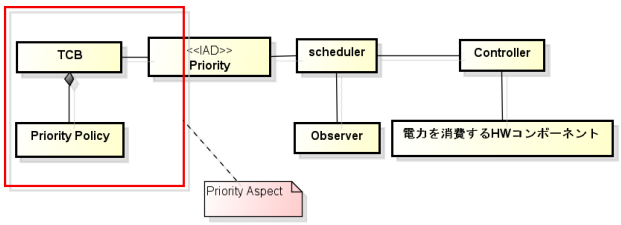

図 3　提案するアスペクト指向アーキテクチャ

開発者は Priority アスペクトに以下の 2 つを記述するだけで，品質特性を考慮した省電力化が実現できる．

- バッテリ残量に応じたモード
- 各モードの電力を消費するハードウェアコンポーネントを OFF にする優先度

優先度は，省電力化と他の品質特性とのトレードオフを考慮する必要がある．考慮する品質特性ごとに，ハードウェアコンポーネントの ON/OFF を制御する順番は異なる．そこで，優先度表の設計方法の提案が必要である．本研究では品質特性ごとに，ハードウェアコンポーネントの ON/OFF の制御をおこなう優先度表を一般化したものを提案する．

3.5　優先度表の設計

優先度表を設計するにあたり，バッテリ残量によるモードを設定する必要がある．設定した各モードごとに電力を消費するハードウェアコンポーネントを OFF にする優先度を定める．また，省電力化に関連する品質特性を整理し，考慮する品質特性ごとに優先度表を定める．

3.5.1　省電力化と関係する品質特性の整理

本研究で目標とするアプリケーションレベルでの省電力化方法を実現するには，省電力化と関係する品質特性を整理する必要がある．ISO9126 [3] の品質特性はソフトウェア品質の評価に関する国際基準である．ISO9126 で定められている品質特性より，本研究では利用する優先度表に違いが出ると考えた，以下の 3 つの品質特性を対象とする．

- 効率性 (efficiency)
- 機能性 (functionality)
- 信頼性 (reliability)

3.5.2　品質特性ごとの優先度の設定

本研究の省電力化を実現するための優先度表を，対象とする品質特性ごとに考えた．品質特性ごとにハードウェアコンポーネントの ON/OFF や消費電力を制御する優先度を表として記述する．Priority アスペクトとして記述した優先度表をいくつか用意する．考慮したい品質特性によって優先度表を動的に切り替える．優先度表の利用により，省電力化と他の品質特性を考慮し，消費電力の削減を可能にする．提案する優先度表を図 4 に示す．

- 機能性

OFFにする優先度	1	2	3
コンポーネント	組込みシステムが要求を正確に満たし続けることができなくなってしまうことを防ぐために必要なもの	組込みシステムの動作にあるべきではあるが確実に必要ではないもの	組込みシステムが動作をするのに確実に必要とされているもの

- 信頼性

OFFにする優先度	1	2	3
コンポーネント	組込みシステムが動作をするのに確実に必要とされているもの	組込みシステムの動作にあるべきではあるが確実に必要ではないもの	組込みシステムが要求を正確に満たし続けることができなくなってしまうことを防ぐために必要なもの

- 効率性

OFFにする優先度	1	2	3
コンポーネント	消費電力の大きいコンポーネントの順番		

図 4　品質特性ごとの優先度表

4　事例検証

自律走行ロボットに提案するアーキテクチャを適用し，妥当性の確認をおこなう．

4.1　本研究で用いる自律走行ロボットの仕様

本研究で取り上げる事例は自律走行ロボットである．仕様を，安全に走行し続ける自律走行ロボットとする．自律走行ロボットの各コンポーネントの責務を以下に示す．

- 超音波センサ：障害物を感知
- カラーセンサ：色を感知
- ジャイロセンサ：角度を計測
- タッチセンサ：衝撃を感知

4.2　非機能特性ごとの消費電力を制御する優先度表

自律走行ロボットのバッテリ残量によって，ノーマルモード，省エネモード，Emargency モードに切り替える．ノーマルモードでは，優先度に関係なくハードウェアコンポーネントを ON にする．省エネモードでは優先度 1 に設定したハードウェアコンポーネントを OFF にする．

Emargency モードでは優先度 1 と 2 に設定したハードウェアコンポーネントを OFF にする．自律走行ロボットのコンポーネントの消費電力を制御する優先度を，各品質特性ごとにまとめたものを図 5 に示す．

• 機能性

OFFにする優先度	3	3	2	2	1 (OFFにしない)
コンポーネント	タッチセンサ	超音波センサ	ジャイロセンサ	カラーセンサ	モータ

• 信頼性

OFFにする優先度	3	3	2 (1/2に制御)	1	1 (OFFにしない)
コンポーネント	ジャイロセンサ	カラーセンサ	モータ	タッチセンサ	超音波センサ

• 効率性

OFFにする優先度	2	2	2 (1/2に制御)	1	1
コンポーネント	超音波センサ	カラーセンサ	モータ	タッチセンサ	ジャイロセンサ

図 5　品質特性ごとの消費電力制御の優先度表

4.3　自律走行ロボットのアーキテクチャ

アプリケーションレベルでの省電力化を考慮したアスペクト指向アーキテクチャを自律走行ロボットに適用したものを以下の図 6 に示す．優先度に関する横断的関心事を，Priority アスペクトとして抽出し，TCB と PriorityPolicy それぞれに，モードと品質特性ごとによる優先度表を記述する．Observer が自律走行ロボットのバッテリ残量を監視し，Scheduler がタスクスケジューリングをおこなう．Body が Controller として優先度に基づきハードウェアコンポーネントの動作を OFF にする．

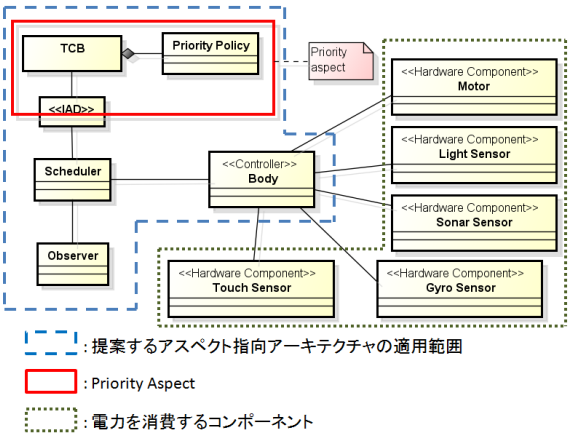

図 6　アスペクト指向アーキテクチャの適用

5　考察

品質特性ごとに記述した優先度表を Priority アスペクトとして分離し，1 つのアーキテクチャにまとめた．これにより，他の品質特性を考慮したアプリケーションレベルでの省電力化を実現できると考えた．Priority アスペクトには，優先度表と，モードごとにハードウェアコンポーネントを OFF にする優先度を記述する．省電力化を考慮したアスペクト指向アーキテクチャを提案することによって，開発者は省電力化を考慮した組込みソフトウェアを設計する際に，Priority アスペクトを記述するだけでよい．これにより，省電力化を意識したアプリケーションを体系的に構築できるようになる．アプリケーションレベルでの省電力化は，各ハードウェアコンポーネントを OFF にする優先度を詳細に設定できることから，OS による省電力化に比べ，非機能要求に対して柔軟に省電力化を実現できる．また，ハードウェア自身の消費電力を削減する場合と違い，アプリケーションレベルの省電力化では，高品質なハードウェアを利用することができない場合でも，要求を満たすことが可能になる．使用するハードウェアに依存せずに省電力化を実現できると考えられる．

6　おわりに

本研究では，アプリケーションレベルで省電力化を考慮したソフトウェアの開発支援のために，省電力化を考慮したアスペクト指向アーキテクチャの提案をした．その際，省電力化と他の品質特性を考慮するにあたり，品質特性ごとにハードウェアコンポーネントの ON/OFF を制御する優先度を表としてまとめた．今後の課題として，品質特性を複数考慮する際の優先度表の設計方法の提案，自律走行ロボットで省電力化が実現できているかの定量的な検証をおこなう必要がある．また，省電力化の関心事を，新たに E-AoSAS++ における 6 つ目の関心事として取り扱う方法の提案をする必要がある．

参考文献

[1] E. Douglas Jensen, *Real-Time Design Patterns Robust Scalable Architecture for Real-Time Systems*, Addison-Wesley, 2002.

[2] E .Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable ObjectOriented Software*, Addition-Wesley, 1995.

[3] ISO/IEC, Software engineering Product quality - Part 1: Quality model, 2001.

[4] M. Noro, A. Sawada, Y. Hachisu, and M.Banno, “E-AoSAS++ and its Software Development Environment,” *Proceedings of the 14th Asia-Pacific Software Engineering Conference(APSEC2007)*, pp. 206–213, 2007.

[5] 高瀬 英希, 小原 俊逸, 深谷 哲司, Lovic Gauthier, 石原 亨, 冨山 宏之, 高田 広章, “ソフトウェアとハードウェアの協調による組込みシステムの消費エネルギー最適化,” 組込みシステム技術に関するサマーワークショップ（SWEST12) 予稿集, vol12, pp. 2-3, 2010.