Topic通信処理記述の解析による ROSアプリケーションのデータフローの可視化

村田　優斗[1,a)]　石尾　隆[1,b)]　嶋利　一真[1,c)]　松本　健一[1,d)]

概要：組込みソフトウェアの信頼性を向上するためには，第三者によるコードレビューが重要である．コードレビューを支援する技術の 1 つとして，ソフトウェア内部のデータフローの可視化が用いられている．組み込みソフトウェアの一種である ROS アプリケーションでは Topic 通信というモデルでデータフローが扱われており，これを可視化するための既存ツールは動的解析を使用している．そのため，コードレビューだけを行う担当者も，アプリケーションのテスト実行環境を準備する必要がある．本研究では，テスト実行環境なしでもコードレビューを容易に実施できる環境を実現するため，ROS アプリケーションのソースコードおよび設定ファイルに書かれた Topic 通信処理の記述を静的に解析し，データフローの可視化を行うツールを開発した．プロトタイプの出力結果を ROS アプリケーションの開発者およびコードレビューの実務者に提供した結果，好意的な意見を得た．

1. はじめに

近年，自動運転技術や IoT の発達に伴い，組込みソフトウェアの開発が盛んに行われている．組込みソフトウェアにおける欠陥は，人命に関わる事故の発生や経済的な損害など甚大な被害につながることがあるため，その信頼性は非常に重要である [10]．一方で，IoT システムではハードウェアの多様な組み合わせが発生するなど，通常のソフトウェア開発よりも開発中に考慮すべき要素が多いことが指摘されている [7]．

組込みソフトウェアの信頼性を向上させることを目的に，コードレビューが行われる．コードレビューは，ソースコードの記述者とは別のレビュー担当者がソースコードやドキュメントを確認し，記述の誤りや欠落などの問題を発見する作業である [9]．この作業により，記述者本人が認知できていない課題に対して様々な観点から調査，検討を行うことができる．組込みソフトウェアにおいては，開発チームに所属していない第三者によって，コードレビューが行われることがある．しかし，第三者によるコードレビューは組込みソフトウェアに対する知識がないことや予算と時間が限られていることから，一般的に難しいとされている．

自動運転技術などで用いられる組込みソフトウェアの一種，ROS アプリケーションにおいても，信頼性の向上は重要な課題となっている．ROS アプリケーションは，世界で最も使われているロボット開発向けのフレームワークである ROS を用いて開発されたアプリケーションである．これらのアプリケーションでは，Node という単位で機能を記述し，それらの間のデータフローを Topic 通信と呼ばれる Publish/Subscribe モデルの通信方式で接続する．このデータフローの把握がレビューにおいて重要であるため，ROS では標準で搭載されている GUI ツール，rqt_graph を用いた Topic 通信の可視化によって，コードレビューの支援が行われている．ただし，rqt_graph は動的解析ツールであり，コードレビューのみを行う担当者も，アプリケーションのテスト実行環境を準備し，その動作を解析する必要がある．そのため，コードレビューを行う担当者にとって，テスト実行環境の準備には大きなコストがかかる場合がある．例えば，本研究で解析対象とする Autoware を実行するには，16 GB 以上のメモリ，30 GB 以上の SSD，NVDIA GeForce GTX980M 以上の GPU を搭載した Linux PC が推奨環境とされており，さらに Docker など，実行に必要なソフトウェアの設定も必要である．

本研究では，テスト実行環境なしでもコードレビューを容易に実施できる環境を実現するため，ROS アプリケーションのソースコードに書かれた Topic 通信処理の記述を静的に解析し，データフローの可視化を行う手法を提案す

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
[a)] murata.yuto.ms6@is.naist.jp
[b)] ishio@is.naist.jp
[c)] k.shimari@is.naist.jp
[d)] matumoto@is.naist.jp

る．また，提案手法に基づいてツールの開発を行う．Topic 通信は，入出力として使用したいデータの名称（Topic 名）を指定して， Publish あるいは Subscribe の処理を呼び出すことで通信が確立される．呼び出しそのものは，アプリケーションの実行時に動的に決定することが可能であるが，実際にはほとんどのソースファイルで文字列定数が使用されており，無条件に，決まったデータの読み書きを行う．そこで Publish および Subscribe の処理を記述するソースコードのパターンを解析することで，アプリケーションを実行せずとも，Topic 通信のモデルを取り出すことができると考えた．ツールの有用性を評価するための解析対象として，自動運転車の制御やシミュレーションを行う Autoware を選択した．Autoware の解析を行うプロトタイプの出力結果を ROS アプリケーションの開発者およびコードレビューの実務者に提供した結果，好意的な意見を得た．

以降，2 章では ROS やコードレビューの関連研究や本研究で用いる ROS の用語について説明し，3 章では本研究で提案する手法とツールの実装について述べる．そして，4 章では本研究で作成したツールを用いた予備評価の結果についてまとめ，5 章では予備評価を受けての今後のツールの改善方針を説明する．最後に 6 章でまとめを述べる．

2. 背景

2.1 ROS

ROS(Robot Operating System) は，ロボット開発を効率的に行うために提案されたオープンソースのプラットフォームである．この ROS は世界で最も使われているロボット開発向けのフレームワークであり，人工衛星や自動車の開発に用いられている．

オープンソースである ROS の利用者数を正確に測ることは難しいが，ROS を管理している OSRF(Open Source Robotics Foundation) が 2011 年から毎年利用レポートを公開している [4]．2011 年は一部データが報告されていないため，2012 年以降のデータで議論する．

ROS パッケージのダウンロード数の推移を図 1 に示す．ROS パッケージのダウンロード数は 2012 年の 8,139 件から 2021 年の 789,956 件と約 97.1 倍に伸びている．また，ROS の代表的な論文である Quigley らの文献 [8] の引用数の推移を図 2 に示す．引用数は 2012 年の 309 件から 2021 年の 9,260 件と約 30.0 倍に伸びている．これらから ROS の需要が増えていることが分かる．

ROS は Python や C++で実装される Node に関するソースコードと Python や XML で実装される起動時の設定について記述した launch ファイルから構成される．ROS を用いて開発される ROS アプリケーションに関わる用語について述べる [12], [14].

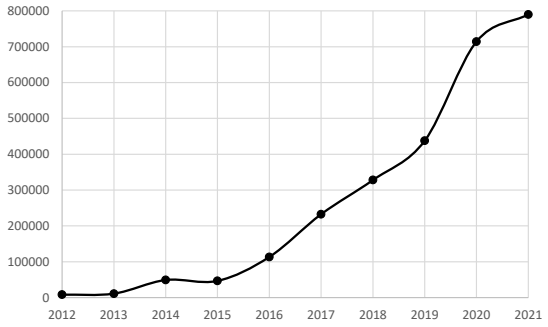

図 1: ROS のダウンロード数の推移

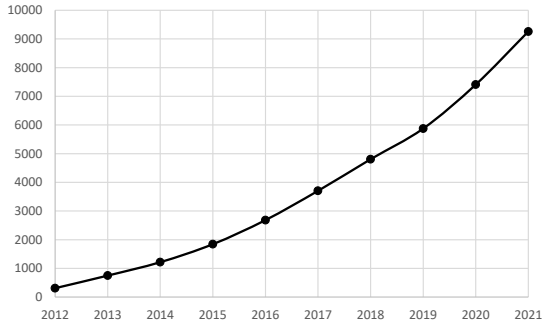

図 2: 文献 [8] の引用数の推移

Node

Node は Linux のプロセスに相当し，それぞれ独立したメモリ空間内でプログラムが動作する．この Node は単純な実行プログラムであり，例えば，センサの数値を読み取る Node，モータの回転数を制御する Node などが含まれる．Node は通信時に識別する必要があるため，各 Node に対して固有の名称が付与される．ROS ではこの Node を組み合わせてソフトウェアを構築する．

Topic 通信

Topic 通信は ROS の Node 間通信のうち最も基本となる Topic を介した Publish/Subscribe モデルの通信方式である．Publish/Subscribe モデルは 1 つのデータを配信する Node に対し複数の Node がデータを購読できる 1 対 N の通信モデルである．これらのうちデータの配信を行う Node を Publisher，データの購読を行う Node を Subscriber と呼ぶ．これらの Publisher/Subscriber をつなぐ Topic はデータのバスのような動きをする．Publisher は Topic にデータを配信し，Subscriber は Topic に対してデータを購読するこ

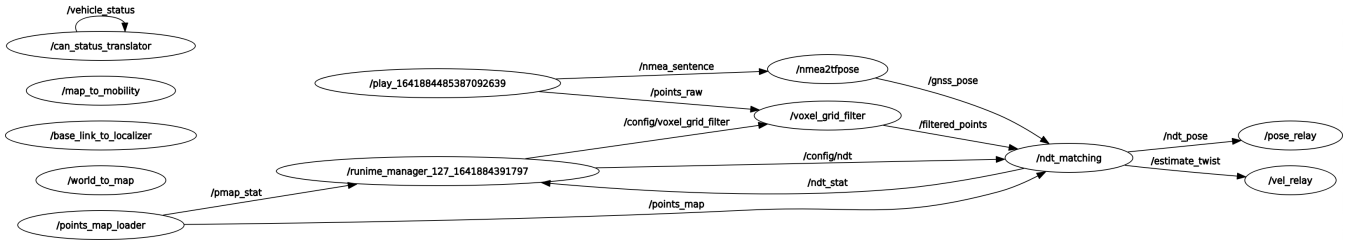

図 3: rqt_graph の出力例

とによって通信を確立させている．

このモデルでは，Publisher はデータを Topic に配信するが，その先の処理については一切考慮していないため，Subscriber が存在しない場合にもエラーにならずに動作する．Subscriber も同様に Topic にデータが存在する場合のみ動作をするため，Publisher が存在しない場合でも動作する．Publish/Subscribe モデルを用いることで処理を Node 単位で分割でき，再利用性，生産性が向上する．このモデルのメリットとしてデータの型を合わせることにより，Node の追加を容易に行える点がある．

Autoware

Autoware [3] は Linux と ROS をベースとした，一般道路での運用を視野に入れた自動運転システム用オープンソースのソフトウェアである．この Autoware は自動運転車の制御とシミュレーションを行うことができる．Autoware のプラットフォーム層には，ハードウェアに CPU・GPU やカメラ・GNSS(Global Navigation Satelite System)・LiDAR などのセンシングデバイスがある．OS には Linux(Ubuntu)，ミドルウェアに ROS を用いており，最上位に Sensing や Localization などのアプリケーションが存在する．LiDAR やカメラ，GNSS などの環境センサを利用して自動車位置や周囲物体を確認しながら，カーナビから与えられたルート上を走行する．Autoware は C++で実装された Node と，Python と XML で実装された launch ファイルで構成されている．

rqt_graph

rqt_graph は ROS に搭載されている Topic 通信を可視化する GUI ツールである．rqt_graph を起動することで，Node，Topic の接続図を出力する．この rqt_graph はソフトウェアを実際に起動する必要があるため，環境が整っている開発者向けの動的解析ツールとなっている．3 つの Node を起動したときの rqt_graph の出力例を図 3 に示す．丸で囲まれているものが Node 名であり，Topic 名は矢印の上に記載されている．Topic 通信に関与していないが起動済みの Node も出力される．図にはユーザが起動した Node 以外の Node 名が出力されており，これらは起動の際に選択した launch ファイルや可視化を行うファイル等である．

2.2 コードレビュー

コードレビューはソフトウェア開発の一環で，コードの記述者ではない別のレビュー担当者がソースコードを閲読し，所感を報告する作業である．このコードレビューは，単純な誤記や根本的な論理の誤り，設計時に策定された仕様や要求の未達成，目的通りに動作はするが将来的にバグにつながりかねない構造の発見などを目的として行われる．

コードレビューの中で，信頼性を高めるために検証と妥当性確認 (V&V) という 2 つの視点から評価を行う手段がある．V&V における検証は，ソースコードが仕様・設計・計画などの要求事項を満たしているかに関する確証のことを指し，妥当性確認はソースコードの機能や性能が本来意図された用途や目的に合っているか，実用上の有効性があるかを指す．

ソフトウェア開発は，ユーザー要求に基づいて仕様書や設計書を作り，それらの文書に従ってコーディングやシステム統合を行う．この工程の中で，設計に従ってコーディングされているか，仕様通りの機能を持っているかという評価が検証となり，ソフトウェア製品がユーザー要求を満たしているかに関する評価が妥当性確認となる．

特に V&V をソフトウェア開発グループから独立した第三者組織の管理に基づいて実施する場合は，ソフトウェア独立検証と妥当性確認 (IV&V) に相当する [5]．IV&V により，客観的な視点からソフトウェアを評価することができる．これはソフトウェアをより安全にするために重要であり，ソフトウェアの信頼性の向上につながる要素となる．

IV&V を用いたコードレビューは，レビュー担当者が事前知識なしにソースコードファイルのセットを調査する必要があるため，一般的に難しいとされている．更に IV&V に利用できる予算と時間は限られているため，効率的なコードレビューを行うことが重要となる [10]．組込みソフトウェア開発においてもこれらのプロセスは行われており，特に宇宙機開発などの高い信頼性が求められる分野において重要な役割を果たしている．[11]．

コードレビューの支援を行う研究として，コード解析支援のために C++で書かれたプログラムファイルに含まれるグローバル変数のモジュール間データフローの可視化

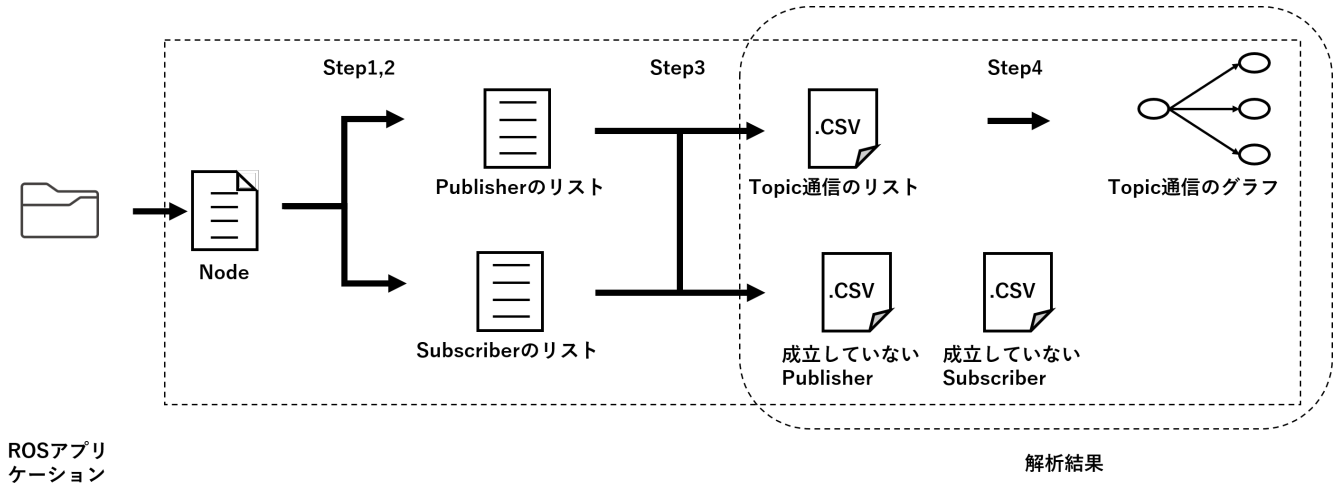

図 4: 提案手法の概要

を行った研究がある [6]．この研究は教育用ロボット作成キットであるレゴマインドストーム [1] に実装されたソースコードを対象として実験を行った．可視化されたデータフローを用いてレビューを行うグループと，ソースコードのみでレビューを行う 2 つのグループに対して，ソースコードの理解度を比較した．結果として，可視化されたデータフローを用いてコードレビューを行う場合はデータフローを用いない場合に比べ，ソースコードの理解度が高いことが分かった．この結果から，データフローの可視化はコードレビューの支援に有効な手段であることが確認された．ただし，レゴマインドストームは教育用ロボット作成キットであるため，実用的なソフトウェアに対して運用可能であるかは明らかになっていない．

3. 提案手法

本研究では ROS アプリケーションを静的に解析する手法の提案を行い，それを実現するツールの開発を行う．具体的な方法としては，ROS アプリケーションにおける Topic 通信を可視化するために命令のパターンを調査し，Publish/Subscribe 命令を正規表現を用いて取得した．

提案手法の概要図は図 4 となっており，これをベースとするツールの開発を行った．ツールへの入力は，解析対象となる ROS アプリケーションのソースファイル集合である．コードレビュー担当者には，レビュー対象としてソフトウェアの一部だけが引き渡されることが多いため，ソースファイルや設定ファイルが一式渡されていても，コンパイルや実行が可能であるとは仮定せず，字句的な解析だけで出力結果を得る設計としている．ツールの出力は，コードレビュー担当者にとって有益な Topic 通信情報である．具体的には，以下の情報を出力する．

Node 間の Topic 通信によるデータフロー
: ある Node n_1 が Publish する Topic t を，別の Node n_2 が Subscribe しているとき，その Topic 通信による接続関係を 3 つ組 (n_1, n_2, t) で表現する．ツールの実装としては，CSV 形式による一覧表として保存するとともに，この通信をグラフとしても可視化する．

通信が成立していない Publisher の一覧
: ある Node n が Topic t を Publish していても，t を Subscribe する Node がアプリケーション内部に存在しないときは，通信が確立しない．これはアプリケーションの実装誤りの可能性があるため，(n, t) の組を孤立した Publisher として報告する．ツールの実装としては，CSV 形式による一覧表として保存を行う．

通信が成立していない Subscriber の一覧
: ある Node n が Topic t を Subscribe しているにも関わらず，t を Publish する Node がアプリケーション内部に存在しないときは，通信が確立しない．これもアプリケーションの実装誤りの可能性があるため，(n, t) の組を孤立した Subscriber として報告する．ツールの実装としては，CSV 形式による一覧表として保存を行う．

ソースコード 1: Publish/Subscribe 命令

```
1 ros::Publisher lanelet_pub = nh.advertise<
      std_msgs::String>("/lanelet_map_bin",1000);
2 ros::Subscriber lanelet_sub = nh.subscribe("/
      lanelet_map_bin",1000,chetterCallback);
```

ソースコード 1 として，実際の Publish/Subscribe 命令の記法の例を示す．1 行目が Publish の命令であり，2 行目が Subscribe の命令である．それぞれ advertise, subscribe という関数呼び出しによって表現されており，それぞれの第 1 引数の文字列が Publish あるいは Subscribe する Topic 名を指定している．1 つ目の命令が Node “lanelet2_map_loader” に記述されており，2 つ目の命令が別の Node “lanelet2_map_visualization” に記述されていた

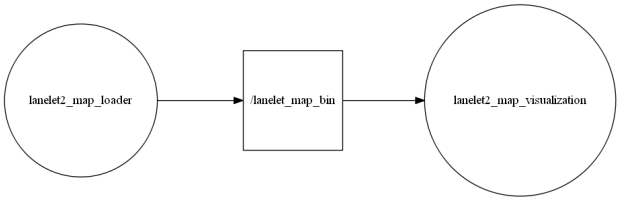

図 5: 開発したツールの出力例

とすると，前者の Node が Topic “lanelet_map_bin” として出力したデータが後者の Node に送られることになる．本研究では，これを図 5 に示すようなデータフローとして表現する．この出力図では，Node 名を丸で囲み，Topic 名を四角で囲んでいる．Topic が Topic 名を囲った四角に入っていく矢印側に存在する Node が Publisher であり，四角から出ていく側の Node が Subscriber である．

3.1 提案手法の手順

提案手法は，以下の 4 つのステップから構成される．これは図 4 の Step 1–4 に対応している．

(1) **Node 名の取得．** Node の入出力に関する Topic 通信を確立する処理は，1 つのファイルにまとめて記載されることが多い．本ツールでは，入力された ROS アプリケーションのディレクトリに含まれる個々の C++ ファイル（拡張子 “.cpp” を持つファイル）を Node とみなす．ファイル名が Node の主要な機能名を表していると仮定し，ファイル名から拡張子を除いた文字列を Node 名として使用する．例えばファイル名が “ROS_Node.cpp” であれば，Node “ROS_Node” として扱う．

(2) **Publish/Subscribe 命令の取得．** C++ ファイルそれぞれに対して正規表現による検索を行い，Publish/-Subscribe 命令を取得する．ソースコード 1 で示した命令の記述例と対応するように，ソースコード 2 の 1–2 行目の正規表現を定義した．関数呼び出しの括弧内，すなわち引数にあたる部分に対して，3 行目に示す正規表現で，最初に出現する文字列定数を取得し，Topic 名として使用する．正規表現では括弧の入れ子構造を扱うことはできないため，関数呼び出しの正規表現が引数部分全体を含むとは限らないが，Topic 名は第 1 引数に登場するため，括弧の入れ子構造に含まれることはないと仮定している．

(3) **Topic 通信の照合．** 得られた Publish/Subscribe のリストを照合し，同一の Topic 名を持つ Publish 命令と Subscribe 命令が通信を成立するものとして対応付け，Node 名と Topic 名をリストに記録する．原理的には N 対 N がありうるが，通常は Publish 命令 1 つに対して 1 つ以上の Subscribe 命令が対応する 1 対 N の通信になる．また，ここで通信が成立しなかった Publisher，Subscriber についても，それぞれ出力する．

ソースコード 2: 正規表現

```
1 \Wadvertise(<[^\(]+>)?\((?P<param>[^;]+)\)
2 \Wsubscribe(<[^\(]+>)?\((?P<param>[^;]+)\)
3 \"([^\"]+)\"
```

(4) **データフローの可視化．** 成立した Topic 通信のリストをグラフに変換し，Graphviz[*1]を用いてグラフを作成する．Publish 命令を持つ Node からその命令で指定された Topic へ，また，Topic からその Topic を指定した Subscribe 命令を持つ Node へ，それぞれ辺を接続する．

3.2 レビュー補助のための機能

本研究で開発したツールでは，Topic 通信の接続図を出力する際に，指定した Node 名，Topic 名を非表示にする機能を実装している．これにより大規模な ROS アプリケーションの解析を行う場合であっても，コードを理解した部分については非表示にすることができる．

また，複数回のコードレビューにおいて，どこが変化したかを簡単に把握できるように，ソースファイルの 2 つのバージョンが存在するときは，一方から抽出したグラフと，他方から抽出したグラフの差分を表示する機能を提供している．

4. 提案手法の予備評価

提案手法による可視化の妥当性を評価するため，提案手法を基にしたツールのプロトタイプを作成し，Autoware の解析結果をコードレビューの実務者と，Autoware の開発者に提供し，有用性について意見を収集した．

4.1 解析対象

事前に配布した Autoware の解析結果とツールの実行方法の説明を基にツールを利用してもらい，出力された図の有用性について意見を受けた．解析対象は GitHub に公開されている Autoware のソースコードで，2020 年 6 月 24 日時点のコミット[*2]を採用した [3].

4.2 解析結果

表 1 に本ツールにおける解析結果である各 CSV ファイルの要素数を示す．ツールが出力した CSV ファイル群から，全部で 71 組の Topic 通信の組が存在し，Topic 通

[*1] https://graphviz.org/
[*2] コミット ID:184ffedf900d648cab7db79abc734dee6c6896a0

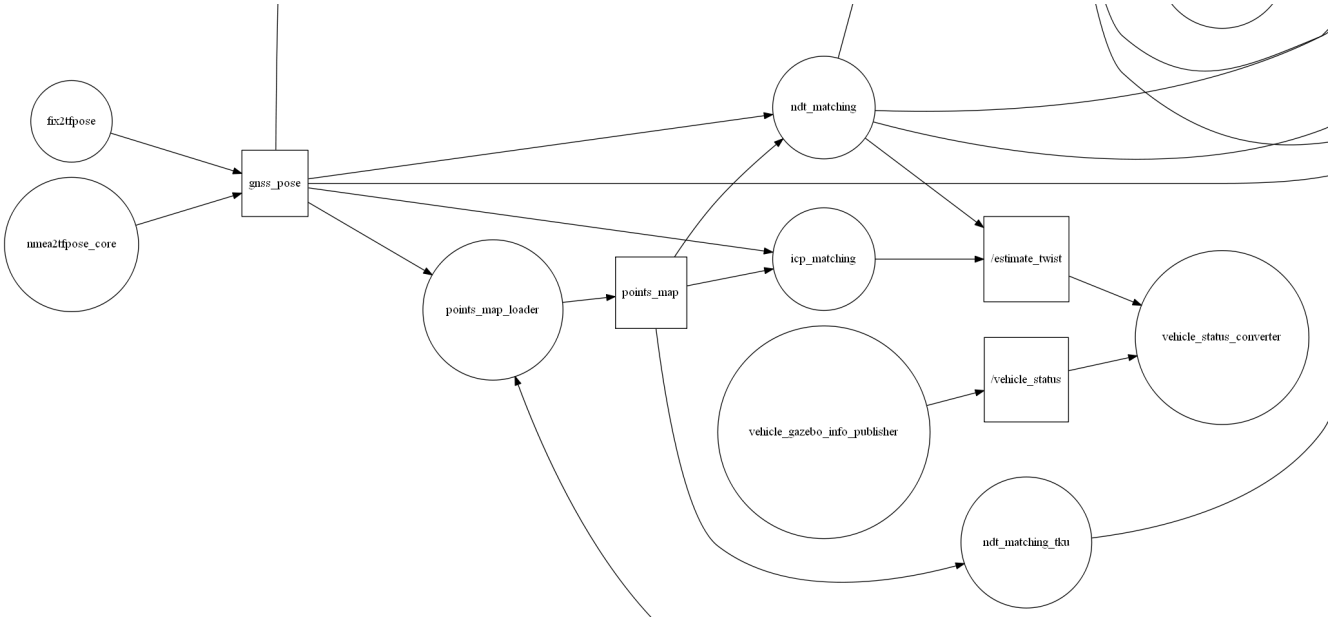

図 6: Autoware の解析結果（一部）

信が成立していない Publisher が 237 個，成立していない Subscriber が 287 個存在することが分かった．図 6 に配布した Topic 通信を解析した結果のグラフの一部 (1/8 程度) を示す．

表 2 に Topic 通信を示した CSV ファイルの一部を示す．このファイルに記録されたものが解析結果であり，図 6 のようにグラフ化される．表 3 に通信が成立していないと判定された Publisher の一部を，また表 4 には通信が成立していないと判定された Subscriber の一部を示す．これらの結果はグラフには含まれていない．

表 1: 出力された各 CSV ファイルの要素数

	要素数
Topic 通信	71
成立していない Publisher	237
成立していない Subscriber	287

表 2: 提案手法で抽出された Topic 通信の例

Publisher	Topic	Subscriber
nmea2tfpose_core	gnss_pose	points_map_loader
		icp_matching
		ndt_matching
		ndt_matching_monitor
		wf_simulator_core
icp_matching	/estimate_twist	vehicle_status_converter

表 3: 通信が成立していないと判定された Publisher の例

Publisher	Topic
health_aggregator	system_status
health_analyzer	system_status/summary
health_checker	node_status

表 4: 通信が成立していないと判定された Subscriber の例

Topic	Subscriber
/points_map	points_map_filter
initialpose	points_map_loader
traffic_waypoints_array	points_map_loader

4.3 評価結果

これらの図表に相当するファイルを提供したところ，コードレビュー実務者からは以下の意見を得た．

- 情報の粒度，内容共に十分なものを取得できている．
- Autoware をレビューした際に把握できなかった情報を取得できている．作業当時にこのツールを活用していれば役に立ったと思われる．
- 宇宙分野で比較的小規模なシステムに ROS を使うケースが増えているので，活用する機会は多くなる可能性がある．

また，Autoware 開発者からは以下の意見を得た．

- 静的解析は今まで行われてこなかったため試してみたい．
- Autoware を動かせる人にはそこまで重要ではないが，Autoware を導入せずとも使える，簡単に確認できるところは意味がある．

ROS の環境を準備せずに ROS アプリケーションのコードレビュー支援を行うことを目的としてツールの開発を行い，プロトタイプに関する評価を受けた．今回は Autoware を対象とした解析を行った．取得できている情報の粒度が十分であったことや実際のレビュー時に取得できていなかった情報の可視化に成功していることが評価から分かり，本ツールの有用性を確認することができた．更に，ROS アプリケーションに対して静的解析が今までに行われていないことや宇宙分野で比較的小規模なシステムに ROS を使うケースが増えていることから，本ツールが有効にはたらくことが期待される．

5. 今後の課題

5.1 ツールの改善

予備評価の過程と実務者らとの議論を通じて，提案手法の改善点が 2 つ見つかった．1 つ目は，Autoware.universe という Autoware の新バージョンがリリースされていることである．2 つ目は，remap という ROS アプリケーションの起動時に行われる処理についてである．

特に予備評価において，Topic 通信が成立していない Publisher/Subscriber が多数存在した原因はこの remap を想定していなかったことが考えられる．

Autoware.universe

Autoware.universe [2] は Autoware の新バージョンであり，Autoware 同様自動運転車の制御とシミュレーションを行うことができる．Autoware は ROS を用いて開発されている一方，Autoware.universe は ROS2 を用いて開発されている．

ROS2 は複数台ロボットを同時制御や組込み系のマイコンなど多様なプラットフォームへの対応を可能とした ROS の新バージョンである．ROS2 における Publish/Subscribe 命令をソースコード 3 に示す．ソースコード 3 に示した 1 行目が Publish の命令，2 行目が Subscribe の命令である．ソースコード 1 に示した ROS における Publish/Subscribe 命令と変化はあるが，大きくは異ならないため，ソースコード 2 で示した命令パターンから正規表現を構築すれば対応できると考えている．

remap

remap は Python や XML で記述される launch ファイルで行われる処理の 1 つであり，Publish/Subscribe 対象となる Topic 名を変更する命令である．これにより，確立している 2 つの通信を 1 つにまとめることや，複数の自動車をシミュレーションする際に各自動車に対応する Node ごとに通信を分けることなどが可能になる．

ソースコード 4 に XML ファイルにおける remap 命令を示す．2 行目の from の部分に書かれた “~time_abs” が remap する前の Topic 名であり，to の部分に書かれた “time_abs_remaped”が remap 後の Topic 名である．XML ファイルにおける remap 情報の取得については，Python のライブラリである lxml を利用して取得することを検討している．

ソースコード 5 に Python ファイルにおける remap 命令を示す．5 行目の remappings の部分に書かれた “string_topic”が remap する前の Topic 名であり，“string_topic_remaped”が remap 後の Topic 名である．Python ファイルにおける remap 情報の取得については Publish/Subscribe 命令と同様に正規表現を用いることを検討している．

ソースコード 3: Autoware.universe_Publish/Subscribe 命令

```
1 Publisher_ = this->create_publisher<std_msgs::
      msg::String>("chatter", 1000)
2 Subscriber_ = this->create_subscription<
      std_msgs::msg::String>("chatter", 1000, std
      ::bind(chetterCallback, this, _1));
```

ソースコード 4: XML ファイルの remap 命令

```
1 <node pkg="ros" type="pub" name="pub">
2       <remap from="~time_abs" to="
            time_abs_remaped" />
3 </node>
```

ソースコード 5: Python ファイルの remap 命令

```
1 def launch():
2    return launch.LaunchDescription([
3       launch_ros.actions.Node(
4          ~~partially omitted~~
5          remappings=[('string_topic', '
               string_topic_remaped')],
6          parameters=[{'string_param':'changed
               '}]
7       )
8    ])
```

5.2 ツールの評価

前節で述べた 2 点にツールを対応させ次第，次の評価を行うことを検討している．

インタビュー． 今回の予備実験同様，Autoware の開発者と共同研究者であるコードレビュー実務者に依頼する予定である．以下に検討している質問を示す．

- ツールを使うために特別な知識が必要か
- ツールを使うために必要な操作が多いと感じたか
- ツールの実行速度は適切であるか
- コードレビューに活用できると感じたか
- ほとんどの人がすぐに使いこなせるツールであるか

- ツールを使うのが面倒だと感じたか

これらの項目は NASA TLX を参考に設定した [13]．各項目について 5 段階のアンケート調査を行う予定である．また，追加機能についても利用しやすさに関するアンケートを検討している．

精度評価． 動的解析ツールである rqt_graph の出力を正しいものとみなし，開発したツールの出力と比較，Precision および Recall による精度の評価を行う．具体的にはそれぞれのツールでグラフ構造を表現した dot ファイルを出力し，2 ファイルの差分によって評価を行う．

6. まとめ

ROS の実行環境なしでも ROS アプリケーションのコードレビューを容易に実施できる環境を実現するため，Node ファイルに書かれた Topic 通信処理の記述を静的に解析し，データフローの可視化を行うツールを開発した．Topic 通信に用いられる Publish/Subscribe 命令を正規表現を用いて取得することでツールを実装した．

開発したプロトタイプツールを用いた Autoware の解析に対して，コードレビュー実務者と Autoware 開発者からの意見を受けた．その結果，ツールの有用性や今後ツールの活躍が期待できることが確認できた．好意的な評価の一方でツールの改善点が見つかったため，現在はその改善点である，Autoware.universe と remap の解析に対応した新バージョンの開発を行っている．このツールが完成し次第，再度評価を依頼するとともに，rqt_graph の出力を比較し，定量的に本ツールの精度評価を検討している．これらの評価実験を行い，よりユーザにとって使いやすいツールの開発を目指す．

謝辞

本研究は，JSPS 科研費 JP20H05706，JP22K21279 の助成を受けたものです．

参考文献

[1] Afrel Col. Ltd.: 教育版レゴマインドストーム EV3 とは，https://afrel.co.jp/technology-info/information-ev3/ev3-product/15365/.
[2] AutowareFoundation: The Autoware Foundation, https://github.com/autowarefoundation.
[3] AutowareFoundation: Autoware.AI, https://github.com/autowarefoundation/autoware/tree/autoware-ai.
[4] Foote, T.: ROS Community Metrics, http://wiki.ros.org/Metrics (Accessed on 11/06/2022).
[5] IEEE Standard Association: IEEE 1012-2004 - IEEE Standard for Software Verification and Validation (2005).
[6] Ishida, N., Ishio, T., Nakamura, Y., Kawaguchi, S., Kanda, T. and Inoue, K.: Visualization of inter-module dataflow through global variables for source code review, *IEICE TRANSACTIONS on Information and Systems*, Vol. 101, No. 12, pp. 3238–3241 (2018).
[7] Makhshari, A. and Mesbah, A.: IoT Bugs and Development Challenges, *Proc. IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 460–472 (2021).
[8] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.: ROS: an open-source Robot Operating System, *ICRA Workshop on Open Source Software* (2011).
[9] Sommerville, I.: *Software Engineering, 10th Edition*, Pearson (2016).
[10] Ujiie, R., Katahira, M., Miyamoto, Y., Nakao, H. and Hoshino, N.: Measurement of JAXA's IV&V Activity Effectiveness Based on Findings, *Proc. Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pp. 291–296 (2011).
[11] 宇宙航空研究開発機構：IV&V ガイドブック導入編 (2019).
[12] 田崎　豪：Autoware ではじめる自律移動技術入門，森北出版 (2021).
[13] 三宅晋司：人間工学のための計測手法　第 3 部：心理計測と解析（6），人間工学， Vol. 51, No. 6, pp. 391–398 (2015).
[14] 安積卓也，福富大輔，徳永翔太，橘川雄樹，清谷竣也：Autoware-自動運転ソフトウェア入門，リックテレコム (2021).