3G-02

コンテキスト指向プログラミングのロボット事例への適用の検討

小野建也†　　菅谷みどり†
芝浦工業大学

1．はじめに

近年ロボットが小型化，高性能化によりさまざまな環境で利用されるようになっている．動作環境が多様なロボットのプログラムにおいて，環境に応じて適切な動作を切り替えて動作することが重要である．しかし多様な動作環境に加え，状況ごとの処理は多岐にわたるため，プログラムのさまざまな場所に散在する．これは，ソフトウエアの構造を複雑化する問題がある．

本研究では，ロボットソフトウエア開発における本問題に対する，コンテキスト指向プログラミング[1,2]（以下 COP）の有用性を調査することを目的とする．COP はオブジェクト指向プログラミング（以下 OOP）の派生のひとつである．COP ではコンテキスト（システムを取り巻く環境）に応じて振る舞いをダイナミックに変更する．またコンテキストに依存した振る舞いをレイヤという単位に分離できそれらをアクティベイト／ディアクティベイト（ON/OFF）させることで動作を切り替える。またそれぞれのレイヤは独立しており，ソースコード間で影響しあわないという特性を持っている．これらの特性を用いて状況に応じた処理の変更を行う．またそれに伴うソフトウエア（ソースコード）の複雑化を軽減する．

2．ロボットプログラムの課題と検証

2．1 研究の目的

本研究では，課題の解決に対して，COP の有効性を実例をもとにコード記述をし，検証を行うことを目的とする．実際のロボット制御における実例をもとに，COP と OOP の両者で記述を行い，それらの差分を検証する．

2．2 ロボットプログラム

ロボットプログラムの例として掃除のアプリケーションを開発する．ここでは，コンテキストとしてネットワーク，バッテリ量を用いる．図 1 に想定する動作例を示した．バッテリ量，ネットワーク状況により，異なる動作が必要であるため，複雑なプログラミングとなる．ロボットは iRobot Create を用いる．既存の実装方法を OOP での実装とし，COP の実装と既存の実装との比較を行う．

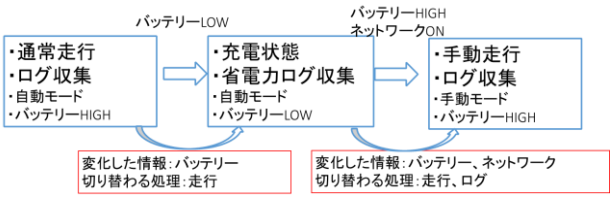

図 1：実装するロボットのアプリケーション動作

2．3 OOP での設計と実装

OOP の記述では仕組みの類似したストラテジーパターンを用いた．OOP を用いて記述した場合の問題点は状況に応じた処理の数が多岐にわたりクラスの数が膨大になることである．たとえばバッテリ情報とネットワーク接続情報を持ってロボットの走行プログラムを切り替える場合，それぞれの情報が 2 値しかとらない場合でも 4 通りの走行プログラムになり，さらにひとつ情報が増えるだけで8通りになる.そのためOOPの設計の場合クラスの数が 4 つ,8 つと増大する．

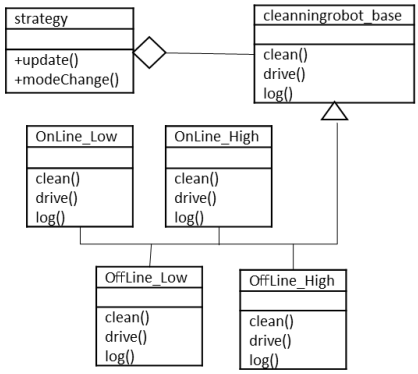

図 2　OOP クラス図

2．4 COP での設計・実装

一方で，COP ではコンテキスト依存する部分をそれぞれ分離,独立し,記述してそれぞれを組み合わせることにより記述するクラス，メソッドの数を減らすことができる（図 2） OOP で示した例の場合，バッテリ情報に依存する走行部分とネットワーク接続情報に依存する走行プログラムの部分を分離することで記述する部分は減少する．図 3 に概要図，図 4 に設計を示した

Case study:Context-Oriented Programming for Developing a Robot Application

†Shibaura Institude of　Technology

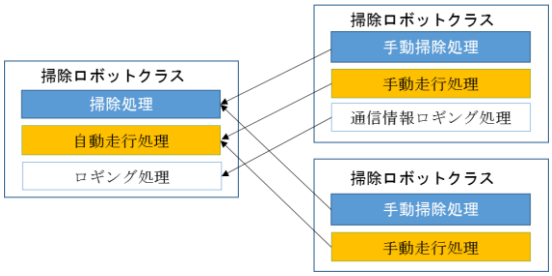

図3　システム概要図

COP でのアプリケーションの実装を行う．実装については以下の図4のクラス図で示す．

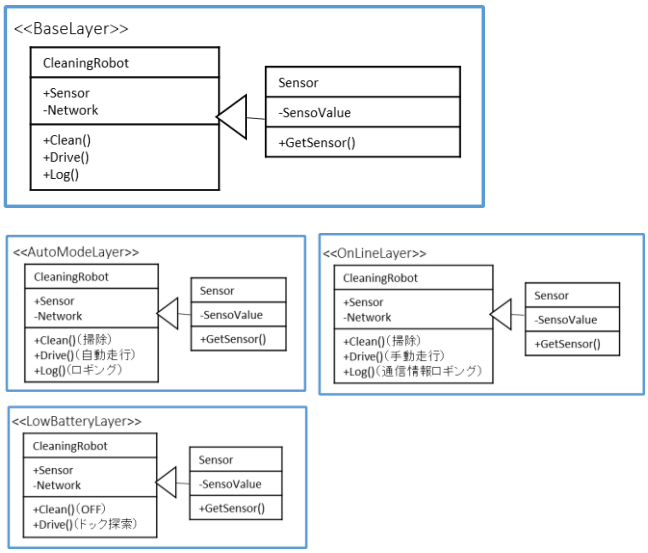

図4　COP クラス図

実装はC#でCOPの開発環境を実装しているフレームワーク RT-COS[3]で行った．

RT-COS の仕様としてレイヤを利用する場合のもととなる BaseLayer 定義し，BaseLayer を継承する形で AutoModeLayer, LowBatteryLayer, OnLineLayer の3つを切り替えて動作させる．

通常は AutoModeLayer が動作しており，Drive では壁の衝突を検知して回転しながら走行を行い，Clean ではブラシのモーターとバキュームを起動して掃除を，Log ではセンサー情報をファイルに書き出す．またソケットによる接続受付を行っており，接続要求があった場合 OnLineLayer がアクティベイトする．OnLineLayer は Drive と Clean が自動から手動になり Log では通常のデータに加えて通信情報を追加したものを書き出す．バッテリ残量が一定値以下になった場合 LowBatteryLayer がアクティベイトする．LowBatteryLayer では Drive がドックの探索（充電スポット）に変更され，Clean の掃除の処理は常に停止される．

レイヤのアクティベイト部分の記述は RT-COS の機能のオブザーバアスペクトを用いて記述した．

オブザーバアスペクトはレイヤのアクティベイトの部分を通常の記述とは分離するための仕組みでレイヤアクティベイトの記述部分をまとめて分離することができる．

4．評価

定量的な評価方法としてコード行数、保守性指標を用いて評価を行った．保守性指標はコード行数，サイクロマティック複雑度（分岐複雑さ），Halsted 複雑度（演算子，オペランドの複雑さ）を用いたコードの保守性を表し，0～100 までの点が高い方が保守性に優れている．

表1に示した通り全体としては OOP で記述した方がコード行数は多くなったが保守性指標は COP の方が高くなった．COP の方がコード行数が多くなった理由としては RT-COS を用いるための API の利用と通常の継承とは異なる BaseLayer の記述の必要があったためと思われる．また保守性指標の値において OOP の方が低い値が出た理由として，ストラテジーの切り替え部分のサイクロマティック複雑度が高く，複雑な分岐が集約されているためだった．これに対して RT-COS レイヤの切り替え部分のオブザーバアスペクトそれぞれのサイクロマティック複雑度が低く分散しているのでそれぞれの保守は容易といえる．

表1　コード評価

	コード行数	保守性指標
オブジェクト指向	437	79
コンテキスト指向	483	83

COP を用いて記述をした主観的な評価としてはレイヤを用いて設計を行う場合オブジェクト指向とは別の観点でモジュールを分ける必要があり，既存の設計方法，経験だけでは設計することが難しい．例えば複数のレイヤがアクティベイトする場合、どのレイヤの優先度が高いか、どのメソッドが優先的に呼ばれるかなどがある．一方で COP はそれぞれのレイヤが他のコンテキストについて把握する必要がなく，状況ごとの一つのアプリケーションとして見通しがいいといえる．

5．結論

本研究では定量的な評価において COP の方が保守性が高く，多くのコンテキストを扱うロボットアプリケーションにおいてコンテキストごとの処理の切り分けが行えることで組み合わせの複雑化を回避できると思われる．

しかし，不十分な点として設計の手法、言語としての実装がある．実際の開発に用いるには今後これらの課題を解決する必要があるといえる．

参考文献

[1] R. Hirschfeld, P. Costanza, O. Nierstarasz, “Context-Oriented Programming”. Journal of Object Technology. 2008, 7(3), p.125-151

[2] M. Appeltauer , et al., “A Comparison of Context-oriented Programming Languages”. COP’09 International Workshop on Context-Oriented Programming Article. 2009, No. 6

[3] 谷川　郁太 C#ベースのコンテキスト指向プログラミング開発環境 http://tanigawaikuta.bitbucket.org/ (2016/01/06)