Vol.2019-ARC-240 No.37
Vol.2019-SLDM-191 No.37
Vol.2019-EMB-53 No.37
2019/2/28

車載知能制御システム向けソフトウェア設計手法の提案

小澤 慶祐[1]　本田　晋也[1]　松原 豊[1]　高田 広章[1]　加藤 寿和[2]　山本 整[2]

概要：近年，先進運転支援システムや自動運転の普及により，車載システムのアーキテクチャおよび車載システムの研究開発が変化している．そのような車載システムの研究開発において，プロトタイプ開発では設計生産性の高い ROS2 を，製品開発では信頼性の高い AUTOSAR-AP をソフトウェアプラットフォームとして使用することが考えられている．しかしながら現状では，ROS2 を用いたプロトタイプから AUTOSAR-AP を用いた製品とする設計フローが確立していない．そこで本研究では，ROS2 から AUTOSAR-AP への移行を含む設計フローの提案，検討を行った．はじめに，設計フローの要件を提案し，提案した要件を満たすような設計フローの提案を行った．次に，設計フローの検討のために，ROS2 で作成された自動走行ロボットのデモアプリケーションを AUTOSAR-AP のアプリケーションに書き換えるケーススタディを行った．最後に，ケーススタディの結果を踏まえ，提案した設計フローのより詳細な検討を行った．結果として，ROS2 から AUTOSAR-AP への開発環境の移行は可能であるが，効率的な移行のために ROS2 での開発時に記法や使用する機能を一部制限することが好ましいと考えられる．

1. はじめに

近年，先進運転支援システムや自動運転の普及により，車載システムのアーキテクチャが変化している．また，車載システムのアーキテクチャの変化に伴って，車載システムの研究開発も変化している．そのような車載システムの開発環境として，プロトタイプ開発では ROS2（Robot Operating System 2）を使用し，最終的な製品開発では AUTOSAR-AP（AUTomotive Open System ARchitecture Adaptive Platform）を使用することが考えられている．

ROS2 はロボットアプリケーションの作成を支援するミドルウェアおよびツール群であり，ライブラリやシミュレーション環境が充実している．そのため，初期の開発段階で使用することに適しており，プロトタイプ開発で使用される．一方，AUTOSAR-AP は世界的に標準化が進められている車載システムの仕様のため，最終段階の製品開発に使用される．ROS2 はアプリケーションの開発環境でありながら実行環境としての側面も持ち合わせている．同様に AUTOSAR-AP も仕様に準ずる実装は開発環境と実行環境を提供する．

車載システムの研究開発では，ROS2 から AUTOSAR-AP へと開発環境が変化し，開発の段階によっては ROS2 と AUTOSAR-AP の開発環境が混在することが考えられる．しかしながら，ROS2 と AUTOSAR-AP を使用した研究開発の設計フローについては研究が十分に行われていない．関連技術の研究としては，丸山らは，ROS2 の通信性能の評価を [1]，Casini らは，ROS2 のスケジューリングモデルと応答時間分析の提案を行っている [2]．また，Menard らは，AUTOSAR-AP の決定論的車載ソフトウェアの設計のためのフレームワークを提案している [3]．

そこで，本研究では ROS2 から AUTOSAR-AP への開発環境の移行を含む設計フローの提案，検討を行う．はじめに，設計フローで使用する ROS2 と AUTOSAR-AP の説明を行う．次に，設計フローの要件の提案を行い，提案した要件を満たすような設計フローの提案を行う．最後に，提案した設計フローのケーススタディを行い，設計フローのより詳細な検討を行う．

2. ROS2 と AUTOSAR-AP

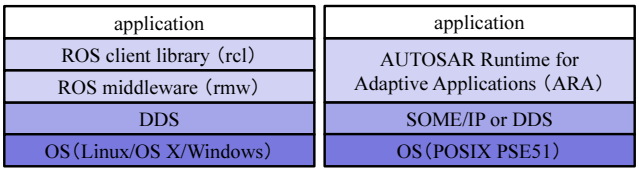
本章では，本研究で対象とするロボット制御ミドルウェア ROS2 および高性能コンピューティング ECU（Electronic Control Unit）向け仕様 AUTOSAR-AP について説明をする．

2.1 ROS2

ROS2 は ROS（Robot Operating System）をベースとしているため，まず ROS について説明する．ROS はロボットアプリケーションのためのライブラリとツールを提供するロボット制御ミドルウェアとして用いられている．Open Robotics(旧 Open Source Robotics Foundation）管

[1] 名古屋大学 大学院情報学研究科
[2] 三菱電機株式会社 情報技術総合研究所

Vol.2019-ARC-240 No.37
Vol.2019-SLDM-191 No.37
Vol.2019-EMB-53 No.37
2019/2/28



(a) ROS2　　(b) AUTOSAR-AP

図 1　ROS2 と AUTOSAR-AP のシステム構成の概要

理の下，オープンソースソフトウェアとして公開され，開発が続けられている [4].

ROS では研究分野での利用を主なユースケースとして想定していたが，ROS2 では製品への利用もユースケースに想定している．ROS2 と ROS のユースケース以外の主な違いに，開発にオープンソースライブラリを使用していることがあげられる．ROS は独自に開発されたシステムによって構築されている．一方，ROS2 はシステムの一部に既製のオープンソースライブラリが使用されている．オープンソースライブラリを使用することで，少ない手間でソフトウェアを維持することができるという利点がある．使用例の 1 つとして，ROS2 の通信用ライブラリは DDS（Data Distribution Service）が使用されている．DDS は publish/subscribe 通信を提供するミドルウェアの国際標準であり，ROS2 は DDS を使用して，プロセス間通信を行っている [5].

ROS2 のシステム構成の概要を図 1 の (a) に示す．図のように，ROS2 は汎用 OS 上で動作することを基本としている．また，ユーザは ROS2 の機能を使用するために ROS client library（rcl）で提供される API を用いてアプリケーション作成する．ROS middleware（rmw）は異なる実装の DDS を使用するために提供されるライブラリである．

2.1.1　ROS2 の処理単位

ROS2 では，ノードという単位で処理を記述する．各ノードには，単一の機能をモジュールとして実装する．また，1 つの実行可能ファイルすなわち 1 つのプロセス内に複数のノードを記述可能である．

メイン関数によりノードが生成され，周期実行や後述する通信機能によりコールバックとしてノードの各種処理が実行される，というのがノードの基本的な振る舞いである．また，ROS2 では，ノードの状態を管理して処理の振る舞いを制御する lifecycle ノードも提供されている．

2.1.2　ROS2 の通信機能

ROS2 は通信機能として，topic，service，parameter を提供している．各通信機能の概要を図 2 の左に示す．

topic

topic とは単方向の通信であり，通信の送信者を publisher と呼び，受信者を subscriber と呼ぶ．通信は publisher が topic にデータを送信し，subscriber が topic からデータを受信する形で行われる．

service

service とは双方向の通信であり，通信の要求者を client と呼び，応答者を server と呼ぶ．service は client が sever に request を送信し，server が response を client に返す形で行われるクライアントサーバモデルの通信である．

parameter

parameter とはグローバル変数のようなデータ共有機能である．ノードで保持している parameter を，ノード内やノード外から変更，参照することができる．

2.1.3　ROS2 におけるアプリケーションの実行

ROS2 では ROS launch と呼ばれる，複数のアプリケーションを起動できる機能が提供されている．ROS launch により，システム全体を容易に起動することが可能となる．また，ROS2 で作成したアプリケーションは ROS2 の実行コマンドである”ros2 run”を用いて実行することも可能である．

2.2　AUTOSAR-AP

AUTOSAR は，自動車メーカ，サプライヤ，サービスプロバイダ，および自動車エレクトロニクス，半導体，ソフトウェア業界の企業の世界的な開発パートナーシップである．AUTOSAR の主な目標は，基本的なシステム機能と機能インターフェースの標準化である [6].

AUTOSAR では開発の成果として仕様をリリースしており，AUTOSAR-CP（Classic Platform）と AUTOSAR-AP（Adaptive Platform）がリリースされている．AUTOSAR-CP は厳しいリアルタイムおよび安全性の制約がある組込みシステム向けの仕様である．AUTOSAR-AP は自動運転などを構築する高性能コンピューティング ECU 向けの仕様である．ECU とは，自動車の電子制御を担うコンピュータのことである．AUTOSAR-AP のシステム構成の概要を図 1 の (b) に示す．通信ライブラリには DDS もしくは SOME/IP（Scalable service-Oriented MiddlewarE over IP）を使用する．SOME/IP は車載ソフトウェアの通信を定義するサービス指向のミドルウェア仕様である．AUTOSAR-AP は AUTOSAR Runtime for Adaptive Applications（ARA）を仕様として定義し，API とサービスを提供する．

2.2.1　AUTOSAR-AP の処理単位

AUTOSAR-AP では，POSIX のスレッドとして処理を記述する．スレッドには単一の機能をモジュールとして実装する．メイン関数でスレッドを作成し，スレッドによって処理を実行する．

2.2.2　AUTOSAR-AP の通信機能

AUTOSAR-AP は通信機能として，event，method，field を提供している．また，AUTOSAR-AP の通信は

Vol.2019-ARC-240 No.37
Vol.2019-SLDM-191 No.37
Vol.2019-EMB-53 No.37
2019/2/28

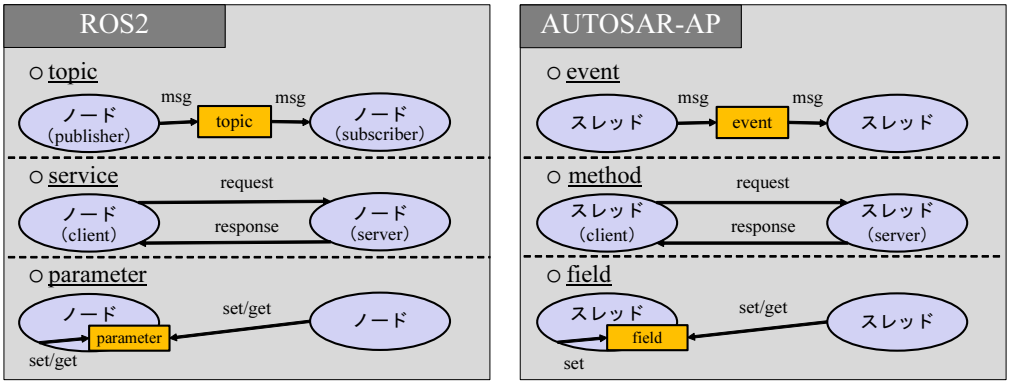


図 2　ROS2 と AUTOSAR-AP の通信の概要

Proxy/Skeleton アーキテクチャで構成されている．インターフェース定義から生成されるクライアント側のコードを Proxy，サーバ側のコードを Skeleton と呼ぶ．各通信機能の概要を図 2 の右に示す．

event

event とは単方向の通信であり，Skeleton はデータの送信者であり，Proxy はデータの受信者である．

method

method は双方向の通信であり，Proxy は request を送る client であり，Skeleton は request を受け response を返す server である．

field

field はグローバル変数のようなデータ共有機能である．field は Skeleton で保持され，Proxy から変更，参照できる．

2.2.3　AUTOSAR-AP におけるアプリケーションの実行

AUTOSAR-AP のアプリケーションは，マニフェストファイルに記述された実行情報に基づいて，エクゼキューションマネージャにより実行される．エクゼキューションマネージャはプラットフォームの初期化と，アプリケーションの実行およびシャットダウンを担当する．アプリケーションのランタイムスケジューリングはエクゼキューションマネージャと OS が連携して構成，実行される．また，作成したアプリケーションの実行ファイルを手動で実行することも可能である．

3.　知能制御システムの設計フロー

本章では，本研究で提案する車載システムの設計フローについて説明する．まず，1 章で述べているように本研究では自動運転を行う車載システムを対象として設計フローの提案を行う．自動運転は大きくアクチュエータやセンサを制御する機能と，自動運転の認知や判断を行う機能に分けることができる．最初に，初期のプロトタイプと最終的な製品の開発環境の構成を決定する．

3.1　プロトタイプ

プロトタイプ開発では作業を効率良く行うために，初期段階では PC 上で開発を行うものとする．PC 上では汎用 OS が動作し，開発環境の ROS2 上でアプリケーションの開発を行う．また，システム開発のために，ROS2 の開発支援ツールの使用を想定している．開発支援ツールの代表例として，Rviz と Gazebo があげられる．Rviz は ROS 独自のツールでロボットの位置情報や地図情報などを可視化するために使用される．Gazebo はロボットのシミュレーション機能を提供するオープンソースソフトウェアである．プロトタイプの開発環境の構成を図 3 の (a) に示す．

3.2　最終製品

製品開発では開発したシステムを ECU 上に機能分配し，製品開発を行う．AUTOSAR-AP 上で動作する先進運転支援システムや自動運転の認知，判断を行う機能を ECU に配置する．AUTOSAR-CP 上で動作するアクチュエータやセンサを制御する機能は上述の ECU とは別の ECU 上に配置する．一般的に認知や判断を行う ECU と制御を行う ECU は異なる ECU を用いる．これは，機能を切り離すことによる信頼性の向上や機能によって適したスペックの ECU を用いることによるコスト削減を目的としている．最終製品の開発環境の構成を図 3 の (d) に示す．

3.3　設計フローの要件

プロトタイプから最終的な製品開発を効率的に実施するために，本研究で定める車載システムの設計フローが満たすべき要件とその要件を設定した動機について説明する．本研究では開発環境の移行に重点を置いており，要件も開発環境の移行に重点を置いて設定している．また，自動運転などの車載システムを対象としているため，AUTOSAR-AP を中心に設計フローの提案や検討を行う．

[要件 1] プロトタイプ開発では ROS2 を用い，製品開発では AUTOSAR-AP を用いること

プロトタイプ開発に適している ROS2 を研究開発の初期

Vol.2019-ARC-240 No.37
Vol.2019-SLDM-191 No.37
Vol.2019-EMB-53 No.37
2019/2/28

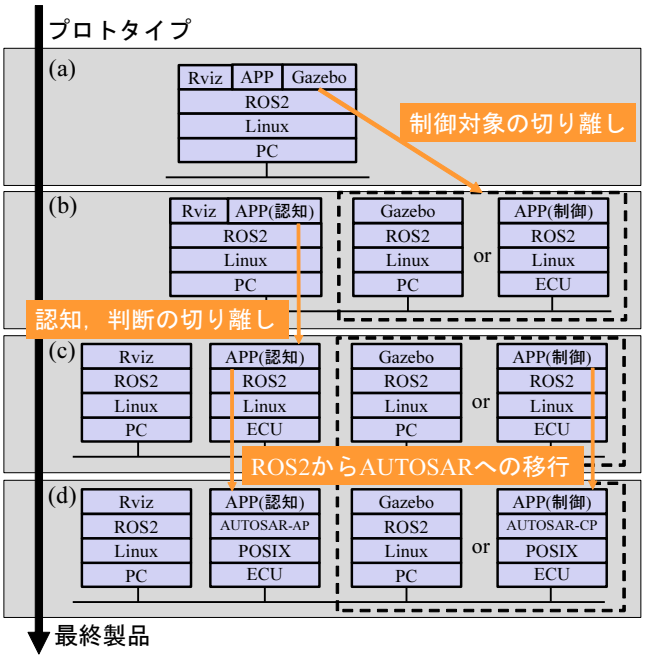


図 3　設計フロー

段階で使用することで，システムの開発を効率的に行うことができる．一方，製品開発に向いている AUTOSAR-AP を最終段階で使用することで，信頼性の高いシステムを開発することができる．

[要件 2] 開発環境の段階的な移行が可能であること

システム全体の移行を一度に行うと設計の手戻りが発生した際のコストが大きくなるため，開発環境の移行を段階的に行うことができるようにする．

[要件 3] 製品開発において **ROS2** の開発支援ツールを使用できること

ROS2 の開発支援ツールは研究開発に有用であるため，AUTOSAR-AP の環境でも ROS2 の開発支援ツールの使用することが可能であれば AUTOSAR-AP での開発も効率的に行うことができる．よって，AUTOSAR-AP へと開発環境が移行した後も ROS2 の開発支援ツールを使用するユースケースを考慮し，AUTOSAR-AP と ROS2 の開発環境が混在する開発環境において ROS2 の開発支援ツールの一部は使用できるようにする．

[要件 4] 環境移行のコストが小さくなること

環境移行にかかるコストが小さくなることはシステムの開発のスピードに繋がる．新しい技術が年々登場し，製品の開発サイクルを短くすることが重要なため，システムの研究開発のスピードの向上は製品の質と同様に重要な項目である．

3.4　設計フローの提案

3.3 節で述べた車載システムの設計フローの要件を踏まえて車載システムの設計フローを提案する．

[要件 2] より，プロトタイプから最終製品への直接の移行は変更量が大きいため，その間に中間的な状態を設けて設計の効率化を図る．中間状態は図 3 の (b)，(c) に示す．初めに，(b) はシミュレーションを行う機能を切り離し，別 PC 上にシミュレーション環境を移す．もしくは，シミュレーションを行う機能を，実機を制御する機能に置き換え ECU 上に移す．次に，(c) は認知や判断を行う機能を ECU 上に移す．最後に，(d) は ROS2 の開発環境とアプリケーションを AUTOSAR-AP に移行し，製品開発を行う．

ここで，提案した設計フローで解決すべき課題を 2 つあげる．

- [課題 1] ROS2 上で動作するアプリケーションを AUTOSAR-AP 上で動作するアプリケーションへ変更する方法 (c→d)
- [課題 2] ROS2 と AUTOSAR-AP の環境が混在する場合に ROS2 が動作する PC と AUTOSAR-AP が動作する ECU 間で通信を行う方法 (d)

表 1　ROS2 と AUTOSAR-AP の機能比較

	ROS2	AUTOSAR-AP
主な用途	研究	製品開発
特徴	ツールが豊富で実装が容易	信頼性の高い仕様
OS	Linux / OS X / Windows	POSIX 準拠
通信形式	DDS	SOME/IP / DDS
記述言語	C++ / python	C++
スケジューリング	OS 依存	動的にスケジューリング可能

4.　ROS2 と AUTOSAR-AP の比較

3.4 節であげた設計フローの課題は ROS2 と AUTOSAR-AP の差異が原因のため，本章では課題解決の前準備として，ROS2 と AUTOSAR-AP の機能比較を行う．

4.1　機能比較

ROS2 と AUTOSAR-AP の機能の簡単な比較を表 1 に示す．AUTOSAR-AP は ROS2 と異なり仕様であるため，以降の比較では実装として APD（AUTOSAR-AP Demonstrator）を使用することを前提に説明をする．APD は AUTOSAR-AP の仕様と同時にリリースされる実装例である．

4.2　処理単位の比較

処理単位は ROS2 ではノードであり，AUTOSAR-AP では POSIX のスレッドである．ROS2 と AUTOSAR-AP では共にメイン関数でノードやスレッドを作成し，その後実行するという振る舞いは類似している．ROS2 ではノードの実行は ROS2 独自のライブラリで行われるが，AUTOSAR-AP ではスレッドの実行は OS によって行われ

る点が異なっている．

また，ROS2 の lifecycle ノードでは各ノードの状態を管理することが可能だが，AUTOSAR-AP ではスレッドの状態を管理することはできない．しかし，AUTOSAR-AP では State Management という機能にユーザが定義した状態を管理することができる Function Group が提供されている．Function Group の状態によってプロセスの振る舞いを制御することが可能である．

4.3 通信機能の比較

ROS2 の通信は DDS を使用しており，単方向通信の topic，双方向通信の service，グローバル変数のような parameter が提供されている．AUTOSAR-AP では DDS か SOME/IP を通信に使用することを規定されており，単方向通信の event，双方向通信の method，グローバル変数のような field が提供されている．

それぞれの通信機能はサービス指向に基づいており，基本的な機能は類似している．機能の異なる点として，通信の情報の記述方法の違いがあげられる．ROS2 では通信を行うインスタンスを作成する際に引数として，通信を行うための情報の指定を行う．すなわち，情報をプログラム中で記述する．一方，AUTOSAR-AP ではサービスインターフェースの定義から生成された Proxy と Skeleton をインスタンス化して通信を行う．このサービスインターフェースに通信を行うための情報を記述する．サービスインターフェースの定義はマニフェストファイルと呼ばれる，プログラムとは別の設定ファイルに記述する．

topic と event の比較

ROS2 の topic では publisher をインスタンス化し，メンバ関数を呼び出すことでデータの送信を行う．データの受信は subscriber をインスタンス化し作成することで行われる．subscriber はコールバックとしてデータの受信時に処理を実行する．

AUTOSAR-AP の event ではサービスインターフェースから生成された Skeleton をインスタンス化し，メンバ関数を呼び出すことで，データの送信を行う．同様に，データの受信についても，Proxy をインスタンス化し，メンバ関数を呼び出すことで行う．

topic と event には大きな違いが 2 つある． 1 つ目は，topic は publisher のインスタンス化や subscriber のコールバックを作成する際に通信相手の情報や通信するデータの情報を決定するが，event では Proxy や Skeleton を生成するサービスインターフェースの定義によって通信相手の情報や通信するデータを決定する点である． 2 つ目は，topic ではデータを受信するための機能が用意されており，subscriber のコールバックを作成するだけでよいが，event では Proxy のメンバ関数を用いて受信データの有無の確認や受信データの取得を行う必要がある点である．

service と method の比較

ROS2 の service の server を担うノードではサービスのリクエストで呼び出されるコールバックである service を作成する．一方，client を担うノードではサービスのリクエストを送信し，そのレスポンスによって呼び出されるコールバックである client を作成する．

AUTOSAR-AP の method では event と同様に Skeleton と Proxy をインスタンス化し，メンバ関数を使用することで，method のコールバック関数の作成や，method の呼び出しを行う．

service と method の違いは，topic と event と同様に通信の情報の記述方法の違いがあげられる．

parameter と field の比較

ROS2 の parameter はノードが parameter を定義することでノード内やノード外から parameter の値を変更および参照することができるようになる．

AUTOSAR-AP の field では field の定義や値の変更や参照を行うためにサービスインターフェースから生成された Proxy や Skeleton のメンバ関数を使用する必要がある．

parameter と field の機能的な違いとして，parameter は定義したノードからでも値の変更と参照の両方を行うことができるが，field は field を定義したインスタンスからは値の参照は行えない点が挙げられる．これは field を定義する Skeleton にはデータを変更するメンバ関数しか用意されていないためである．

4.4 アプリケーションの実行方法の比較

手動でアプリケーションを実行する場合については単純に実行するのみのため，ROS launch による実行とエクゼキューションマネージャによる実行を比較する．ROS launch は ROS launch を実行することでアプリケーションを実行する．一方，AUTOSAR-AP ではエクゼキューションマネージャが OS の起動時にアプリケーションを自動的に実行する．ROS launch ではシステム全体の実行を簡単に行えるため，プロトタイプ開発に適している．エクゼキューションマネージャでの実行は OS 起動時に自動で実行されるため，最終的な製品開発に適している．

4.5 独自機能の比較

1 つ目にログの出力について，ROS2 と AUTOSAR-AP では共にログの出力を行う機能を提供している． 2 つ目に周期実行のためのタイマについて，ROS2 ではノードを周期実行するためのタイマを提供しており，ノードを周期実行するアプリケーションの作成が容易である．一方，AUTOSAR-AP にはタイマが用意されておらず，周期実行を行うためには，ユーザが周期実行を行う処理を実装する

Vol.2019-ARC-240 No.37
Vol.2019-SLDM-191 No.37
Vol.2019-EMB-53 No.37
2019/2/28

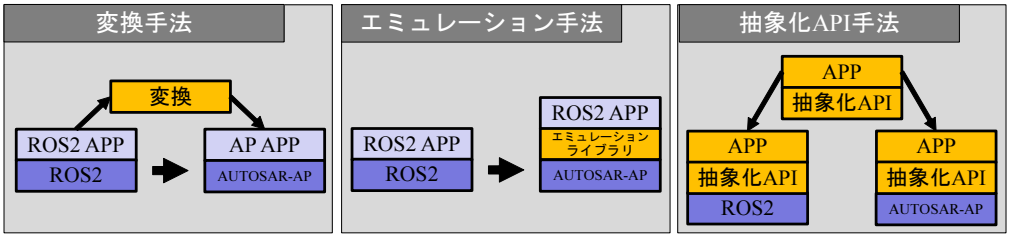


図 4　ROS2 から AUTOSAR-AP への移行方法

必要がある．

5.　設計フローの課題解決手法

4 章の ROS2 と AUTOSAR-AP の機能比較を踏まえたうえで，3.4 節であげた設計フローの課題解決手法を提案する．

5.1　ROS2 から AUTOSAR-AP への移行手法

[課題 1] であげた ROS2 から AUTOSAR-AP へと移行する手法を 3 つ提案する．それぞれの移行手法の概要を図 4 に示す．

変換手法

ROS2 で作成したアプリケーションを AUTOSAR-AP で実行可能なアプリケーションに変換する．変換の方法として，手作業でアプリケーションを書き換える方法と，アプリケーションを自動で変換するツールを作成する方法が考えられる．手作業での書き換えのメリットは，特別に必要となるものがなく作成したアプリケーションをすぐに書き換えることが可能なことである．デメリットは，手作業での書き換えを行うことができる知識を持った開発者が必要であることと，書き換えの作業自体にコストがかかることである．自動変換ツールのメリットは，変換を自動で行うことができるため，変換のコストが小さくなることである．デメリットは，ツールの作成自体にコストがかかることである．

エミュレーション手法

ROS2 で作成したアプリケーションを変更することなく AUTOSAR-AP で実行可能にするためのライブラリを使用する．ライブラリは ROS2 の API を AUTOSAR-AP の API に変換するライブラリである．メリットは，アプリケーションの変更を行う必要がないため移行のコストが小さくなることである．デメリットは，ライブラリの作成にコストがかかることと，変換ライブラリの存在によってアプリケーションの振る舞いが変わってしまう可能性があることである．

抽象化 API 手法

ROS2 と AUTOSAR-AP のアプリケーションの両方に変換可能な中間ライブラリを作成する．これは ROS2 から AUTOSAR-AP へと移行するという考え方とは異なり，ROS2 のアプリケーションを作成する段階から AUTOSAR-AP への変換を考慮したライブラリを使用する．メリットは，アプリケーションの変更を行う必要がないため移行のコストが小さくなることである．デメリットは，中間ライブラリを使用するため，ROS2 や AUTOSAR-AP の標準的なアプリケーションとは異なる記法になる可能性があることである．

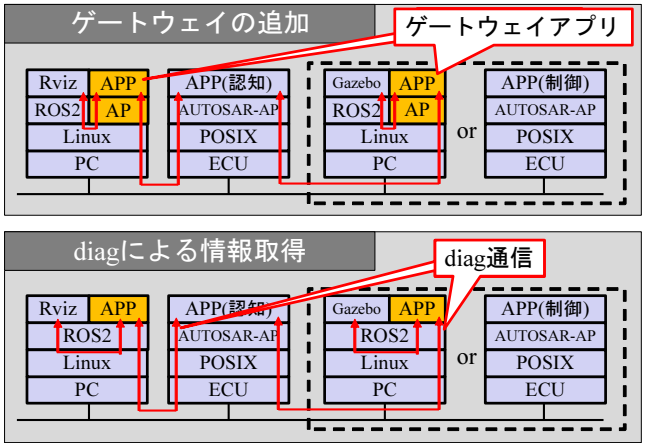


図 5　ROS2 のツールと AUTOSAR-AP アプリケーションの通信方法

5.2　ROS2 ツールと AUTOSAR-AP アプリケーションの通信手法

[課題 2] であげた PC と ECU 間で通信を行う手法を 3 つ提案する．ROS2 のツールと AUTOSAR-AP のアプリケーションの通信手法の概要を図 5 に示す．

DDS の使用

4.3 節で述べたように，ROS2 では DDS の仕様の準じた通信を行う．また，AUTOSAR-AP の仕様では SOME/IP または DDS を用いることが規定されている．よって，AUTOSAR-AP で DDS を用いて通信を行うことで ROS2 と AUTOSAR-AP のアプリケーション間で通信を行うことができる．しかし，AUTOSAR-AP は仕様であるため，実装によっては SOME/IP を用いて通信を行うことも考えられ，その場合には ROS2 と AUTOSAR-AP のアプリケーション間で通信を行うことはできない．

Vol.2019-ARC-240 No.37
Vol.2019-SLDM-191 No.37
Vol.2019-EMB-53 No.37
2019/2/28

表 2　TurtleBot3 の構成情報

SBC	Raspberry Pi 3 Model B+
Embedded Board	OpenCR1.0
Sensor	360 Laser Distance Sensor LDS-01
Actuator	Dynamixel XL430

ゲートウェイの追加

ROS2 と AUTOSAR-AP で通信の規格が異なる場合にはそのままでは通信を行うことができない．そこで，提案する方法は ROS2 のアプリケーションが送信した通信を AUTOSAR-AP のアプリケーションが受信できるように変換するゲートウェイを追加することである．ROS2 のツールを実行する PC 上に AUTOSAR-AP の環境を追加し，ECU と通信を行うゲートウェイを作成する．

diag による情報取得

ROS2 のツールが必要なデータを AUTOSAR-AP の Diagnostic Management を使用して取得することである．Diagnostic Management は ISO 14229-1 に準拠した UDS（Unified Diagnostic Services）の実装である．UDS とは，ECU 環境の診断通信プロトコルの国際規格である．

6. ケーススタディ

設計フローの中でも特に ROS2 から AUTOSAR-AP への移行方法を検討するために，本章では ROS2 のアプリケーションを AUTOSAR-AP のアプリケーションへ変換するケーススタディを行った．

6.1 対象アプリケーション

ROS2 のアプリケーションを AUTOSAR-AP のアプリケーションへと移行するケーススタディの題材として，TurtleBot3 のデモアプリケーションを使用した．ROS2 で作成された TurtleBot3 のデモアプリケーションを AUTOSAR-AP のアプリケーションに書き換え，アプリケーションの移行ための知見を得ることをケーススタディの目的としている．

6.1.1 TurtleBot3

TurtleBot3 は ROBOTICS によって開発された ROS を標準のプラットフォームとするロボットである．TurtleBot3 で実現可能な主な技術は SLAM（simultaneous localization and mapping）やナビゲーション，操縦である．これらの機能はホームサービスロボットに適したものであるが，本研究で対象としている自動運転にも関係のある技術であるため，ケーススタディの題材として選択した [7]．TurtleBot3 を構成するハードウェアの情報を表 2 に示す．

6.1.2 TurtleBot3 のデモアプリケーション

デモはオープンソースソフトウェアとして公開されており，ROS で開発されたデモと ROS2 で開発されたデモが公開されている．TurtleBot3 のデモの構成の概要を図 6 に示す．以降の説明は ROS2 のデモについての説明である．

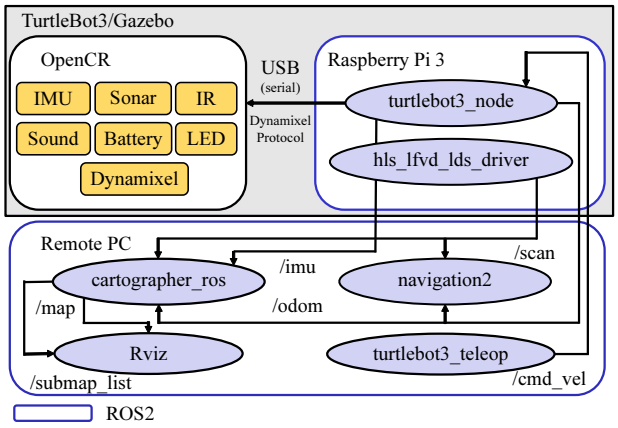


図 6　TurtleBot3 デモアプリケーションのノード構成

表 3　アプリケーション書き換えの環境

Remote PC	OS	Ubuntu 18.04
	ビルドシステム	yocto 3.1.9
	AUTOSAR-AP	APD1903
ECU	board	Raspberry Pi 3 model B+
	OS	Ubuntu 18.04

SLAM

SLAM は，任意の空間での現在位置を推定することで地図を作成するデモである．SLAM では TurtleBot3 に搭載されている 360°センサが読み取った情報を基にマップを作成する．図 6 では cartographer_ros が SLAM の機能を実装したノードである．

Navigation

Navigation は指定した目的地へ TurtleBot3 を移動させるデモである．目的地へ移動するために，TurtleBot3 に搭載されたロボットのエンコーダと IMU センサ，距離センサを使用する．また，周囲の環境を知るために，障害物の情報を含む地図を用意する必要がある．その地図は，SLAM で作成した地図を使用することができる．図 6 では navigation2 が Navigation の機能を実装したノードである．

6.2 アプリケーションの書き換え

次に，実際に行った ROS2 のアプリケーションから AUTOSAR-AP のアプリケーションへの書き換えについて説明する．表 3 にアプリケーションの書き換えを行った環境を示す．6.1.2 節で述べた Navigation のアプリケーションについて書き換えを行った．このアプリケーションは C++で記述されており，書き換え後のアプリケーションも C++で記述した．アプリケーションの書き換えは大きく分けて，通信機能の書き換えと処理単位の実行の書き換え，ROS2 独自の機能の書き換えに分類することができる．書き換えの結果を表 4 に示す．

表 4　アプリケーション書き換えの結果

機能	ROS2	AUTOSAR-AP	結果
処理単位	ノード	スレッド	可能
	lifecycle ノード	State Management	検証不可能
通信機能	topic	event	可能
	service	method	可能
	parameter	field	可能
独自機能	ログ出力		可能
	周期実行		可能

6.2.1　処理単位の書き換え

ノードの実行は POSIX のスレッド実行に書き換えを行うために，C++のライブラリである”std::thread”ライブラリを用いて書き換えを行った．

lifecycle ノードから State Management への書き換えは，State Management の Function Group を管理する機能が使用した APD1903 では未実装のため書き換えを行うことができなかった．

6.2.2　通信機能の書き換え

通信機能の書き換えは 4.3 節の機能比較から対応する機能に書き換えを行った．Navigation のアプリケーションは複数のノードで構成されており，各ノードが複数の topic と service，parameter を使用している．それらの通信機能をそれぞれ event と method，parameter へと書き換えた．また，通信の情報を記述するために，プログラムとは別にマニフェストファイルを新たに作成した．マニフェストファイルには，通信を接続するための情報や，通信するデータの型を記述した．

6.2.3　ROS2 独自機能の書き換え

次に，ROS2 の独自機能として書き換えを行ったログ出力と周期実行のためのタイマの機能について説明をする．ログ出力は AUTOSAR-AP にも同等の機能が提供されているため，その機能へと書き換えを行った．また，周期実行のタイマは AUTOSAR-AP には同等の機能がないため，C++のスレッドのスリープ機能と while 文で書き換えを行った．

6.3　アプリケーションの書き換えの結果と考察

ROS2 のアプリケーションの通信機能を AUTOSAR-AP の通信機能へと書き換えを行った結果，ROS2 と AUTOSAR-AP の機能の違いから，全ての機能を再現することはできなかった．しかし，大半の機能を書き換えることができたため，ROS2 のアプリケーションを作成する際に lifecycle ノードなど，対応する AUTOSAR-AP の機能が未実装のために書き換え不可能な機能に一部制限を設けることで，ROS2 のアプリケーションを AUTOSAR-AP のアプリケーションに書き換えを行うことが可能であると考えられる．

次に，移行のコストを軽減する手法として提案した自動変換ツールと中間ライブラリについて考察する．自動変換ツールは，処理の流れや API の使用方法などの ROS2 のアプリケーションの記法を制限することで実現可能だと考えられる．中間ライブラリについても，実現可能であると考えられるが，ROS2 と AUTOSAR-AP の記法の差異を埋めるために通常と異なる記法になることが考えられる．

したがって，既存の ROS2 のアプリケーションを AUTOSAR-AP に移行する際には手作業での書き換えが現実的な手法であると考えられる．また，今後 AUTOSAR-AP に移行することを前提として ROS2 でアプリケーションを作成する場合は，自動変換ツールや中間ライブラリを使用した方が，研究開発を効率良く進めることができると考えられる．

7.　おわりに

本研究では，自動運転の登場により変化する車載システムの研究開発の設計フローの提案と検討を行った．また，ケーススタディを通して，提案した設計フローのより詳細な検討を行った．結果として，ROS2 から AUTOSAR-AP への開発環境の移行は可能であるが，効率的な移行のために ROS2 での開発時に記法や使用する機能を一部制限することが好ましいと考えられる．

今後，ROS2 のアプリケーションをより効率的に AUTOSAR-AP のアプリケーションへ移行する方法や，ROS2 ツールと AUTOSAR-AP アプリケーションの通信方法の詳細な検討を進めていく．

参考文献

[1] Maruyama, Y., Kato, S. and Azumi, T.: Exploring the Performance of ROS2, *Proceedings of the 13th International Conference on Embedded Software*, EMSOFT ’16, New York, NY, USA, ACM, pp. 5:1–5:10 (online), DOI: 10.1145/2968478.2968502 (2016).

[2] Casini, D., Blaß, T., Lütkebohle, I. and Brandenburg, B. B.: Response-Time Analysis of ROS 2 Processing Chains Under Reservation-Based Scheduling, *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)* (Quinton, S., ed.), Leibniz International Proceedings in Informatics (LIPIcs), Vol. 133, Dagstuhl, Germany, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 6:1–6:23 (online), DOI: 10.4230/LIPIcs.ECRTS.2019.6 (2019).

[3] Menard, C., Goens, A., Lohstroh, M. and Castrillon, J.: Achieving Determinism in Adaptive AUTOSAR (2019).

[4] : Documentation - ROS Wiki, Open Robotics (online), available from ⟨http://wiki.ros.org/⟩ (accessed 2019-12-17).

[5] : ROS 2 Overview, Open Robotics (online), available from ⟨https://index.ros.org/doc/ros2/⟩ (accessed 2020-1-14).

[6] : AUTOSAR - Enabling Innovation, AUTOSAR (online), available from ⟨https://www.autosar.org/⟩ (accessed 2020-1-16).

[7] : TurtleBot3, ROBOTICS (online), available from ⟨http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/⟩ (accessed 2019-12-17).