Vol.2017-ARC-225 No.40
Vol.2017-SLDM-179 No.40
Vol.2017-EMB-44 No.40
2017/3/10

コンテキスト指向ソフトウェア評価アプリケーションの提案

上條弘貴[†1]　大塚崇弘[†2]　新井達也[†2]　谷川郁太[†3]　渡辺晴美[†2]

概要：本稿では，コンテキスト指向ソフトウェアの評価アプリケーションについて提案する．コンテキスト指向プログラミング(Context-Oriented Programming: COP)は，コンテキストに依存した振る舞いをモジュール化し， 実行時にコンテキストに応じて動的に変更するためのプログラミング技術である．現在 COP は先進技術として注目されており，様々な事例が存在するが，これらの事例の目的は，その提案方法を評価することである．本稿はロボット開発により，オペレーティングシステム(OS)における 5 人の哲学者問題のような COP の現実的な問題を見つけるために, 我々は単純な協調ロボットシステムアプリケーションを提案する．

キーワード：コンテキスト指向技術，レイヤ，協調ロボットシステム

1. はじめに

IoT，Industry 4.0，Indus-try Internet の出現により，環境適応型の開発技術の重要性が高まっている．このような高度な技術では，次世代ロボットが複数のサービスをサポートし，周辺環境に応じて最適なサービスを提供することが求められる．環境適応型の開発技術は、これらのロボットサービスのメカニズムを実現するための重要な機能である．一方，これらの機能は本質的に複雑であるため，深刻な問題を引き起こす可能性がある．コンテキスト指向プログラミング（Context-Oriented Programming: COP）言語[1] [4] [5]は，それらの機能を実現するための適切なモジュール性を提供する。コンテキスト指向プログラミング言語は、環境の変化などの文脈（コンテキスト）に応じて、ソフトウェアを再構築し振る舞いを変更する。様々な COP が提案されている[1][3-15]。我々は、クラス群を内包するレイヤ単位で実行時書き換えを行う COP に着目している．しかし，COP でプログラム構築を行っても，その行動を十分に予測することは困難である．オペレーティングシステム（OS）の考え方は, この問題を克服するのに適切であると考える．OS において，研究者は、5 人の哲学者の問題[2]のような共通の問題に基づいて通信アルゴリズムを提案している．5 人の哲学者の問題のような COP の典型的な問題を見つけるために，我々は単純な協調ロボットシステムアプリケーションを提案する．このアプリケーションでは，複数のロボットを制御するための COP の 1 つである RT-COS [3]にプログラムを記述し，複数の異なる環境でロボットを動作させる．

COP に関連する多くの研究は，スマートフォンや小型ロボットなどのさまざまなアプリケーションに適用されている．しかし，これらのアプリケーションの目的は、その提案方法を評価することである．一方、我々のアプリケーションの目的は，COP に関連する問題を議論することである．さらに、同一ユーザの COP プログラムは，現実世界でロボットを駆動する．実際の世界では，ロボットの物理マシンに依存した問題を見つけることが予想される．物理マシン依存の問題は実用的な次世代ロボットを構築するためのソフトウェアの現実的な問題を想像するのに貢献すると期待している．以降，本稿は次のように構成される．第 2 節では問題点を明らかにする。第 3 節では、IoT ロボットアプリケーションを紹介する。第 4 節では、アプリケーション上の COP の問題について議論する。最後に、第 5 節で本稿での提案をまとめ，今後の課題を述べる．

2. 課題

本節では，提案アプリケーションを構築するための課題について記す．本アプリケーションの最終目標は，IoT ロボットを開発するための COP に関連する問題を見つけることである．本稿では，COP の問題を発見するためのアプリケーションの以下の課題を想定している．

(1) IoT
- (a) ロボットはインターネットに接続されている．
- (b) インターネット上には，クラウドとしての制御サーバが存在する．
- (c) 複数のロボットは協調動作を行う．

(2) COP
- (a) COP プログラムはロボットを制御する．
- (b) ロボットは環境に依存した複数の振舞いを持つ．

(3) より実用的
- (a) マシン依存機能を実現する．
- (b) IoT の適切なスケーラビリティを表す．

3. 評価アプリケーション

本節では，提案する評価アプリケーションについて，4.1 節で提案アプリケーションの概要及び構成をそれぞれ記し，4.2 節で提案アプリケーション実現のための機材及び機能

†1 東海大学大学院情報通信学研究科
Tokai University School of Information and Telecommunication Engineering
†2 東海大学情報通信学部組込みソフトウェア工学科
Tokai University School of Information and Telecommunication Engineering Department of Embedded Technology
†3 九州大学
Kyushu University

Vol.2017-ARC-225 No.40
Vol.2017-SLDM-179 No.40
Vol.2017-EMB-44 No.40
2017/3/10

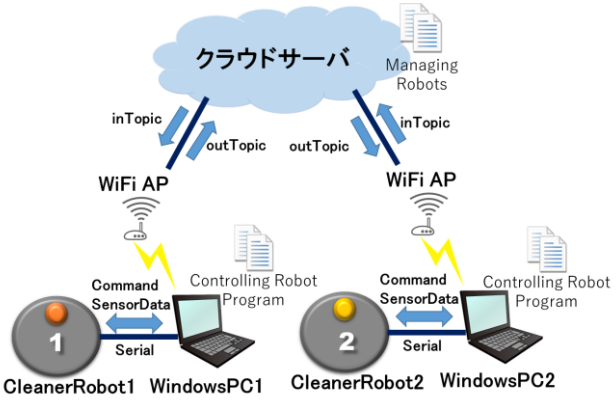

図 1　アプリケーションの構成図

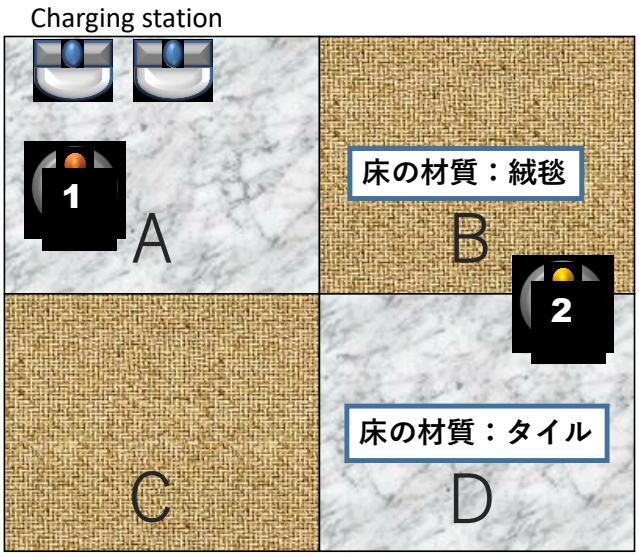

図 2　ロボット動作環境例

の実現方法について記す．以降では提案アプリケーションのレイヤとクラス構成，COP 言語でのプログラム例についてそれぞれ説明する．

3.1 概要・構成

提案アプリケーションの構成図，ロボット動作環境例を図 1，図 2 にそれぞれ示す．本システムはクラウドサーバに接続されている各掃除機ロボットが掃除環境の異なる複数のエリアを最適に掃除するように管理する．提案アプリケーションでは，2 台の掃除機ロボットと 4 つのエリアを想定している．各エリア内でのロボットの振る舞いとして，通常モード，集中モードの 2 つの機能を持つ．通常モード時，2 台の掃除機ロボットは異なるエリアを掃除し，すべてのエリアの掃除を効率的に行うよう動作する．掃除エリアのゴミの量が多い場合などで掃除に時間を要しているエリアには，もう 1 台の掃除機ロボットをそのエリアに向かわせ 2 台でエリアの掃除を行いなるべく早くエリアの掃除が完了するよう動作する．掃除エリアは，それぞれ A，B，C，D の 4 つのエリアで構成され，A，D エリア，B，C エリアでそれぞれ床の材質が異なる．そのため掃除機ロボットの動作アルゴリズム，各パラメータを床の材質に応じて変化させる必要性がある．またエリア A には，掃除機ロボットの充電ステーションが設置されている．従って，いずれの掃除機ロボットもエリア A から動作を開始する．

3.2 アプリケーションの実現方法

表 1 にモード，床の材質の組み合わせにおけるロボットの機能概要表を示す．実現に際し，4.1 節で示したカーペット，タイルでの掃除方法の違いについてはロボットに搭載されている掃除用ブラシ，吸引機の切り替えによって掃除方法の違いを表現する．コンテキストの判定についてはロボットに搭載されている赤外線センサの値から，地面の色によってカーペット，タイルを判断するものとして，機能を切り替える．ゴミの有無に関してはゴミ検知センサを使用し判断する．ロボットへの通信はシリアル通信を用いる．本システムを iRobot 社が製造，販売する自動掃除機 iRobot Create2 に適用する．本アプリケーションを C#ベースの COP 言語である RT-COS で構築し，COP の問題について議論する．

3.3 レイヤ

掃除機ロボットの制御プログラムのレイヤ図について，図 3 に示す．本アプリケーションはベースとなるレイヤと各モード，環境に依存した以下の 4 つのレイヤの組み合わせで表現される．

(1) ベースレイヤ:ロボットを動作させるための基本的なクラスとプロセスを持つ．
(2) ノーマルレイヤ:ノーマルモード時にアクティベートするレイヤ．ノーマルモード時のロボットの各パラメータ，ロボットの振る舞いを持つ．ベースレイヤ:ロボットを動作させるための基本的なクラスとプロセスを持つ．
(3) ダートレイヤ:ダートモード時にアクティベートするレイヤ．ダートモード時のロボットの各パラメータ，ロボットの振る舞いを持つ．
(4) タイルレイヤ:エリア A，D を掃除時にアクティベートするレイヤ．タイルエリアを掃除するための制御対象，床の材質の変化に対応するため，固有のエンコーダ値補正のクラス，プロセスを持つ．
(5) カーペットレイヤ:エリア B，C を掃除時にアクティベートするレイヤ．カーペットエリアを掃除するための制御対象，床の材質の変化に対応するため，固有のエンコーダ値補正のクラス，プロセスを持つ．

3.4 プログラム例

RT-COS を用いた本アプリケーションのプログラム例について図 4 に示す．RT-COS は C#をベースとした COP 言語である．RT-COS の特徴の一つとして，レイヤアクティベーションに関する処理をアプリケーション実現の為のプログラムすなわちレイヤからオブザーバーアスペクトと呼ばれるモジュールに分離している．これにより，レイヤやレ

Vol.2017-ARC-225 No.40
Vol.2017-SLDM-179 No.40
Vol.2017-EMB-44 No.40
2017/3/10

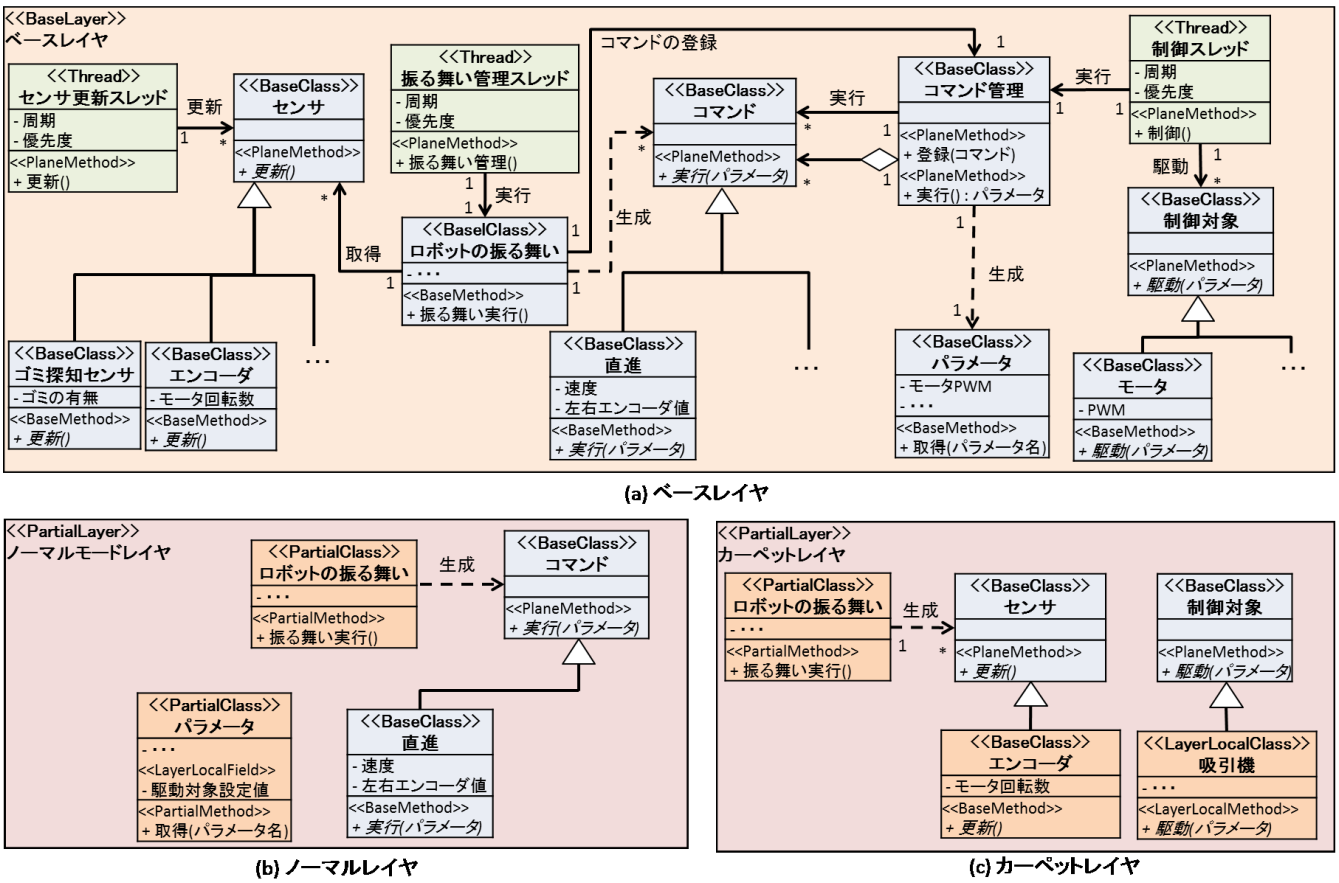

図 1　プログラムのレイヤとクラス構成

イヤアクティベーションに対して，再利用や変更を容易に行えるという点が挙げられる．図 4 の例では 4.3 節で示したベースレイヤ，カーペットレイヤ，ノーマルレイヤについて，機能及び環境に依存した処理をそれぞれ(ⅰ)，(ⅱ)，(ⅲ)に記述する，(ⅳ)についてはノーマルレイヤのオブザーバーアスペクトとしてレイヤ切り替えの条件，タイミングを記述する．

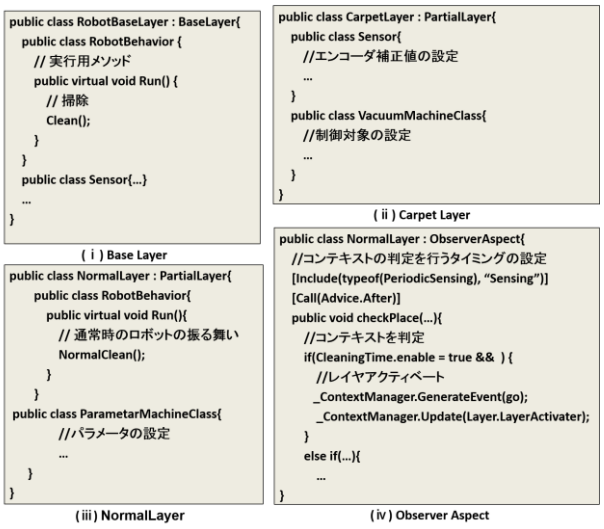

図 2　RT-COS でのプログラム例

4. 議論

本節では，アプリケーションが COP 問題を見つけることができるかどうかについて説明する．以降では，アプリケーション上の小さなシナリオでの COP 問題を考察する．

4.1 シナリオ

COP 問題を見つけるためのシナリオとして，以下のシナリオを作成する．

シナリオ：ロボット A がフロアレイヤからカーペットレイヤへアクティベート中、ロボット B がダートモードに入りもうロボット A を呼び出した．

4.2 COP 問題

上記シナリオ(2)を使用して，コンテキスト指向プログラミングに関連する問題について議論する．このシナリオでは，プログラムを適切に実装していないと，ロボット A はロボット B の掃除エリアに呼び出されず，現在の掃除エリアの掃除を行う．この問題を解決するために，各レイヤは優先度に応じてレイヤアクティベーションの横取りを行うためのメカニズムを持つ必要がある．

この問題はプロセス間通信で解決するよう思える．もちろん，プロセス間通信アルゴリズムとプログラム実装方法の工夫だけで問題を解決することが可能である．ただし，レイヤの変更による影響はプロセスとは異なる．プロセス間通信はプロセスのために機能する．一方，この問題はシ

ステム全体を変更するレイヤに焦点を当てている. さらに, COP 言語がこの問題をサポートする言語構造を持つのならば, 次のようないくつかの利点が期待できる.

(a) 開発者はCOP問題が存在することを意識することができる.

(b) 問題解決のためのメカニックの効率的なアルゴリズムを考えることが出来る.

したがって, 我々はアプリケーション上で COP 問題を発見したと考える.

5. 結論

本稿では、コンテキスト指向プログラミング言語を議論するための IoT ロボットアプリケーションの提案をおこない, COP 問題を発見するための環境を提案した. 議論では、小さなシナリオで COP 問題を検討した。 そこで、次世代ロボットに関連した COP 問題を見通すことができた. 今後の研究では、同一の COP プログラムで仮想地図と現実世界の両方で協調ロボットを駆動することが可能なシミュレータを構築し、より多くの COP 問題を発見していきたい.

参考文献

[1] G. Salvaneschia, C. Ghezzia, M. Pradellaa: Context-oriented Programming: A Software Engineering Perspective, Journal of Systems and Software archive, Vol.85 Issue 8, pp. 1801-1817,2012.

[2] A. S. Tanenbaum, H. Bos: Modern Operating Systems (4th edition), Pearson, 2014.

[3] Tanigawa, I., Ogura, N., Sugaya, M., Watanabe, H. and Hisazumi, K. : A Structure of A C# Framework Con-textCS based on Context-Oriented Programming, MOD-ULARITY Companion'15, pp. 21{22 (2015).

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

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

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

[7] Appeltauer, M., Hirschfeld,R., Haupt, M., Lincke, J. and Perscheid, M. : A Comparison of Context-oriented Programming Languages, In Proceedings of the Workshop on Context-oriented Programming (COP) 2009, ECOOP 2009, pp. 1-6, (2009).

[8] Costanza, P. and Hirschfeld R. : 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).

[9] Hirschfeld, R. Costanza, P. and Haupt M. : An Introduction to Context-Oriented Programming with ContextS, In Generative and Transformational Techniques in Software Engineering (GTTSE) II, Springer LNCS 5235, pp. 396-407, (2008).

[10] Schmidt, G. : ContextR & ContextWiki. Master's thesis Hasso-Plattner-Institut, Potsdam, (2008).

[11] Lincke, J., Appeltauer, M., Steinert, B. and Hirschfeld R.: An Open Implementation for Context-oriented Layer Composition in ContextJS. In Elsevier Journal on Science of Computer Programming, Special Issue on Software Evolution, (2011).

[12] Schubert, C. : ContextPy & PyDCL - Dynamic Contract Layers for Python, Master's thesis, Hasso-PlattnerInstitut, Potsdam, (2008).

[13] von Lowis, M., Denker, M. and Nierstrasz, O. : Contextoriented Programming: Beyond Layers, In ICDL '07: Proceedings of the 2007 international conference on Dynamic languages, volume 286 of ACM International Conference Proceeding Series, pp. 143-156, (2007).

[14] Appeltauer, M., Hirschfeld, R. and Rho T. : Dedicated Programming Support for Context-aware Ubiquitous Applications, In Proceedings of the 2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM) 2008, pp. 38-43, (2008).

[15] 青谷知幸, 紙名哲生, 増原英彦: オブジェクト毎の層遷移を宣言的に記述できる文脈指向言語 EventCJ, コンピュータソフトウェア, Vol. 30, No.3, pp. 130-147, (2013).