Vol.2019-ARC-235 No.10
Vol.2019-SLDM-187 No.10
Vol.2019-EMB-50 No.10
2019/3/17

宇宙機向けソフトウェアプラットフォームにおける アプリケーション間通信モデルの比較検討

高田光隆[†1]　松原豊[†1]

概要：ネットワーク接続を前提とした組込みシステムが多く採用されるようになり，アプリケーション/プロセッサ間通信の通信ミドルウェアが数多く仕様化・実装されている．しかしそれらの通信ミドルウェアは個々のシステム特性に適する機能を有しているが，新規にアプリケーション間の通信ミドルウェアを採用するシステムの場合，指針となるような情報を見つけることが難しい．
本論では，宇宙機向けソフトウェアプラットフォームと他の仕様化・実装されている組込み向けのアプリケーション/プロセッサ間通信を行う通信ミドルウェアを対象に，その通信モデル，通信プロトコル，QoS，セキュリティ，メッセージタイプ，適用ドメインといった項目に対して比較・分類を行い，宇宙機向けソフトウェアプラットフォームに対しての改善について提案する．

キーワード：アプリケーション間通信，core Flight System，AUTOSAR，ソフトウェアバス

1. はじめに

近年 CubeSat[1]に代表されるような小型・超小型衛星の開発が活発になってきており，大学やいままで宇宙分野に参入してこなかった企業が研究や実験目的で衛星を開発し，他の衛星と一緒に打ち上げることができるようになってきた．CubeSat は国内では大学を中心に活動が行われているが，海外では企業が宇宙ビジネスの一環として利用しているケースも散見され，今後一層小型・超小型衛星による開発が見込まれる．

車載分野では安全性向上のための要求や自律的な支援運転など高機能化の要求に応えるためソフトウェアが大規模化・複雑化の一途をたどっている．そのような中でこれ以上のコスト増を抑えるためにもソフトウェアの再利用，モデルベース開発やコンポーネント開発など抽象度の高い設計・開発が求められており，ソフトウェアプラットフォームとして AUTOSAR 仕様[2]の採用検討などが行われている．

宇宙分野での衛星搭載ソフトウェアはプロジェクト・ミッションごとに独立した開発が行われている状況であり，搭載ボードとともにミッションアプリケーション開発に必要なミドルウェアが搭載ソフトウェア開発企業から提供される場合もあるが，車載アプリケーションのような量産開発ではないため，アプリケーションの再利用などの観点というよりも搭載ボードのハードウェア制御を吸収するためのソフトウェア設計となっている場合が多い．

一方海外では，NASA ゴダード宇宙飛行センター（GSFC）を中心として衛星搭載ソフトウェアの開発コストを削減するため，アプリケーションの再利用を目的としたソフトウェアプラットフォームである core Flight System（cFS）[3]を開発し，いくつかの衛星プロジェクトにおいて採用の実績がある．

日本においては衛星内ネットワークの通信プロトコルを共通化する動きとして SpaceWire[4]を採用する取り込みが行われ，中・大規模衛星では搭載の通信ネットワークとなりつつある．しかしより上位のアプリケーション間通信やアプリケーションそのものの再利用に関する議論をするには至っていない．

本論文では車載向けソフトウェアプラットフォームの仕様の違いや特徴を整理し，宇宙機向けのソフトウェアプラットフォームの改善を提案する．

2. core Flight System (cFS)の特徴

2.1 core Flight Executive (cFE)

GSFC はプロジェクトに依存しない資金調達を獲得し，過去に再利用されたフライトソフトウェアや設計情報を評価・分析し， Operating System Abstraction Layer(OSAL) をオープンソースとして開発し公開した[5]．その後，OSAL は NASA 月周回無人衛星 LRO のソフトウェアとして最初に搭載され，OSAL とともにフライトソフトウェアの基本モジュールをまとめた core Flight Executive(cFE)が開発・搭載された[6]． cFE は表 1 に示す 6 つのサブシステムから構成されている．

表 1　cFE のサブシステム

名称	概要
Executive Service	cFE の起動やアプリケーション/ドライバ，メモリ管理
Time Service	時刻管理，配信，同期サービス
Event Service	アプリケーションのイベントメッセージ処理の管理
Software bus	アプリケーション間の通信処理
Table Service	フライトソフトウェアのパラメータセット（テーブル）の管理
File Service	ファイル管理

†1 名古屋大学
Nagoya University

Vol.2019-ARC-235 No.10
Vol.2019-SLDM-187 No.10
Vol.2019-EMB-50 No.10
2019/3/17

2.2 階層化ソフトウェア

cFS ではアプリケーションからハードウェアに至るまでのフライトソフトウェアをいくつかのレイヤに分け，レイヤ間のインタフェースを規定するといった階層化ソフトウェアのアーキテクチャになっている．

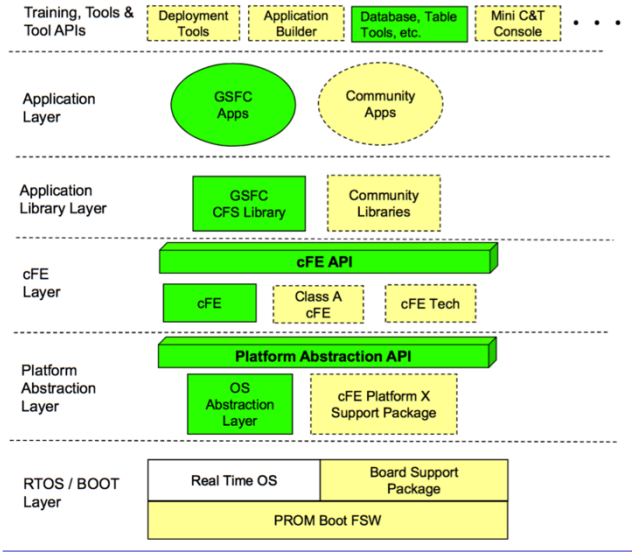

図 1　core Flight System のアーキテクチャ

(1) RTOS / BOOT 層

搭載ボードの IO デバイスに必要なドライバソフトウェア群（Board Support Package）および一般的な RTOS とファイルシステムから構成されている．現在サポートしている RTOS 環境としては，VxWorks，RTEMS があるが Linux にも対応している． Linux 環境は主に実機レスでのフライトソフトウェア開発やデバック環境として使われ，搭載ボードの開発と並行・先行してアプリケーション開発を行うことができる．

(2) プラットフォーム抽象化層

OSAL 層とその上位階層へのアプリケーションインタフェースから成る．そのため OSAL は RTOS 毎に必要となるが，プラットフォーム抽象化層の API は仕様化されているため，異なる OS を使用する場合でも API からは同じ機能を提供することができる．

OSAL が提供する API としては OS 機能であるタスク，キュー管理，排他制御，タイマ管理などがあり，そのほかファイル機能, ネットワーク機能の API が規定されている．ただし使用する OS の制約によっては OSAL API を全てサポートする必要はなく，RTEMS による実装では一部の OSAL 機能がサポートされていない．

また筆者らは TOPEPRS/HRP2 カーネル上で cFE を動作させるために，TOPPERS/HRP2 カーネルと FatFS を用いて OSAL API の実装検討を行なった[7].

(3) アプリケーション層

cFE の API を使ってミッションを遂行するためのフライトソフトウェアのレイヤである．プロジェクトごとに異なるアプリケーションを開発するが，再利用可能なフライトソフトウェアのライブラリとして GSFC がオープンソースとして提供しているものもある（表 2）．

表 2　GSFC が公開している cFS アプリケーション

名称	概要
CFDS	衛星－地上とのファイルデータの送受信
Checksum	メモリ，テーブル，ファイルのデータ整合性チェック
Data Storage	ダウンリンクのための HK，エンジニアリング，科学データを搭載ボードに記録
File Manager	ファイル管理のためのインタフェース
Housekeeping	テレメトリを収集し，再パッケージを行う
Health and Safety	重要タスクの確保, サービス監視, CPU 使用率
Limit Checker	しきい値監視
Memory Dwell	地上からのメモリ内容の測定
Memory Manager	メモリのロード/ダンプ
Software Bus Network	メッセージをネットワーク経由で通信する機能
Scheduler	搭載ボードの起動スケジュール
Stored Command	搭載ボードコマンドシーケンサ

2.3 動的ランタイム環境

cFE API の Executive Service（ES）を用いて，cFS アプリケーションとそのパラメータ設定を行うためのスクリプトファイルを用意し，実行時にそれらのファイルを ES がロードすることで，アプリケーションの起動やパラメータの設定を実現している．

組込みシステム向けのソフトウェアでは OS やアプリケーションは設計・ビルド時にパラメータ情報などをソースコードへ変換し，開発ソフトウェアと共にひとつの実行可能形式（単一リンクモデル）プログラムとした後に搭載することが一般的である．cFS ではソフトウェアの再利用を目的とするため，アプリケーションのロード/アンロードを動的に行うことが可能となっている．しかしモジュールの動的ロードをサポートしていない OS もあるため必須の機能ではない．

Linux 環境を用いた場合では，cFE と OSAP API は ELF 形式のバイナリで実現し，cFS アプリケーションは静的リンクライブラリ形式でビルドし，cFE の Executive Service を用いてロード・実行することでランタイム環境を実現している．

2.4 コンポーネントベース設計

cFS ではアプリケーションの動的ランタイム環境が導入されていることから，プロジェクトに応じて柔軟で入れ替え可能なコンポーネント指向設計に基づいたアプリケーション開発を行う要求が暗黙的に想定されていると思われる．

Vol.2019-ARC-235 No.10
Vol.2019-SLDM-187 No.10
Vol.2019-EMB-50 No.10
2019/3/17

cFS では Software Bus（SB）を用いてその実現を図っている．

cFS でのフライトソフトウェア設計は，プロジェクトで必要な cFS アプリケーションのコンポーネントを検討導入し，プロジェクト独自アプリケーションを加えることでミッションを遂行するためのシステムを構築する（図 2）．SB は導入するコンポーネントのアプリケーション間通信を支えるためのサービスであり，SB を利用することでソフトウェアの凝集度が高くなり，アプリケーションの再利用性を高める効果を得ることができる．

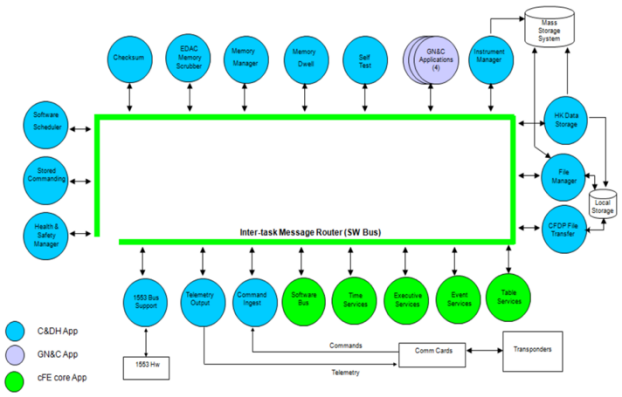

図 2　SB を用いたコンポーネントベース設計の例

3. cFS におけるアプリケーション間通信

3.1 SB を用いたプロセッサ内のアプリケーション間通信

cFS 環境におけるアプリケーション間通信は SB の機能を介して提供している．SB による通信で伝送されるデータを“メッセージ”と呼び，メッセージ本体，メッセージ ID，メッセージ長を管理している．SB は Pub/Sub メッセージングモデルを採用しているが，プロセッサ内通信に限定しているところが他の Pub/Sub メッセージングモデルとは特徴的な点である．

アプリケーションが SB と接続するために，“Pipe”と呼ばれる論理チャネルを生成する．Pipe 生成時にメッセージをバッファリングするためのメモリ確保と購読するメッセージの ID 登録を行う．SB は Pipe 生成時に Pipe とメッセージ ID を管理テーブルへ登録し，出版側から流れてきたメッセージを ID に応じて必要な Pipe へ送信する．アプリケーションからは Pipe への受信処理を行い，目的のメッセージを受信する仕組みとなっている．

3.2 Software Bus Network（SBN）を用いたプロセッサ間のアプリケーション間通信

SB はプロセッサ内のアプリケーション間通信のみサポートしているため，プロセッサを越えた通信を行いたい場合は通信プロトコルを用いた別のサービスを使用する必要がある．

SBN の基本的な動作は， SB で登録された購読情報を他のプロセッサの SBN と共有する．そうすることで個々の SBN はシステム全体の購読情報を保持することができる．購読情報はシステム全体で管理する必要があるため，メッセージ ID はシステムで一意になるように設計する必要がある．SB はメッセージ ID に応じて自プロセッサ内のアプリケーションの Pipe 宛か SBN の Pipe へメッセージを送信する．SBN は送信されたメッセージ ID に応じて，宛先 SBN へ伝送する．このように SB と SBN を組み合わせることでプロセッサ内とプロセッサ間のアプリケーション間通信を柔軟に実現している（図 3）．

SBN では下層の通信プロトコルを特に規定せず，複数の通信プロトコルモジュールがオープンソースとして実装されている．

- TCP
- UDP
- Delay Tolerant Networking（DTN）
- Serial
- SpaceWire

SBN では複数の通信プロトコルモジュールを混在した環境での利用も可能で，プロトコルゲートウェイの役割も果たす．

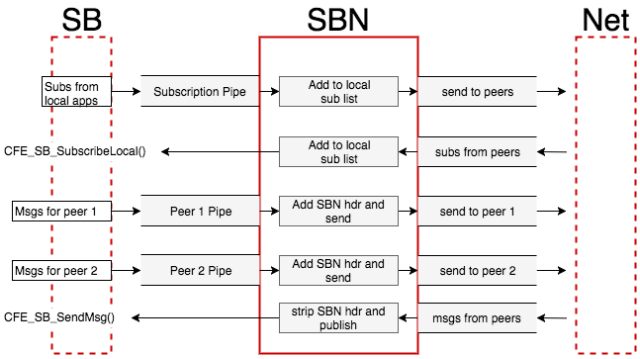

図 3　SB と SBN によるプロセッサ間メッセージ通信例

3.3 cFS のアプリケーション間通信の特徴と課題

SB と SBN を使ったアプリケーション間通信は次のような特徴を持つ．

- Pipe による，アプリケーションと SB 間の通信チャネルの構築
- Pub/Sub メッセージングモデルを用いたメッセージ送受信
- SBN による複数の通信プロトコルのサポート
- 片方向の非同期通信
- CCSDS データフォーマット[8]を用いた地上―衛星間のデータ伝送

想定しているアプリケーション間通信の要求は宇宙デー

Vol.2019-ARC-235 No.10
Vol.2019-SLDM-187 No.10
Vol.2019-EMB-50 No.10
2019/3/17

タシステムでの国際的な標準化活動として行われている CCSDS 委員会からの勧告を元に，cFS でも非同期のコマンドやテレメトリ通信を行うこととして設計されている．そのため次のようなアプリケーション間通信の要求を実現するには SB や SBN の仕組みを見直す必要がでてくる．

- 同期通信
- 通信応答要求
- 遠隔手続き呼び出し
- セキュリティ
- リアルタイム性

これらの要求については想定しているアプリケーションのユースケースが既存のアプリケーションとは異なると考えた．そこで同様の要求に対応するよう標準仕様が策定されている車載のソフトウェアプラットフォームを参考に，cFS のアプリケーション通信機能の改善方法を議論する．

4. AUTOSAR Classic Platform（CP）によるアプリケーション間通信

AUTOSAR Classic Platform（CP）は車載制御向けソフトウェアプラットフォームとして欧州を中心に導入が進んでいる．AUTOSAR CP は cFS 同様にコンポーネント設計を採用し，システムのネットワークトポロジを設計する前に Virtual Functional Bus(VFB)と呼ばれる仮想バスにシステムで必要なアプリケーションを設計し，システム・ネットワーク設計と並行してアプリケーションを先行開発することが可能となっている（図 4）[9].

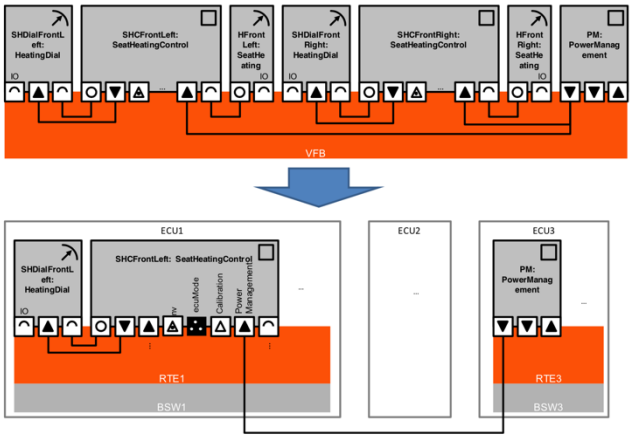

図 4　VFB の通信パスを RTE で実現した場合の例

AUTOSAR CP でのアプリケーション間通信は “Signal” と呼ばれる単位でデータ伝送が行われる．通信方式としては，片方向の Sender-Receiver Communication（S/R 連携），手続き呼び出しの Client-Server Communication（C/S 連携） があり，バッファリングの有無によって API も異なる．また通信応答要求用の通知 API も存在する．

VFB はネットワークトポロジに応じて，プロセッサ内通信，プロセッサ間通信などで実装する必要があり，その実現方法としてはアプリケーションと基本ソフトウェアの間に Run Time Environment (RTE)と呼ぶレイヤがシステムのトポロジや設定情報から通信経路を実現するための実行コードを生成する．通信経路は設計時にあらかじめ決定しておく必要があり，RTE が基本ソフトウェアのレイヤを隠蔽することで，アプリケーションは抽象度の高いインタフェースを用いてアプリケーション間通信の開発を行うことができる．

AUTOSAR CP ではアプリケーションの独立性を高めることに成功しているが，従来型の組込みソフトウェアと同じく OS とアプリケーションが一体となった単一リンクモデルであり，設計時に RTE と接続するための情報を Signal ごとに設定し，かつ基本ソフトウェアで Signal をどのように取り扱うかにいたるまで細かく設定・記述する必要がある．そのためシステムの設計情報を全て把握して記述する上流設計やインテグレーションツールなどの支援が必要となる．

また AUTOSAR CP は CAN 通信を基本とした制御/センサーのデータ伝送が主体であったが，AUTOSAR Adaptive Platform との接続のために Ethernet など長いデータフレームへの対応が行われている[10].

5. AUTOSAR Adaptive Platform(AP)によるアプリケーション間通信

AUTOSAR CP は車載制御システム向けのソフトウェアプラットフォームであるが，高度な運転支援の実現などより複雑で大規模な車載アプリケーション開発に対応するため AUTOSAR Adaptive Platform（AP）が策定されている．

AUTOSAR AP ではサービス指向アーキテクチャ（SOA），並列処理への対応や C++11 の導入が特徴としてあげられる．AUTOSAR CP では OSEK/VDK という車載向けの組込み RTOS が OS 仕様となっていたが，AUTOSAR AP では POSIX PSE51 プロファイルが OS 仕様となっており，Linux など汎用 OS を使ったミドルウェアやサービスの導入を積極的に取り込んで AUTOSAR AP の開発を行っている．

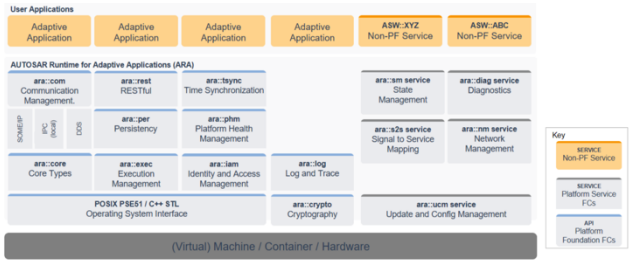

図 5　AP のアーキテクチャ構成図

AUTOSAR AP でのアーキテクチャは，Adaptive Application（AA）と呼ばれるアプリケーションサービス，AP で仕様化されているサービス群（Functional Cluster（FC））および，AA を動かすためのランタイム環境である AUTOSAR

Runtime for Adaptive Application (ARA)から構成される（図5）[11].

表 3　アプリケーション間通信におけるソフトウェアプラットフォーム要件比較

要件	要件項目	cFS	AUTOSAR CP	AUTOSAR AP
機能	メッセージフォーマット	CCSDS	Signal（データフォーマットの規定は無い）	規定なし
	通信方式	単方向のメッセージ通信	・S/R 連携（メッセージ通信） ・C/S 連携（手続き呼び出し）	・Events（メッセージ通信） ・Methods（手続き呼び出し） ・Fields
	同期・非同期通信	非同期のみ	・同期 ・非同期	・同期 ・非同期
	通信応答，タイムアウト	なし，ポーリングのみ	あり	あり
	バッファリング	有無の選択が可能	有無の選択が可能	有無の選択が可能
	接続トポロジ	1:1, 1:N, N:1	0:1, 1:1, 1:N, N:1, N:M	0:1, 1:1, 1:N, N:1, N:M
	ソフトウェアバス	SB	RTE	ARA
	接続設定方式	Pub/Sub メッセージングモデルによる動的設定	ビルド時の静的設定のみ	・Manifest による静的設定 ・サービス指向通信による計画された動的設定
	通信プロトコル	TCP, UDP, DTN, SpaceWire	PDU，SOME/IP	TCP，UDP，SOME/IP，DDS
	通信物理層	Wireless，Ethernet，Serial, SpaceWire	CAN, LIN, FlexRay, Ethernet	Ethernet
非機能	QoS	オプション	なし	なし
	セキュリティ	なし	SecOC	DTLS
	セーフティ	あり	E2E	E2E
	リアルタイム性	低	高	中
	フレームデータサイズ	1 - 64k byte (Space Packet)	0 - 8 byte（CAN）	38 - 1492 byte (Ethernet)
	適用ドメイン	宇宙機（地上-衛星間通信）	車載制御	ADAS，自動運転
	設定記述フォーマット	EDS（XML）	ARXML（XML）	ARXML（XML），JSON

AUTOSAR AP のアプリケーシン間通信は CP と同様に，片方向のデータ通信を行う Events，手続き処理を行う Methods がある．その他遠隔データの読み書きや通知を行う Fields と呼ばれる通信方式がある．通信方法の実現のために Communication Manager（CM）というサービスがあり，アプリケーションは CM のサービスを介してアプリケーション間通信を行うことができる．

AUTOSAR AP は CP と同様に VFB によるコンポーネント設計が可能であるが， CP とは異なりアプリケーションや FC ごとにプロセスを分離し，ランタイム時にプロセス単位での動的ロードが可能となっている．そのため設定記述をサービスごとに分割して設計・実装を行うことができるため，CP よりも設計記述の負荷が軽減できる効果がある．

プロセッサ内のアプリケーション間通信については POSIX PSE51 ではプロセス間通信をサポートしていないため，CM を介した通信を行うもしくはアプリケーションの独自実装となる．

6. cFS のアプリケーション間通信の改善

これまで述べたソフトウェアプラットフォームにおけるアプリケーション間通信の比較を表 3 に示す．

cFS ではセキュリティに関する仕様・実装はないが，CCSDS 委員会において，Security Architecture for Space Data Systems（SASDS）が勧告されている．SASDS では TCP/UDP を想定した暗号化オプションが示されているが，具体的な仕様は決められていない．

cFS では複数の通信プロトコルをサポートしているため，SBN 間を SSL/TLS や DTLS といった暗号化プロトコルを用いることで通信プロトコル上の伝送データの秘匿を行うことが可能となる．しかし，SBN 間での通信ごとに暗号化のハンドシェイクによる通信オーバーヘッド，リソースの確保やセッションライフタイムなどが課題となる．

ソフトウェアバスによるコンポーネント指向設計の実現により，製品出荷後のソフトウェア更新も可能となる．AUTOSAR CP ではメインテナンス時に Diag によるファイルの書き換えが可能で，AUTOSAR AP では Update and Configuration Management（UCM）サービスを用いて実行時でのアプリケーション更新も実現可能となっている．cFS では表 2 にある CFDS を用いてファイルのアップロードが可能であり，アップロードされたアプリケーションと EM を連携してソフトウェア更新を行うことが考えられるが，UCM で検討されているアプリケーションの切り替え方法を参考に宇宙分野での要求を加味して実装検討を行う必要がある．

Vol.2019-ARC-235 No.10
Vol.2019-SLDM-187 No.10
Vol.2019-EMB-50 No.10
2019/3/17

7. まとめ

cFS のアプリケーション間通信の機能を持つ SB や SBN は，CCSDS による地上-衛星間通信設計を踏襲し，片方向による非同期通信，遅延耐性に強い通信プロトコルも利用することを想定して設計されている．しかし衛星内通信でのアプリケーションでは遠隔デバイスに対するメモリアクセスや手続き呼び出し，動画像といった高スループットなデータ転送などの要求があり，衛星内通信プロトコルである SpaceWire を使ったミッションアプリケーションが存在している．

そこで cFS と同様にソフトウェアバスを有し，近いアーキテクチャ構成である AUTOSAR CP と AP のアプリケーション間通信について調査を行い，cFS との比較を行った．

cFS のアプリケーション間通信の改善提案として，通信路のセキュリティやソフトウェア更新などの非機能要件に対するアプリケーションの検討を行った．また同期通信，遠隔デバイスへのメモリアセス，手続き呼び出しや通信応答要求などといった機能面でのアプリケーション要求に対しては，これらの要求の実現可能性について実装検討を行っていきたい．

参考文献

[1] “CubeSat”. http://www.cubesat.org/, (参照 2019-02-10).
[2] “AUTOSAR”. http://www.autosar.org/, (参照 2019-02-10).
[3] “cFS”. https://coreflightsystem.org/, (参照 2019-02-10).
[4] ESA. SpaceWire – Links, nodes, routers and networks , ECSS-E-ST-50-12C, 2010.
[5] “OS Abstraction Layer”. https://sourceforge.net/projects/osal/, (参照 2019-02-10).
[6] David McComas. cFS Introduction. FSW2018.
[7] Mitsutaka Takada, Hiroaki Takada. Porting cFE to spacecraft onboard with SpaceWire Engine and RTOS based on uITRON specification. FSW2017.
[8] CCSDS. Space Packet Protocol. CCSDS 133.0-B-1, 2003.
[9] AUTOSAR. Virtual Functional Bus. AUTOSAR CP Release 4.4.0, 2018, 31p.
[10] AUTOSAR. Specification of Large Data COM. AUTOSAR CP Release 4.4.0, 2018.
[11] AUTOSAR. Explanation of Adaptive Platform Design AUTOSAR AP Release 18-10, 2018, 14p.