コンテキスト指向プログラミング言語のためのコンテキスト生成ツールの提案

茂木 康太郎† 渡辺 晴美†

概要 ：本論文では，環境の変化の想定を支援するために，コンテキスト指向言語(COP)に基づいたコンテキスト生成ツールについて提案する. COP は様々な環境の変化に応じた振る舞いを扱う言語として知られている。環境に応じた適切な振る舞いを行うためには，環境の変化を想定し，それに対応したプログラムを用意する必要がある. しかし，外乱や予想外の変化など，環境の変化を全て想定するのは難しい. 本稿では，ロボットを用いた環境の変化を計測し、その結果をコード先生に反映可能なコンテキスト生成ツールを提案する.

Propose Context Generation Tool for the Context-Oriented Programming Language

Kotaro Mogi† Harumi Watanabe†

Abstract: This article proposes a context-layer generation tool based on Context-Oriented Programming language (COP). COP is well known for dealing with behavior to correspond to changing surrounding environments (contexts). To build a program for such contexts, we need to foresee the changing surrounding environments. However, we cannot easily foresee contexts because of noise or disturbance. In this article, we introduce a code generation tool for COP with reflecting the result of measuring the surrounding environments by using robots.

1. はじめに

近年ロボット技術が盛んになり、様々な場所でロボットが使われている. 災害現場で活動する救助ロボットやショッピングモールなどで案内を行うコミュニケーションロボットなどが例として挙げられる. このようなロボットは環境や状態の変化に対応し，適切な振る舞いを行うことが求められる. 様々な環境の変化すなわちコンテキスト(文脈)に応じた振る舞いを表現できるプログラミング言語としてコンテキスト指向プログラミング言語(Context-Oriented Programming：COP)が存在する. この言語の特徴は，レイヤと呼ばれるプログラム群を切り替えることにより，環境の変化を表現することである[1][2][3]. 近年，様々な研究者によって研究が行われており[4][5]，ロボットなどの組込みシステムへの適用の検討も行われている[6]. コンテキストをモデリングできる図には、UML のコンテキスト図や、Kyo C. Kang のコンテキストモデル[7]が知られており、我々のアプローチと類似した方法に、[8]があるが、これらは、COP とは関連づいていない。COP と関連付いたモデルとして、小川らの研究[9][10]、Nicolás Cardozo らの研究[11]がある。これらはコンテキストの競合の発見を目的としてる。[11]は競合について学習することも可能である。一方、モデルから COP を生成する研究は行われていない。プログラムを生成するにあたり、我々は、コンテキストすなわち環境の変化を識別する部分について着目する。

環境の変化に応じた適切な振る舞いに変化させるにはあらかじ環境の変化を想定し，コンテキストを用意しておくことが必要となる. しかし，外乱などにより想定外な環境の変化が増えている場合や，特にレスキュー

†東海大学大学院情報通信学研究科情報通信学専攻
Tokai University School of Information and Telecommunication Engineering

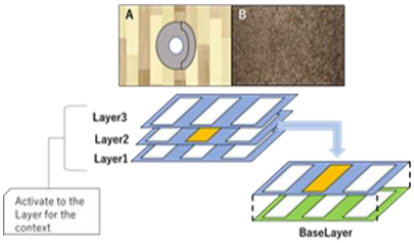

図 1 COP 概要

ロボットのような人間が立ち入ることのできない場所で活動するロボットの場合，環境の変化を想定することすら難しい場所で活動する場合など，事前に組み込まれているコンテキストだけでは十分に振る舞いの変更が行えないことが考えられる．このような問題に対し，ロボットが自身の活動する環境の変化を認識し，コンテキスト，レイヤの生成を支援することができれば解決できると考える．そこで，本稿では，ロボットを用いた，コンテキストの生成とコンテキストの変更条件の記録を行うコンテキスト生成ツールを提案する．以下，2 章では COP の概要について説明する．3 章ではコンテキスト生成ツールの提案を行う． 4 章ではシステムの検討を行い，5 章では本稿のまとめと今後について述べる．

2. COP 概要

本章ではコンテキスト指向プログラミング言語(COP)の概要について述べる．COP は，プログラム実行時の外部環境に応じてプログラム群を活性化させることでシステムを再構築するメカニズムを有するプログラミング言語である．コンテキストに依存した振る舞いをレイヤという単位でモジュール化し，レイヤを切り替えることでシステムを再構成する．レイヤにはクラス群が記載されており，レイヤが切り替わる際，レイヤに記載されているクラス群を書き換えることで振る舞いを変更する．図 1 はレイヤ切り替えの例である．領域ごとに違う掃除方法が割り当てられたロボットがあるとする．領域 A では拭き掃除，領域 B では掃き掃除を行うものとすると，このロボットが領域 A から領域 B に移動した場合，レイヤは拭き掃除のレイヤから掃き掃除のレイヤに切り替わる．その際，システムは掃き掃除を行うためのクラス群に書き換えられ，システムが再構築される．このように，クラス群を書き換えることで動的にシステムを再構成するのがコンテキスト指向の特徴である．

3. コンテキスト生成ツールの提案

本章ではロボットを用いたコンテキスト生成ツールの提案を行う．コンテキスト生成ツールでは，環境の変化

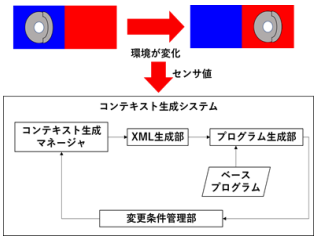

図 2 コンテキスト生成ツール構成

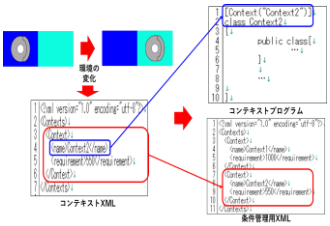

図 3 コンテキスト生成プロセス概要

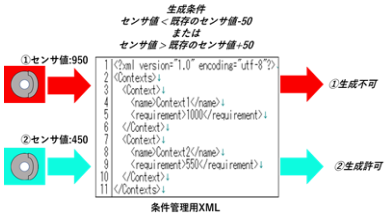

図 4 コンテキスト生成マネージャ動作

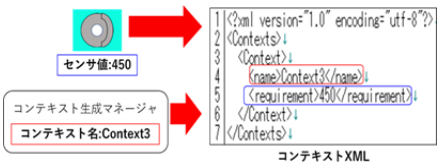

図 5 XML 生成部動作

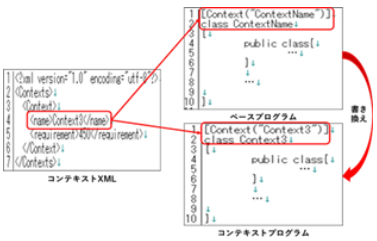

図 6 プログラム生成部動作

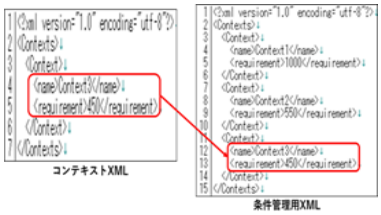

図 7 変更条件管理部動作

を認識し，その環境のロボットのセンサの値からコンテキストを生成し，センサの値をレイヤの変更条件として記録することでレイヤの生成と切り替えの支援を行う．以下，生成ツールの概要及び各システムについて記す．

3.1. 概要

コンテキスト生成ツールの構成を図 2 に示す．提案するシステムでは，ロボットが持つセンサにより環境の変化を認識し，そのセンサ値を用いてシステムがコンテキストの生成を行う．システムは「コンテキスト生成マネージャ」，「XML 生成部」，「プログラム生成部」，「変更条件管理部」の 4 つで構成される．図 3 にシステムの生成プロセスの概要を示す．このシステムでは環境の変化に応じて XML 生成部で XML 文書を生成し，その XML 文書を基にプログラム生成部でベースとなるプログラムを書き換えることでコンテキストを生成する．コンテキスト生成マネージャではコンテキストの生成を管理し，変更条件管理部では生成されたコンテキストの変更条件を管理する．

3.1.1. コンテキスト生成マネージャ

コンテキスト生成マネージャの動作の例を，図 4 を用いて説明する．コンテキスト生成マネージャはコンテキストの生成に一定の条件を設けることでコンテキストの生成を管理する役割を持つ．あらかじめコンテキストの生成条件を設定し，環境の変化を認識した際に条件管理用 XML を参照する．これにより，これまでに生成されたコンテキストの変更条件の中で設定した条件に合致する変更条件が存在するかを確認する．存在する場合にはコンテキストの生成は行わず，存在しない場合のみプログラム生成マネージャは XML 生成部を起動し，コンテキストの生成を行う．図 4 の場合，生成条件は「新しいセンサ値 < 既存のセンサ値 - 50」または「新しいセンサ値 > 既存のセンサ値 + 50」である．①のセンサ値「950」は既にセンサ値「1000」が存在しており，条件を満たしていないためコンテキストの生成は行わず，②のセンサ値「450」は一つ目の条件を満たしているためコンテキストの生成を行う．

3.1.2. XML 生成部

XML 生成部の動作の例を，図 5 を用いて説明する．XML 生成部では，プログラムの生成及び変更条件の管理に用いるコンテキスト XML を生成する．この XML にはコンテキストの名前とレイヤの変更条件を記載する．図 5 ではセンサ値を変更条件としてコンテキスト XML を生成する．コンテキスト XML は全体のタグとして<Contexts>タグ一つ，その中の一つ一つのコンテキストを表す<Context>タグ一つ，更に<Context>タグ内の詳細な部分を表す<name>タグと<requirement>タグ各一つずつの計四つのタグで構成される．<name>タグにはコンテキストプログラムを生成した際に書き換えるためのクラス名が，<requirement>タグには変更条件となるセンサ値をそれぞれ格納する．図 5 の場合，<name>タグには既存のコンテキストとして既に「Context1」と「Context2」が存在していたため，コンテキスト名として「Cotext3」を，<requirement>タグはセンサ値から「450」を格納する．

3.1.3. プログラム生成部

プログラム生成部について，図 6 の例を用いて説明する．プログラム生成部ではコンテキスト XML を基にコンテキストプログラムを生成する．あらかじめ用意されたベースとなるプログラムの一部を書き換えることでプログラムを生成する．図 6 では，あらかじめ用意したベースプログラムの一部をコンテキスト XML の<name>タグの内容に書き換えることでコンテキストプログラムを生成する．ベースプログラムにある「ContextName」の部分をコンテキスト XML の<name>タグにある「Context3」に書き換えることでコンテキストプログラムを生成する．プログラム生成後，新しいコンテキストプログラムが生成されたことを記録するために変更条件管理部を起動する．

3.1.4. 変更条件管理部

変更条件管理部の動作の例を，図 7 を用いて説明する．変更条件管理部では生成されたコンテキストの変更条件を条件管理用 XML にまとめることで管理を行う．この XML を生成することにより，過剰なプログラムの生成を抑制する．生成されたコンテキストの変更条件を生成に使用されたコンテキスト XML に記載された<name>タグと<requirement> タグを 条件管理用 XML にまとめることで管理する．条件管理用 XML の構成はコンテキスト XML と同じであるが，複数のコンテキストの変更条件をまとめて管理するため，複数個の<Context>タグを持つ．図 7 の場合，図 4 で用いられている条件管理 XML に図 6 で生成に用いたコンテキスト XML のデータを追加する．これにより，次にコンテキスト生成マネージャが起動した際には「Context3」までの変更条件が記載された条件管理用 XML を用いることができ，過剰なコンテキストの生成を抑制することができる．

4. 議論

本章ではシステムにより環境の変化を認識し，コンテキストを生成するまでをシナリオを作成しシステムを考察する．シナリオは以下の通りである．

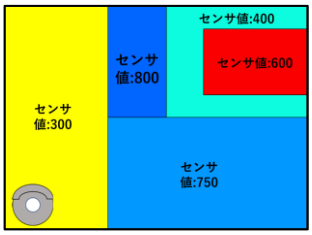

図 8 シナリオ動作環境

シナリオ ： 図 8 に示す環境からコンテキストを生成する．ロボットは初期地点から右回りに環境を一周し，コンテキストを生成する．生成条件には図 4 で用いたものを使用する．

動作の初め，ロボットが現在存在する環境としてセンサ値「300」のコンテキスト「Context1」を生成する．その後，センサ値「800」の領域に環境は変化し，「Context2」が生成される．このように，最初の地点に戻るまでコンテキストの生成が行われる．シナリオの動作終了後のコンテキスト生成マネージャの記録を表 1 に示す．前半の環境の変化ではコンテキストが生成されていることが確認できる．また，二回通る領域に対しては，二回目にはコンテキストの生成を行わないようになっていることが確認できる．これにより，未知の環境に対するコンテキストの生成に貢献できたと考える．しかし，センサ値「750」の領域については生成条件に抵触するセンサ値「800」の領域のコンテキストが生成されているためコンテキストの生成が行われていないことがわかる．このように，このシステムでは環境が変化しても生成条件によって生成されないという問題があることが確認できる．解決策としては，センサのみではなくカメラやホイールの回転数などの複数の機器を用いて環境の変化を認識する方法などが考えられる．

5. おわりに

本稿ではコンテキスト指向プログラミング言語のためのコンテキスト生成ツールの提案を行った．環境の変化を認識し，ロボット自身がコンテキストを生成することで，想定の難しい環境に対するコンテキストの生成を支援することに貢献した．しかし，4 章で挙げたような問題など，まだ解決すべき課題は残っている．今後の課題として，実際にロボットに実装し運用することで，システムの更なる問題点を洗い出し，コンテキスト生成ツールの完成を目指していきたい．

表 1 コンテキスト生成マネージャ動作結果

	センサ値	コンテキスト生成可否	コンテキスト名
1	300	可	Context1
2	800	可	Context2
3	400	可	Context3
4	600	可	Context4
5	400	否	
6	750	否	
7	300	否	

参考文献

[1] R. Hirschfeld, P. Costanza and O. Nierstrasz:Contextoriented Programming, Journal of ObjectTechnology, Vol.7, No.3, pp.125-151, 2008.

[2] 紙名哲生: 文脈指向プログラミングの要素技術と展望，コンピュータソフトウェア, Vol.31, No.1, pp.3-13, 2014.

[3] H. Watanabe, M. Sugaya, I. Tanigawa, N. Ogura and K. Hisazumi: A Study of Context-Oriented Programming for Applying to Robot Development, COP'15 Proceeding of the 7th International Workshop on Context-Oriented Programming, Article, No.4, 2015.

[4] G.Salvaneschia, C.Ghezzia, M.Pradella: Context-Oriented Programming: A Software Engineering Perspective, Journal of systems and Software archive, Vol.85 Issue 8, pp.1801-1817, 2012.

[5] I.Tanigawa, N.Ogura, M.Sugaya, H.Watanabe and K.Hisazumi: AStructure of A C# Framework ContextCS based on Context-Oriented Programming, MODULARITY Companion'15, pp.21-22, 2015.

[6] 小野健也，菅谷みどり: コンテキスト指向プログラミング手法におけるロボット事例への適用検討，研究報告システム・アーキテクチャ, No.10, pp.1-6, 2016.

[7] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, A. Spencer Peterson : Feature-Oriented Domain Analysis (FODA) Feasibility Study, CMU/SEI Report Number: CMU/SEI-90-TR-021, 1990.

[8] 杉田達哉，中道上，青山幹雄: 概念コンテキストモデルに基づくコンテキストウェアサービス開発手法の提案，情報処理学会全国大会講演論文集, Vol.73, No.3, pp.301-302.

[9] 小川英理，森谷大輔，渡辺晴美: コンテキスト指向プログラミングのためのレイヤアクティベイトモデルの考察，組込みシステムシンポジウム 2015 論文集, pp.133-134, 2015.

[10] 小川英理，海老原秀亮，茂木康太郎，齋藤祐里，渡辺晴美: レイヤ相互作用問題の協調ロボットシステムへの適用，研究報告組込みシステム, Vol.44, No.38, pp.1-5, 2017.

[11] Nicolás Cardozo, Ivana Dusparic, Jorge H. Castro: Peace COrP: Learning to solve conflicts between contexts, COP' 17 Proceeding of the 9th International Workshop on Context-Oriented Programming Article, No.4, 2017.