コンテキストアウェアネスを考慮した組込みソフトウェアアーキテクチャの設計と評価

―お掃除ロボットを題材に―

2012SE253 竹内 大地　2013SE198 杉山丈浩

指導教員：沢田篤史

1　はじめに

近年，計算機の高性能化により様々な場所で組込みシステムが活用されるようになった．システムの置かれている外部環境の情報をセンサにより正確に取得できるようになった．外部環境によってシステムの振る舞いを変化させる技術をコンテキストアウェアネスと言い，システムの振る舞いを変化させる外部環境の情報をコンテキストと言う．コンテキストアウェアネスを考慮したシステムの開発やセンサーの追加やシステムの仕様変更を行う場合，手続き的にソースコードを記述すると各モジュール間に振る舞いが横断的にまたがり，可読性，試験性が低下する．コンテキストアウェアネスを考慮した組込みシステムのために，関連する振る舞いを横断的関心事として抽出したアスペクト指向アーキテクチャが提案されている [1]．

コンテキストアウェアな組込みシステムでは，様々なコンテキストに応じた振る舞いを実装しなければならない．そのような組込みシステムにセンサーを追加しそれに応じた振る舞いの追加を行うと，各モジュールに振る舞いが横断的にまたがる．それによりソースコードの可読性や，システムの試験性が低下することが問題である．これを解決するために江坂らの提唱したアーキテクチャ [1] がある．このアーキテクチャはコンテキストに関連する処理をモジュール化することを目的にしているがシステムの動作モードに応じた振る舞いをアスペクトとして分離，抽出した場合を十分に考慮しておらず，これを改善する必要性がある．

本研究では，システムの動作モードごとの振る舞い，すなわち構成に関する関心事を構成アスペクトとして扱えるようなアーキテクチャを提案する．それによってソースコードの可読性や変更のしやすさ，システムの試験性の向上を目指す．

提案したアーキテクチャの有用性を確かめるために，事例としてお掃除ロボット [2] を参考にした組込みシステムを取り上げる．埃の量，湿度をコンテキストとし，掃除のモードを構成アスペクトとして抽出したシステムの実装を行う．オブジェクト指向アーキテクチャに基づく場合と提案アーキテクチャに基づく場合とで，ソースコードの可読性や変更のしやすさ，システムの試験性について定性的に比較し確認を行う．

2　背景技術

2.1　コンテキストアウェアネスを考慮したアスペクト指向アーキテクチャ

各モジュールにまたがる振る舞いを横断的関心事として抽出した，組込みシステムのためのアーキテクチャ [1] を図 1 に示す．

センサ，アクチュエータ，センサアクチュエータは図の中央上に表現されておりコンテキスト処理に関するモジュール (Context，Layer) はセンサアクチュエータや，アクチュエータにまたがっている．そして関心事としてリアルタイム性，対故障性，並行性がアスペクトとしてモジュール化されている事を示している．

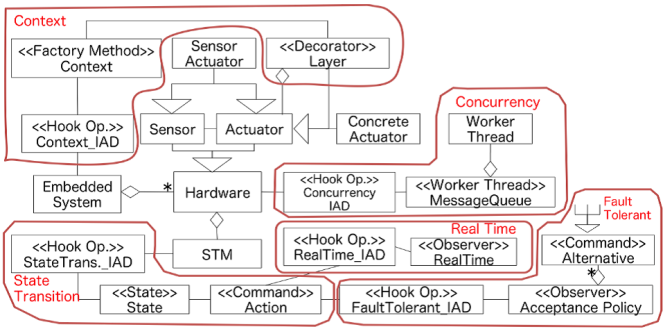

図 1　コンテキストアウェアネスを考慮した組み込みソフトウェアアーキテクチャ [1]

2.2　コンテキスト指向プログラミング（COP: Context Oriented Progrummig）

コンテキストに依存する振る舞いをモジュールとして実現するための手法は，主に以下の三要素から構成されている．

- 層 (レイヤ)…特定のコンテキストで実行される振る舞いを抽象化したもの．[3]
- 部分メソッド…特定のコンテキストの個々の振る舞いを実現されたプログラムの断片．[3]
- 活性化…層を動的に活性化・非活性化させ，現在活性な層の部分のメソッドのみ実行させる構造．[3]

2.3　デザインパターン

デザインパターンとは,，オブジェクト指向で関心事を分離するための手段であり，ソフトウェア設計時に遭遇する

典型的な問題に対し，その解決策に名前を付けて整理し，再利用しやすいようにカタログ化したものである．デザインパターンを利用することにより，再利用性の高い柔軟な設計が可能となる．

設計にはデザインパターンの Decorator パターンを適用している [4][5].

Decorator パターンのデコレーターとは修飾者のことであり，システムの機能を組み合わせる必要がある場合に機能の追加を実現可能とする．

3　構成アスペクトとコンテキストを統一的に扱うアーキテクチャの設計

本研究では，[1] のアーキテクチャを基礎にアスペクト指向の考え方でコンテスト，構成アスペクトを統一的に表す．アスペクトとしてコンテキストと構成アスペクトをそれぞれ分離して考え，それぞれを統一的に扱うためのアーキテクチャを提案する．

3.1　レイヤの設計

アクチュエータの振る舞いをレイヤとしてモジュール化し制御構造における条件と処理が分離されることから複雑さを軽減する．

レイヤの設計には Decorator パターンを用いている．オブジェクト指向では振る舞いに関する記述は様々なクラスに散在する．横断的にまたがった記述を Decorator パターンを用い，振る舞いに関する記述をウィーブさせるために用いている．実装手段として，関心事の分離の方法を Decorator パターンの次元と機能の分割を分離して実装を行っている．

コンテキストの変化に応じてアクチュエータを起動する処理をレイヤとして定義しレイヤの振る舞いをコンテキストに応じて切り替える層のことをレイヤアクティベータという．

図 2 では, 構成アスペクトとコンテキストの関連を示している．コンテキストには 2 種類準備し, 構成アスペクトを切り替えるコンテキスト (図中 ModeContext) とアクチュエータの振る舞いを切り替えるコンテキスト (図中 Context) が存在する．

アクチェータの振る舞いごとにレイヤを分割してそれぞれの振る舞いを状態遷移図で表し，レイヤアクティベータの切り替えを示した表をまとめて表した図を図 2 に示した．

3.2　構成アスペクトについて

3.2.1　構成コンサーン

システムの動作モードという観点を関心事としたときそれによって構成ということを実現するモジュール構造が特定されるという考えから構成コンサーンが存在する．

3.2.2　構成アスペクトの設計

構成コンサーンは，[1] で提案されているアーキテクチャではレイヤ層に横断的にまたがっている関心事である．その振る舞いを構成コンサーンごとにモジュールを分割しモードレイヤ層に振る舞いを記述する．構成コンサーンごと振る舞いが違ってくるのでモードクラスのサブクラスにモードレイヤが存在する．

3.3　状態遷移

それぞれのモードにおける振る舞いを切り替えるレイヤアクティベータやモードによって振る舞いが変わる．それぞれの状態遷移図を以下の図 2 に示す．

図 2 では, 左下の SensorActuator で外部環境の Update を行ない右上の Context の状態を更新する．更新されたコンテキストによって左上の LayerActivator を切り替える．そしてレイヤアクティベータによって活性層の変更を行なう．変更された活性層によって右下の Actuator の振る舞いを変更する．これらのやりとりをシステムの起動中はリアルタイムで行われている．

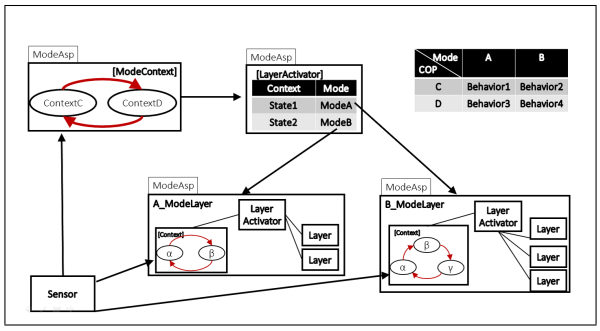

図 2　モードの状態遷移

3.4　アーキテクチャ

前章で述べた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をコンテキストと構成アスペクトをそれぞれ統一的に扱ったアーキテクチャに改良する．以下の図 3 に改良したアーキテクチャを示す．図 3 では ConfigrationContext_IAD でコンテキストによって変化する振る舞いを，ConfigrationContextAspect でアスペクトを分離している. そして，コンテキストも上記と同様にコンテキストに関する記述と，アスペクトに関する記述を分離している．

4　お掃除ロボットの設計

本研究ではお掃除ロボット [6] を題材として，提案しているアーキテクチャの事例検証を行う．

我々が題材とするお掃除ロボットは [5] を元に以下に仕様を示す．

お掃除ロボットにはノーマルモード，しっかりモード，マナーモードの 3 つのモードがある．コンテキストは埃

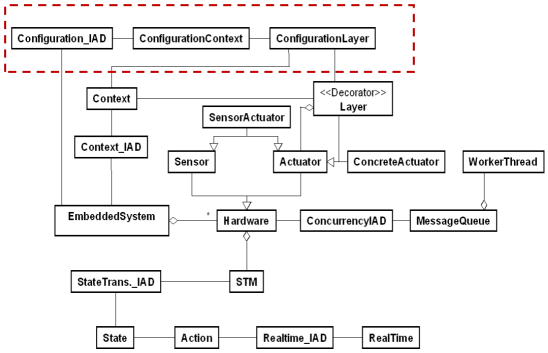

図 3　提案するアーキテクチャ

の量，湿度，時間とした．アクチュエータはドライビングモーターとクリーニングモータの 2 つ準備した．

ノーマルモードでは，埃の量と湿度関係なく 2 つのモーターの回転力は中で回転する．しっかりモードでは埃の量と湿度に合わせて 2 つのモーターの振る舞いは変わる．そしてノーマルモードとしっかりモードでは掃除する時間が夜であるとマナーモードに移行する．マナーモードでは，埃の量と湿度に関係なく 2 つのモーターの回転力は弱で回転する．そしてそれぞれのモードごとの振る舞いを設計した．

以下に事例検証を基にした，提案したアーキテクチャにおける設計を行う

4.1　モードレイヤについて

我々はお掃除ロボットのモードを構成コンサーンとし，構成アスペクトとして抽出した．お掃除ロボットのモードによってはコンテキストとは関係なく振る舞いは変化しない．我々は振る舞いに関するレイヤ層をモードレイヤとして振る舞いの実装を行った．

4.2　状態遷移モデルの設計

モードごとに取得するコンテキストが違ったりモードレイヤが違う場合がある．事例に基づくモードごとの状態遷移機械を以下の図 4 にハードモードの状態遷移機械を示しハードモードの状態遷移図を図 5 示した．

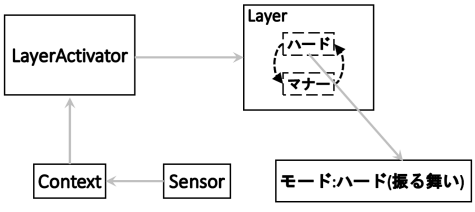

図 4　ハードモードの状態遷移機械

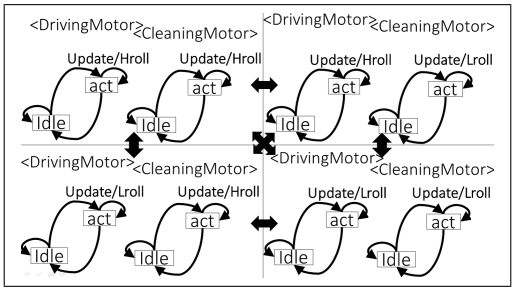

図 5　ハード:LayerActivator

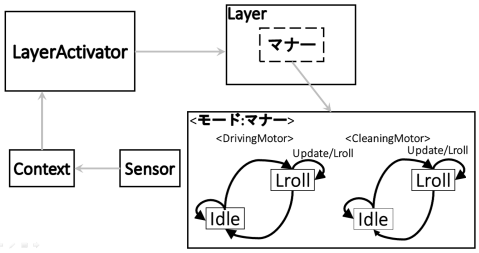

図 6　マナーモードの状態遷移機械

4.3　詳細設計と実装

4 章でこれまで記した状態遷移機械やモードレイヤの振る舞いを考慮したクラス図は以下の図 7 となる．

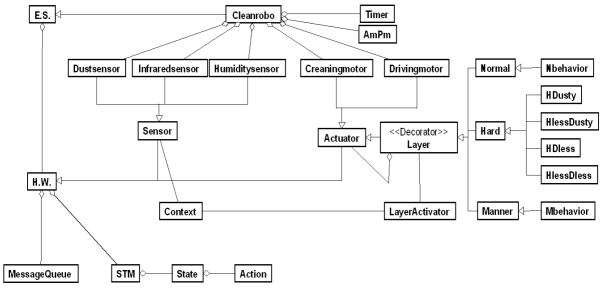

図 7　クラス図

事例に基づく LayerActivator の振る舞いを記述したソースコード以下の図 8 となる．

5　考察

本章では，本研究の評価を行なう．システムの動作モードに応じた振る舞いを構成アスペクトとして組込みシステムのモードによる振る舞いの切り替えがある．その箇所の比較を行なう．我々が行った本研究の比較の対象としてオブジェクト指向での設計，実装での比較を行ないアーキテクチャの有用性 (ソースコードの可読性，変更のしやすさ，静的な試験性) を考察する．

5.1　設計の比較

設計での比較した場合では，オブジェクト指向での設計の方が高度に抽象化されており全体の仕組みを把握が容易

```
#include"Arduino.h"
#include"motor.h"
void LA::LA(event mode){

 if(//夜でない){
  if(//マナー){

    mana mode = mana();
      mode.manamode();
     }

if(//通常){
 mana mode = nomal();
      mode.nomalmode();
 }

if(//しっかり){
 mana mode = sikkari();
      mode.sikarimode();
 }
}

if(//夜){
     mana m = mana();
       m.manamode();
 }
}
```

図 8 LayerActivator のソースコード

である．しかし提案しているアーキテクチャではオブジェクト指向を用いて設計するより可読性，試験性が向上し再利用性が容易になる．

以下の図 9 は事例 [6] をオブジェクト指向に基づいてクラス図を書いた．オブジェクト指向での設計はメインとなるクラスにハードウェアが持っているアクチュエータの抽象クラスを持つ．そのサブクラスにそれぞれのアクチュエータの振る舞いを持ったクラスを作った．

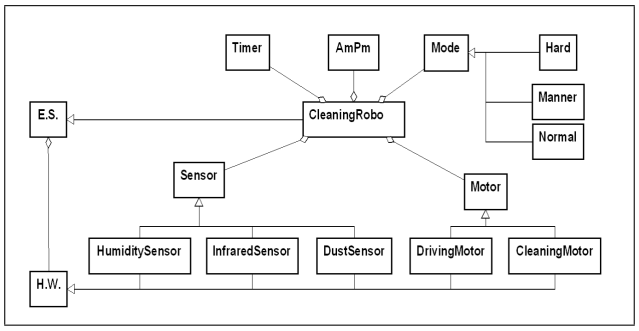

図 9 オブジェクト指向に基づくクラス図

今回我々は Decorator パターンはモードの追加しか考えておらず，モードごとによる振る舞いの追加は考慮していない．モードごとの振る舞いにも適用することで修正，追加を行うことが容易になり保守性が高まると考えられる．

実際に事例を用いた設計に適用した場合のレイヤ以下のクラス図を図 10 に示した．

5.2 実装の比較

オブジェクト指向で実装した際，コンテキストに関する振る舞いが複数のモジュールに横断的にまたがり可読性，試験性の低下を引き起こした。提案したアーキテクチャで実装した際，オブジェクト指向で横断的にまたがる振る舞

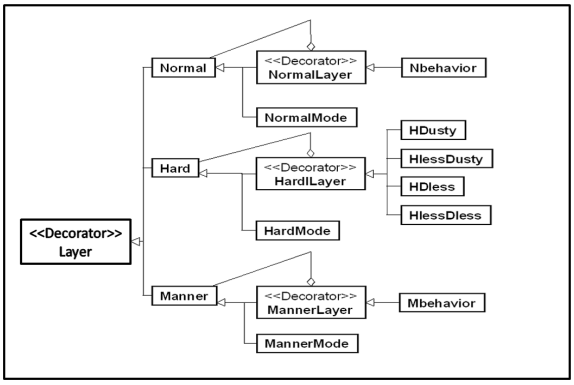

図 10 モードごとの振る舞いも Decorator を用いた場合

いを横断的関心事として抽出，さらに機能を構成アスペクトとして抽出したため，オブジェクト指向に比べ可読性と試験性の向上がみられた．

6 おわりに

本研究では，コンテキストアウェアネスを考慮した組込みソフトウェアアーキテクチャの設計と評価を行なった．我々はオブジェクト指向との比較しかできなく，他の設計方法を用いた場合との比較や似た研究との比較を行えなかった．今後の課題として他の設計方法との比較や同じような事例で提唱されている設計手法との比較が挙げられる．

参考文献

[1] 江坂篤侍，野呂昌満，沢田篤史，繁田雅信，谷口弘一，” 組込みシステムへのコンテキスト指向プログラミング技術の適用”，研究報告ソフトウェア工学 (SE)，2016-SE-193，no．11，pp．1-8，July．2016.

[2] 紙名哲生，” 文脈指向プログラムの要素技術と展望”，コンピュータソフトウェア，Vol．31，No．1(2014)，pp．3-13．April．2013.

[3] 紙名哲生，青谷知幸，増原英彦，玉井哲雄，” ユースケースを用いた文脈指向ソフトウェア開発”，ソフトウェアエンジニアリングシンポジウム，pp.1-8，September，2011.

[4] Erich Gamma, Richard Helm, Ralph Johonson, John Vlissides, *”Design Patterns - Elements of Reusable Object-Oriented Software”*, Addison-Wesley，1995.

[5] Gustavo Rossi, Silvia Gordillo, Fernando Layrdet *”Design Patterns for Context-Aware Adaptation”*, SAINT 2005 Workshops，pp.170-173, July．2005.

[6] 東芝：スマートロボットクリーナー，’http://www.toshiba.co.jp/living/cleaners/vc_rvs2/index_j.html