令和 6 年度卒業研究発表会（日本大学工学部情報工学科）
A-3-23

CeruleanBlue: ROS2 ノード開発を容易にするドメイン固有言語の提案

CeruleanBlue: A Domain-Specific Language for Simplifying ROS2 Node Development

20216041　小澤　拓人　[関澤研究室]

1　はじめに

近年、オープンキャンパスにおける本研究室の紹介コンテンツの不足が課題となっている。特に初心者にプログラミングへの興味を喚起する取り組みが求められている中、ロボットなどの物を実際に動かすという体験は、参加者の興味を惹きつける有効な手段となりうると考えられる。しかし、ロボットの制御に用いる技術は初心者にとって非常に難解であり、容易に取り組める仕組みの整備が必要不可欠である。

本研究では、ロボットプログラミングに広く利用されているミドルウェアである Robot Operating System 2 (以下、「ROS2」と略記)[1]の Wrapper となる、新たなプログラミング言語を提案する。本取り組みにより、初学者にも扱える簡易な仕組みの提供と、それに伴う ROS2 開発の効率化を目指す。

2　本研究の概要

本研究では、ROS2 の仕組みを用いた容易なロボットプログラミング手段の提供を目指し、ROS2 の Wrapper となる新言語 CeruleanBlue の設計/開発を行った。

ここでは、CeruleanBlue の提供が ROS2 ノード開発にもたらす容易化と効率化の効果について考察し、本言語の持つ将来性について述べる。それに伴い、本言語が ROS2 ノード開発を容易にするための手法を取り上げ、現状得られている効果と課題について注目する。

3　提案手法

本章では、新言語 CeruleanBlue を用いた ROS2 ノード開発の容易化の手法について述べる。

3.1　CeruleanBlue の概要

CeruleanBlue は、従来の ROS2 プログラミングに比べて容易なロボットプログラミングを実現するために、Visual Scripting による効率的で直感的な開発環境の提供を目的としたドメイン固有言語(Domain-Specific Language: 以下「DSL」と略記)である。CB は ROS2 の環境構築、ソースコードの生成、ビルド構成の生成を行い、ROS2 開発の工程を完全に包括した Wrapper である。

CeruleanBlue の主な特徴として、ROS2 のノード開発における特定のタスクに特化しており、C や Java 等の汎用言語ではない。また、CeruleanBlue で記述された処理は、コンパイル後に機械語へ変換されない。これにより、Java や C#のように、VM(Virtual Machine)に対する中間コードを出力する非ネイティブコンパイル言語の特徴も備えている。本言語は ROS2 を VM とし、C++を中間コードとして扱うため、その実行基盤は完全に ROS2 Humble に依存している。

本言語は GUI ベースのブロック型インタフェースを採用することで直感的な操作を可能にし、ブロックの組み合わせから ROS2 プログラムを出力・コンパイルする機能を備える。しかし本稿の執筆段階では、GUI 部分は設計のみに留まっている。そのため、本研究ではその前身となるテキストベースのプログラミング言語である CeruleanBlue Text(以下、「cbt」と略記)」を開発する。

3.2　CeruleanBlue の入力と出力

CeruleanBlue 内部のコンパイラは、ブロック型インタフェースと 1 対 1 で対応する cbt で記述された ROS2 ノードを入力とし、入力を C++で記述された ROS2 ノードのソースコードに変換して出力を行う。また、コンパイルした ROS2 ノードの cbt プログラムに基づき、実行ファイル名や依存ライブラリなどの情報を集積し、ROS2 ワークスペースのビルドに必要なファイルの自動生成も行う。これに伴う情報の集積ファイルとして JSON 形式でのファイル出力も同時に行うため、最終的な CeruleanBlue の入出力は図 1 に示す構造となる。

3.3　cbt プログラムのフォーマット

cbt プログラムは、C++の ROS2 ノードを大きく抽象化した構造を持つ。図 2 は、Hello, World を 1 秒間隔でパブリッシュし、その内容を出力する cbt プログラムの一部である。

cbt プログラムは、Node や Functions、Publish の「ブロック」と、name や value、level の「属性」の組み合わせで構成される。

ブロックは C や C++における「関数」に相当し、パブリッシュやサブスクライブ、ログ出力、変数定義に対応する機能を持つ。ブロックは、機能のもつ情報を修飾する属性を 1 つ以上持ち、その属性は関数内の「文」や「式」に相当する。属性はそのブロックの持つ出力内容や四則演算の内容など、機能のディテールを定義する。

そのため、cbt で記述された ROS2 ノードは、属性によって修飾されたブロックを集めることで成り立っているといえる。

3.4　提案手法によって実現できている機能

3.4 章では、次の 3.4.1, 3.4.2 章に分け、提案手法によって実現できている ROS2 の機能について述べる。

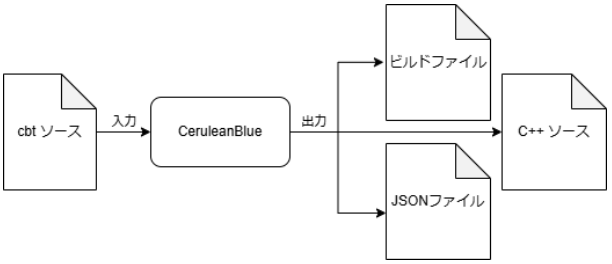

図 1. CeruleanBlue の入力と出力

```
Node <Publisher> : {
  ...
  Functions<interval : 1000> : {
    Output : {
      level : info;
      contents: “Publishing Hello, World!:“use @int_var;
    }
    Publish : {
      contents : “Hello, World!: “ use @example_int;
    }
  ...
  }
}
```

図 2. Hello, World をパブリッシュする cbt プログラム

3.4.1 C++への変換、環境構築の自動化

CeruleanBlue は、cbt ソースファイルを C++で記述された ROS2 ノードへ変換する。具体的には、cbt ファイルを解析して得られた抽象構文木(AST)から特定の要素を抽出し、各要素によって C++のテンプレートの整形を行い、ファイルを生成している。

また、ビルドファイルの生成と C++の include 文生成による、ライブラリの依存関係解決を行う機能を実現している。その他、自動的な環境構築を行う機能を備えている。

3.4.2 Publisher-Subscriber 間の一部トピック通信

cbt では、Publisher、Subscriber の ROS2 ノードをそれぞれ作成し、std_msgs/msg/String 型トピックと geometry_msgs/msg/Twist 型トピックによる通信を実現している。String 型トピックでは、文字列・変数を用いた任意のメッセージの出力、Twist 型トピックでは、並進速度成分、角速度成分のデータの送受信が行える。

また、Twist 型トピックによる、シミュレータ上の差動二輪ロボットへの移動命令を発行し、動作を可能としている。また、秒数や変数の状態による分岐命令を含む複雑な移動命令の実行も可能である。なお、実機に対しての検証は。また、sensor_msgs/msg/LaserScan 型等、いくつかのトピックのサブスクライブは実現できていないため、センサを利用した自律走行などの処理の実現には至っていない。

4　提案手法の評価

本章では、3 章で述べた提案手法による容易化・効率化の効果と、CeruleanBlue の抽象度、提案手法の有用性について述べる。

4.1　容易化・効率化の効果

CeruleanBlue は、次に示す二つの観点より、容易化・効率化に成功していると考える。

最初の観点は、ROS2 プログラムとしてみたときの構造の明確さである。提案手法の cbt は従来のソースコードが持つ構造を抽象化しており、ノード、トピック、関数といった構造が明示的に示されている。ROS2 の概念を学んだ初心者にとって理解しやすい構文を持っており、容易化を達成していると考える。

二つ目の観点は、ROS2 ノード開発における効率である。本言語は、ROS2 ノード開発で必要な、定型的で煩雑なコードの削減を行うことができる。また、定義ベースのコード生成を行うため、手作業によるタイポや、構文エラーによるデバッグ作業の発生を防ぐことができる。その他、前述の環境構築の自動化なども相まって、単純な作業時間と手間の短縮という点で効率化に貢献していると考える。

4.2　C++に対する cbt の抽象度

cbt と C++に対して、代入や引数、コンストラクタ、ブロック宣言のようなプログラムを構成するパーツ(構文要素)の数を比較した。その結果、cbt は C++に比べて 69%少ない構文要素数でプログラムを表現していることが明らかになった。cbt は高レベルの処理を少ない要素数で表現できるため、低レベル処理を柔軟に操作する C++に比べて大きく抽象化されていると考える。

4.3　提案手法の有用性と課題

提案手法について、4.1 章で述べた容易化・効率化の効果を受け、その有用性は非常に大きいと考える。しかし、現在はまだ開発の初期であり、実用化には課題が残る。

現状の本言語が持つ有用性は、抽象化された比較的容易なプログラム構造と、効率化の度合いにある。後述の Visual Scripting の実装により本言語の有用性は飛躍的に増大するが、現状でも抽象化されたプログラム構造が持つ利点は大きい。

大きな利点として、抽象化によって「ノードの機能部分のみに集中できる」ことがあり、ROS2 ノードの設計が視覚的に明確になる。シンプルな DSL として設計をそのまま落とし込める可視化力の高さにより、開発現場での効率上昇や、教育現場での構造化指向の訓練に役立つ効果が見込め、非常に有用であるといえる。

これに対し、cbt には多くの課題も存在する。cbt は ROS2 ノードを構成するプログラムを抽象化しているが、表現できる範囲に制限が生じることは、本言語が潜在的に持つ課題の一つである。

ROS2 ノードをはじめ、メモリ管理やポインタの扱い方、関数の記法によって、多くのプログラムは性能差が生じることがある。これらはプログラムを構成する言語の持つ柔軟な表現力によって担保されており、大きく抽象化が進んだ本言語では、従来よりも表現の幅が狭まっているといえる。

パフォーマンスのオーバーヘッドも本言語の課題の一つである。自動生成された C++コードは、手書きのコードに比べて冗長になる可能性を含んでいる。特にロボット開発は、計算リソースが限られた環境での実行効率を重視する必要がある。しかし本言語は、冗長なコードによってパフォーマンスの低下を招く危険性を持つため、コード生成プロセスの最適化などを行うことによる危険性の軽減が課題である。

本言語の特徴である「抽象化されたプログラム」というものは大きなメリットとなるが、相応にデメリットも含まれている。これは抽象化と表現力のトレードオフであり、完全に避けることは困難であるが、軽減すべき課題であると考える。

5　結言と今後の発展

本研究では、ROS2 ノード開発を容易にする仕組みとして、新言語 CeruleanBlue を提案した。今後、現時点で実装できていない Visual Scripting を実現することにより、ROS2 開発の更なる容易化と効率化が見込まれる。現在の cbt では確保できていないソフトウェア的なユーザーサポートや、より親しみやすいインタフェースの実装による利点は多い。現状の課題解決に加え、「初学者向け」「開発現場での効率化」のいずれの観点でも、ロボット開発の汎用言語として活用できるポテンシャルを持つたせたい。

参考文献

[1] S.Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, “Robot Operating System 2: Design, architecture, and uses in the wild,” Science Robotics vol.7, no.66, 2022.