 AUTOSARとROSそれぞれの特徴

背景/目的

- 自律制御向けコントローラのプラットフォームとして，AUTOSARもしくはROSのどちらかの開発環境を手の内化しておきたい.
- 次世代のアーキテクチャおよび通信プロトコル(特にTCP/IPベース)として，サービス指向アーキテクチャ(SOA:Service Oriented Architecture)やサービス指向通信(SOME/IP: Service-Oriented MiddlewarE over IP)などが登場しており，前提としてプラットフォームベースでの開発環境が用いられる.
- アーキテクチャ内の構成の整合性をツールで実装することが主流となってきており，MATLABがAUTOSARおよびROS対応のtoolboxを用意しているので動向を把握する.
- 標準化の意味でAUTOSARのBSWで用意される，故障状態管理(DEM)，機能抑制管理(FIM)の考え方を取り入れる(※ AUTOSAR ClassicPlatformにのみ具備される)

AUTOSARについて

1. CP(ClassicPlatform)が従来のECU向け(RTOS，MCU)，静的メモリ管理．
2. AP(AdaptivePlatform)がPOSIX(≒Linux)等の高度OS(仮想OSも)を想定した汎用PCに近いハードウェア向け．動的メモリ管理．

となっており，CPはソフトウェアパラメータをオフラインで構築する特性上，コンフィギュレーションツールが必須(dSPACE，Vectorなど)．またRTOS(OSEKベース)とRTE，BSW環境をライセンス購入が必要なことからツール費用面と学習コストの面で参入障壁がある．

TTTechをベースにCPを導入し，リアルタイムコントローラのプラットフォームとして構築するという構想を持つが必要なツールチェーンの見積から必要(ベンダ単体では成立しないので困難)．

TTTechのドライバ層を使用せずにTIのMCALを使用することになるのでこれもデメリット．

APについては，Linuxベースであるので比較的環境が用意しやすいメリットがあると思っているが，コンフィギュレーションツール等がどの程度必要なのかが不明なのと，ECU的に使えない

ROSについて

1. ROS1 : 研究用途向け，単体ロボット，非リアルタイム，必要な演算資源が多い，通信品質に依存(遅延等に対処する仕組みがない)
2. ROS2 : 組込対応，リアルタイム対応，複数台のロボットに対応

用途として，低速自律走行，非産業用ロボット(特殊用途，特殊車両にも？)，産業用ドローン等

プロセスノード間をEthernetでつなぐ方式なので，汎用性は高いがオーバーヘッド大きめ.

広義のロボティクス系，ネットワーク制御を見込むか.

オープンソース故に開発環境の導入ハードルが低い．(基本がUbuntuベースなので，各種パッケージは豊富だが，信頼性は担保されない.)

通信について

1. AUTOSAR + SOME/IPでサービス指向通信を
2. ROSもPublish/Subscribeという形式でサービス指向通信を持っている(topic)

双方，IP通信をベースにアプリケーションレイヤとして定義．

SOME/IPはAPの仕様．CPではデンソーなどが独自拡張している(標準に取り込まれたかどうかは不明)

→ サービス指向通信として，ROSの機能概要を見ればSOME/IPのおおよその概要が理解できそう．

MATLAB対応

AUTOSAR，ROSともにMATLAB/Simulinkがツールボックス対応をしている．(※AUTOSARはSW-Cコンポーネントおよび DEM/FIUサービスの定義のみ，コンフィギュレータおよびOSそのものは別途用意必要)

どちらかの環境を整備，集中することで，アプリ開発環境を統一することが期待．

ハードウェアによらない抽象化を期待できる(反面，下回りのポーティングには相応の知見が必要．自前でやらずに対応ハードを購入するのが良さそう)

シミュレータを活用し，机上と実機に分けることでリソースを統合集約しデータ蓄積と有効活用を期待する．

統合コントローラに相当するので，当然ながら，モータ制御やバッテリ制御などのドメイン特化用途には向かない．

Python等のOSS資源との相互利用が期待できる(機械学習分野，画像処理分野，特殊プロトコルの利用，など)，機能試作用途に有用．