多様な教育ロボット向けライブプログラミング環境の試作

谷川 郁太†, 大江 信宏‡, 佐藤 未来子‡, 大川 猛‡, 渡辺 晴美‡, 久住 憲嗣†, 福田 晃†

本研究では，多様な教育ロボットを対象としたライブプログラミング環境を試作する．近年，自身を取り巻く周囲の状況の変化に応じて実行時に振る舞いを変えるロボットの実現が求められている．このようなロボットは，自身のセンサから受け取る情報や，他のシステム・ロボットとの通信で送られてくる情報に基づき，周囲の状況の変化を感知する必要がある．これを実現するためのプログラムについて，アルゴリズムや対象とするロボット，部屋の状況等を変えつつ，実験できるライブプログラミング開発環境があれば，そのためにかかる時間や手間の削減が期待できる．本稿では，上記の開発環境を試作し，その狙いや実現のアイデアを述べる．

1. はじめに

近年，Internet of Things(IoT)や Industry 4.0 により，自身を取り巻く周囲の状況の変化に応じて，実行時に振る舞いを変えるロボットの実現が求められている．このようなロボットは，自身のセンサから受け取る情報や，他のシステムとの通信によって送られてくる情報に基づき，周囲の状況の変化を感知する必要がある．例えば，人が部屋を出したことを確認してから掃除を行うような掃除機ロボットでは，自身の持つカメラによって人がいなくなったことを感知するプログラムを書かなくてはならない．上記のプログラムについて，アルゴリズムや使用するロボット，実験を行うための部屋の状況を変えつつ，動作確認を行えるライブプログラミング環境があれば，実装や実験のためにかかる時間や手間が削減できる．

ロボットを取り巻く周囲の状況の変化の感知と，それに伴う振る舞いの変更の実現に有用なプログラミング技術として，コンテキスト指向プログラミング(Context-Oriented Programming: COP)[1][2][3][4][5]がある．COP では，状況の変化の感知や，状況ごとに異なる振る舞いをモジュール化するための機能を備えており，前述のようなロボットを実現することに向いている．

本研究の目的は，周囲の状況によって振る舞いを変えるロボット開発を対象としたライブプログラミング環境を実現することである．本開発環境では，状況の変化の感知のためのアルゴリズムを実行時に書き換えるために，COP の該当部分をスクリプト言語によって記述可能な仕組みを提供する．また，使用するロボットの変更などにも対処するために，できる限り汎用的に，様々なロボットに対して用いることが可能な形で実現する．

†九州大学 Kyushu University
‡東海大学 Tokai University

多様なロボットに適用するための問題として，以下が挙げられる．(1)ロボットによってプログラミングから実行までの手順が異なる，(2)ロボットごとに通信方法や送受信されるデータのプロトコルが異なる，(3)ロボットによって形状や搭載しているセンサ・アクチュエータが異なる．特に問題(3)は，状況の変化を感知するプログラムにも影響がある．提案開発環境は，これらの問題を解決し，様々なロボットに対して利用可能な機構を実現する．

以降，2 節で COP のライブプログラミング実現について述べ，3 節で多様なロボットへの適用するためのアイデアを紹介する．最後に本稿での提案をまとめ，今後の課題を述べる．

2. COP のライブプログラミング実現

本節では，COP のライブプログラミングを実現するために，2.1 節で COP について概説し，2.2 節でライブプログラミングの対象とする部分について述べる．

2.1. コンテキスト指向プログラミングの概要

COP は，コンテキストに依存したソフトウェアを開発するためのプログラミング技術であり，本稿冒頭で述べた掃除機ロボットのような，実行時の状況によって振る舞いを変えるようなシステムを実現する技術として注目されている．ここでいうコンテキストとは，システムの振る舞いを変えうる要因であり，プログラムから観測可能なものである[1]．コンテキストとして挙げられるものは，システムを取り巻く外部環境や，システムの内部状態，あるいはそれらの変化の順序が挙げられる．COP はコンテキストに依存した振る舞いをレイヤとしてモジュール化し，それらを実行時のコンテキストの変化に応じてアクティベートする機構を備えている．これにより，COP はコンテキストに応じて全体の振る舞いを変えるようなソフ

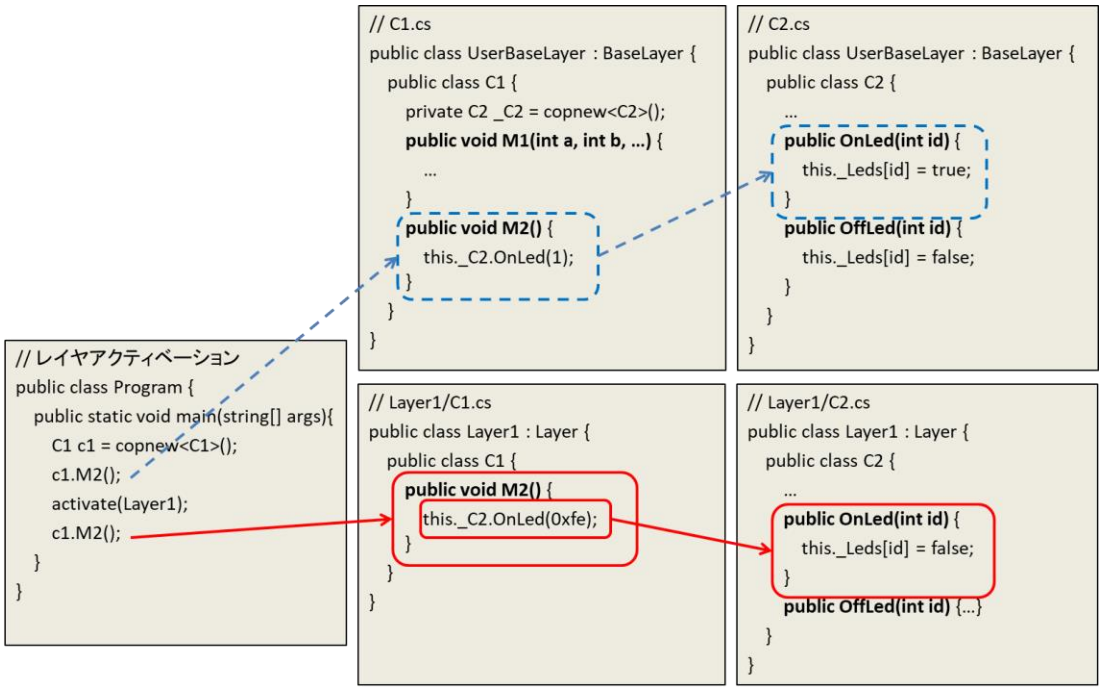

図 1. COP のプログラム例

トウェアの開発を容易にする.

コンテキスト指向プログラミングの例として，図 1 に我々が以前に提案した C#を対象とした COP のプログラム例を示す[6]. 例では，図の上のプログラムで C1 というクラスを定義し，下の図で Layer1 というレイヤがアクティブにされたときの C1 クラスの振る舞いを記述している. プログラム中でアクティベート命令を実行すると，図に示す通り，M2 メソッドのディスパッチ先が変わる. これらの仕組みは，C#のリフレクション機能を用いて実現している. 提案環境では，図で示すような，COP プログラム記述と実行をサポートする.

2.2. COP のライブプログラミング

本節では，COP に対し，どのようにライブプログラミングを適用するかについて述べる. 本稿冒頭で述べた通り，本開発環境は，ロボットが周囲の状況の変化を感知するためのプログラムを実行中に書き換えながら，動作確認を行えるようにする. これにより，上記プログラムの実装や実験のためにかかる時間や手間が削減することを目的としている.

図 2 に我々が想定している COP システムの構成を示す. COP の主な適用対象としては，IoT システムなどが考えられるため，今回の研究では，複数台のロボットが連携するシステムを想定している.

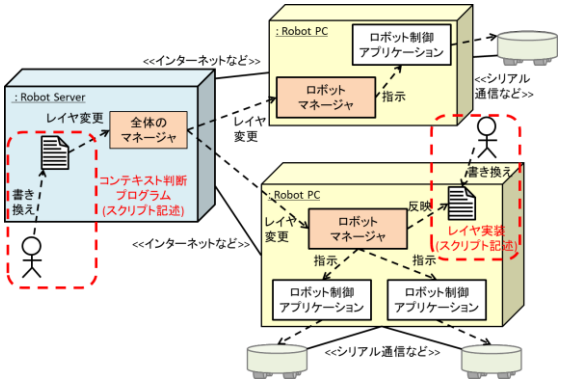

図 2. 想定する COP システムの構成

これらのロボットは，開発用の PC とシリアル通信などを用いて通信され，PC からの指令を受けて動作する. また，各ロボットに指示を出すための PC 上のアプリケーションは，ロボット間の連携のために，ロボットマネージャというアプリケーションと通信する. システムの規模によっては，さらに上位のマネージャが存在することとする.

提案開発環境は，図 2 に示すように，各マネージャで行うコンテキストの判断や，コンテキストごとの振る舞い実行をスクリプトとして分離することで，COP のライブプログラミングをサポートすることを想定している.

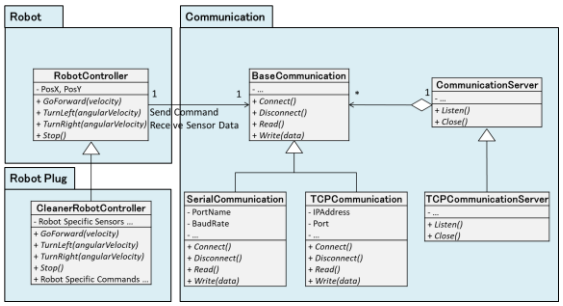

図 3. ロボットとの通信のための構造

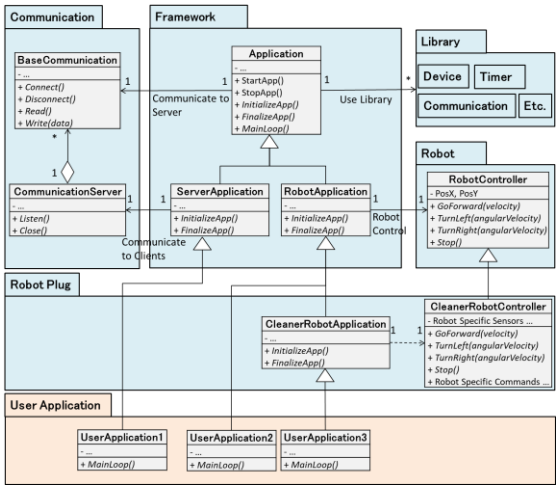

図 4. アプリケーションの構造

3. 多様なロボットへの適用

本節では，冒頭で述べた，提案開発環境を多様なロボットへ適用する際の問題について取り上げ，その解決策を示す．

冒頭で述べた問題は次の三つである．(1)ロボットによってプログラミングから実行までの手順が異なる，(2)ロボットごとに通信方法や送受信されるデータのプロトコルが異なる，(3)ロボットによって形状や搭載しているセンサ・アクチュエータが異なる．

問題(1)に関して，ロボットは大きく分けて次の二種類があり，それぞれで開発の手順が異なる．一つはロボットが基本的な機能を初めから提供しており，PC や他の機器からの通信によって，常に指令を送ることで，ロボットの振る舞いを制御する場合である．この場合，開発者は PC や他の機器からロボットに指令を出すプログラムを書くこととなる．もう一つは，ロボット上で実行されるプログラムをロボットのマイコンに書き込む方法である．

提案開発環境では，前者の方法のロボットを対象とする．理由は，開発に用いるプログラミング言語をロボットに依存させないためである．大抵のロボットのプログラムは C/C++で記述し，プログラム上で用いる API もロボットに依存する．通信による方法では，通信に用いる手段やプロトコルさえ合っていれば，どのようなプログラミング言語からでも，ロボットの動作を制御することができる．

後者のロボットに対して，提案環境を適用する場合は，PC と通信し，PC からの指示によって動作を決定するプログラムをロボットのマイコンに書き込む必要がある．プログラムを書き込む方法はロボットによって異なる．提案開発環境では，この違いを解消するためのユーザインタフェースを提供する．開発者は，このユーザインタフェースからの操作によってロボットのマイコンにプログラムの書き込みを行う C#プログラムを実装することで，異なるロボットに同じ手順でプログラムの書き込みを行えるようにする．

問題(2)に関して，ロボットを通信によって動かす際に，ロボットごとに通信方法や送受信されるデータのプロトコルが異なるため，ロボットごとに通信プログラムが必要となる問題がある．例えば，教育用の掃除機ロボットで代表的なものだと，USB ケーブルで接続し，シリアル通信によって通信を行っているが，教育用の Drone だと，Wi-Fi で UDP によって通信をしている．これらの違いを解消するために，

提案開発環境では，図 3，4 の構造によってこの問題を解決している．提案環境では，シリアル通信やネットワークによる TCP/UDP の通信を同様のメソッドで実行できるように抽象化したクラス群をライブラリとして提供している．また，送受信されるデータのプロトコルの違いを解消するために，ロボットを抽象的に扱うためのクラスも提供しており，それぞれのロボットに対して具体的なクラスを実装することで，異なるロボットに対して同じ命令で動作させることができる仕組みを提供している．アプリケーションはこれらのライブラリを用いることで，図 4 に示す形で実装する．

問題(3)に関して，ロボットには様々な種類があるため，問題(2)の解決策で示したような，抽象化したロボットクラスを作る際の共通した命令は用意しにくい．例えば，Drone と掃除機ロボットでは，飛ぶか飛ばないかという違いがあり，同じ命令でプログラムを記述することは困難である．そのため，抽象化したロボットクラスは，ロボットの種類によって，継承の階層がさらに二，三階層に分かれるようにした．

今回の研究では，二輪走行ロボットのみを対象としているので，二輪走行ロボットを動かすための共通の命令について考える．二輪走行ロボットの移動方法は，直

進や方向転換といったもののほかに，旋回半径を与える方法がある．これはパラメータとして与えられた半径の長さの円に沿うようにロボットを移動させる方法で，比較的簡易な計算でモータに与える速度を決定できるため，よく使われている．その他に，ロボットが初期位置からどれだけ移動したかを示す座標の計算や，ロボットの衝突検知といった機能が考えられる．

これらの機能をいくつかの掃除機ロボットや，それ以外二輪走行ロボットで実装することで，これらのロボットが同様の命令によって動作するようにしている．

4. おわりに

本研究では，教育用ロボットを対象とした，COP のライブプログラミング環境を試作した．本開発環境では，多様なロボットに対して適用できるように，下記の問題点に取り組んだ．(1)ロボットによってプログラミングから実行までの手順が異なる，(2)ロボットごとに通信方法や送受信されるデータのプロトコルが異なる，(3)ロボットによって形状や搭載しているセンサ・アクチュエータが異なる．

提案環境では，これら三つの問題を解決するために，通信やロボットを動かすためのライブラリクラスを抽象的に扱い，容易に拡張できるようにした．また，二輪走行ロボットを対象とするときに必要な最低限の命令の具体例を示し，共通の命令でこれらのロボットを動かす方法を示した．

今後は，提案開発環境を実践的なアプリケーションで実験し，ライブプログラミング化による効果を明らかにする．

参考文献

[1] R. Hirschfeld, P. Costanza and O. Nierstrasz: Context-oriented Programming, Journal of Object Technology, Vol. 7, No. 3, pp. 125-151, (2008).

[2] G. Salvaneschi, C. Ghezzib and M. Pradella: Context-oriented Programming: A Software Engineering Perspective, Journal of Systems and Software, Volume 85, Issue 8, pp. 1801-1817, (2012).

[3] M. Appeltauer, R. Hirschfeld and J. Lincke: Declarative Layer Composition with the JCop Programming Language, Journal of Object Technology, Vol. 12, No. 4, pp. 4:1-37, (2013).

[4] M. Appeltauer, R. Hirschfeld, M. Haupt and H. Masuhara: ContextJ: Context-oriented Programming with Java, In proceedings of the JSSST Annual Conference 2009, pp. 1-15, (2009).

[5] P. Costanza and R. Hirschfeld: Language Constructs for Context-oriented Programming: An Overview of ContextL. In DLS ’05: proceedings of the 2005 symposium on Dynamic languages, pp. 1-10, (2005).

[6] 谷川郁太，久住憲嗣，小倉信彦，菅谷みどり，渡辺晴美，福田晃: コンテキスト指向プログラミングにおける優先度に応じたレイヤスケジューリング手法の提案，組込みシステムシンポジウム 2016 論文集, pp.47-54, (2016).