C-022

人工衛星向け組込みソフトウェアアーキテクチャの検討
Study of Embedded Software Architecture for Satellites

荻野 慎平† 加藤 寿和† 平山 芳和†
Shimpei Ogino Toshikazu Kato Yoshikazu Hirayama

1. はじめに

近年、人工衛星は多様化する顧客のニーズに対応するため、搭載ソフトウェア（OBS：On-Board Software）に求められる機能が増加し、ソフトウェアの開発規模が拡大している。その結果、ソフトウェアの構造が複雑化し、従来と同等の品質を維持するために多大な労力が必要となっている。一方、SpaceX の Starlink 衛星に代表されるような、複数の衛星を一体化して軌道上で運用する衛星コンステレーションが日本でも検討されており、今後打ち上げ機数の増加が見込まれる。このため、1 機あたりの開発コストの低減や工期短縮が求められている。

本論文では、これらの課題を解決するために、人工衛星に搭載されるハードウェアおよびソフトウェアの構成と、それらを適用することによって得られる効果について検討した結果を示す。

2. 人工衛星向け組込みソフトウェア開発の課題

人工衛星の OBS 開発における課題は、以下の 3 点が挙げられる。

①ニーズの多様化・開発規模増・複雑化

衛星ごとに異なる要求に対応するため、ソフトウェアおよびハードウェアに複数のバリエーションが発生している。そのため、流用性の低下や構成管理の複雑化により、開発コストの増加につながりやすくなっている。

②低コストかつ高速な開発

競争力向上のためには、低コスト・短納期化は必要な要素の一つである。そのためには、ハードウェアの製造コストを下げ、要求の変更に対してはソフトウェアで対応できるようにスケーラビリティを持たせる必要がある。

③限られた計算機リソース

人工衛星に搭載される CPU やメモリは、宇宙の放射線や温度差などの厳しい環境に耐えるため、民生品に比べてリソースが限られている。性能を満足させるために、限られたリソースで効率的に動作するアルゴリズムの設計や最適化技術の導入が求められるため、コストが増大している。

従来の開発では、ベースライン機種を再利用して開発してきたが、①および②の課題に対応するためには、同様のアーキテクチャでは限界があるため、新たなアーキテクチャを検討した。上記の課題を踏まえ、アーキテクチャに必要な要件を図 1 に示す通り整理した。

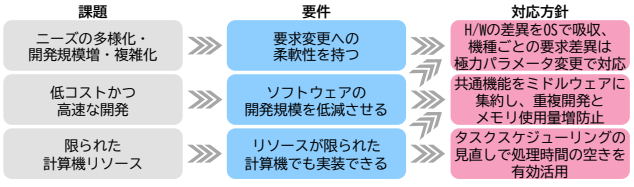

図 1　アーキテクチャの要件

† 三菱電機株式会社　Mitsubishi Electric Corporation

3. 衛星の制御器の構成

3.1 ハードウェア構成

人工衛星に搭載されるメイン制御器の構成を図 2 に示す。大きく分けて「計算機ボード」、「データ処理ボード」、「インタフェースボード」の 3 つから構成され、ソフトウェアは計算機ボード上の ROM に搭載される。各ボード間は SpaceWire という宇宙用の通信規格で接続されており、それぞれのボードの役割は以下のとおりである。

①計算機ボード

計算機ボードにはソフトウェアが搭載されており、姿勢・軌道制御や衛星と地上間のデータ分配などのメイン制御を行う。CPU やメモリは厳しい宇宙環境に耐える部品や機能を備えている。

②データ処理ボード

地上局との通信データ（コマンド(*1)・テレメトリ(*2)）などのデジタルデータを外部の機器とインタフェースする。

(*1) コマンド　：地上局からの制御指示

(*2) テレメトリ：地上局への衛星状態の送信

③インタフェースボード

温度、電圧値などのアナログデータを外部の機器とインタフェースする。オプションで複数枚を搭載可能である。

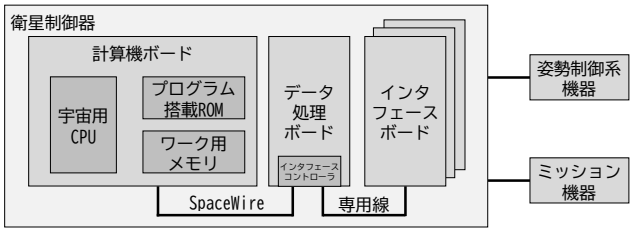

図 2　衛星制御器の構成

3.2 ソフトウェア構成

2 項で示した課題を解決するためのソフトウェアアーキテクチャの検討結果を以下に示す。アーキテクチャは、ソフトウェアの構造とスケジューリングの観点から検討した。

(1) ソフトウェアのレイヤー構造

従来のソフトウェアのレイヤー構造と改善後の構造の比較を図 3 に示す。従来の構造はアプリケーションとファームウェアのみで構成されていたが、改善後の構造ではミドルウェアとリアルタイム OS を導入した。各レイヤーの役割と開発コンセプトを表 1 に示す。改善後の構造では、ソフトウェアの要求変更に対してアルゴリズムを変更するのはアプリケーションレイヤーのみとし、内部および外部のインタフェースの変更はミドルウェアで具備しているパラメータの変更のみで対応する。また、共通的な機能は可能な限りミドルウェアレイヤーに集約し、異なるアプリケーション間で同一機能の複数の実装バリエーションを抑制することで、開発コストの低減と品質の向上を目指す。

表 1　ソフトウェアの各レイヤーの役割・開発コンセプト

レイヤー	役割・開発コンセプト
アプリケーション	各衛星の固有の要求を実現するためのレイヤー。共通的な処理をミドルウェアに移管することで、アプリケーション間での重複した作りこみを抑制する。基本的に機種ごとに変更するのはこのレイヤーのみである。
ミドルウェア	
データ処理(機種共通)	コマンドの受信やテレメトリの送信など、データ処理のアプリケーションのうち、機種に依存しない処理を担当する。
ソフトウェアバス	アプリケーション間、およびアプリケーションとミドルウェア間のインタフェースを提供するレイヤーである。各アプリケーションやミドルウェアはデータを配置する ID のみを知っていればよく、アドレスを意識する必要はない。
ネットワークコントローラ	外部との通信を制御するレイヤーである。アプリケーションは衛星制御器と接続された機器との通信プロトコルを意識することなくデータを送受信できる。
ライブラリ	各アプリケーションやミドルウェアで共通的に利用できる処理を集めた API 群である。行列・ベクトル計算、メモリ制御や衛星の制御に必要な計算処理を提供する。
リアルタイム OS	ハードウェアの差異を吸収し、上位のレイヤーに影響を与えないようにする。通信ペリフェラルドライバを有し、ネットワークコントローラから利用される。

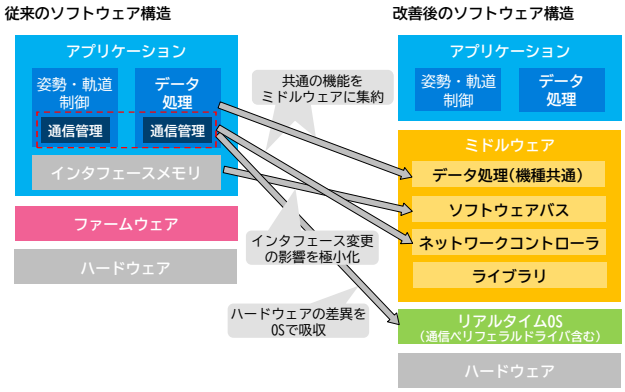

図 3　従来と改善後のソフトウェア構造の比較

(2) スケジューリング

衛星搭載ソフトウェアは、定期的な割込みにより周期動作(処理のスケジューリング)を行う。従来のスケジューリングと改善後のスケジューリングを図 4 に示す。従来のスケジューリング方法では、システムメジャー周期(一連の動作を完了する周期)を複数の処理スロットに分割し、各ソフトウェアの処理を割り当てていた。そのため、各処理はスロットに割り当てられた時間内に完了させる必要があり、処理時間の猶予がないスロットが存在する一方で、逆に時間が余って有効活用できていないスロットもあった。このスケジューリング方式のデメリットとして、処理時間の猶予がないスロットに対して機能拡張を行う場合、コードのリファクタリングをして処理時間を削減するか、処理時間に余裕がある他のスロットに処理を移動するなど、改修に時間を要していた。

改善後のスケジューリングでは、固定のスロット割当てを廃止し、スケジューリングタスクを導入した。本スケジューリングでは、処理時間が短いスロットの空き時間を有効活用し、CPU 全体の処理効率が向上した。その結果、今までスロットの規定時間内で完了させるための処理の分割・スロット移動が不要となり、機能拡張に対して柔軟に対応可能となった。

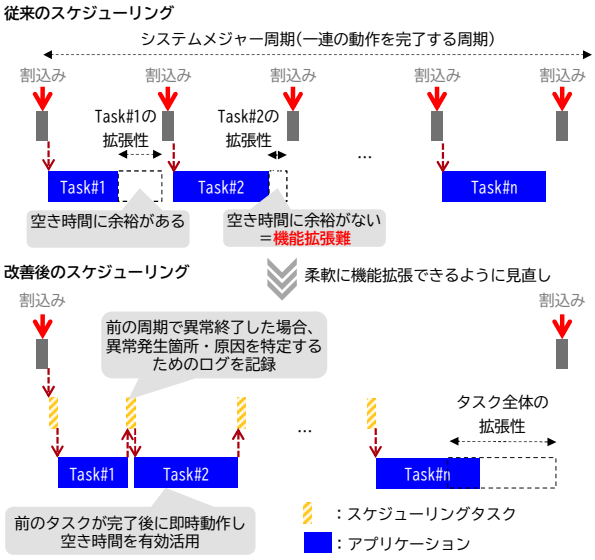

図 4　従来と改善後のスケジューリングの比較

4. 効果

本論文に示したアーキテクチャにより、2 項に示した課題①～③を解決した。また、QCD の観点から以下の効果が得られると試算している。

Q　：共通のミドルウェアを全機種で適用することで、技術が成熟し、品質が向上

CD：ソフトウェアの変更をアプリケーションのみに集約することで、開発工数・工期を約 20%削減

5. おわりに

本論文では、2 項で示した 3 つの課題である「ニーズの多様化・開発規模の増大・複雑化」、「低コストかつ高速な開発」、「限られた計算機リソース」を解決するために、新たなソフトウェアアーキテクチャを構築した。今後は、このアーキテクチャを適用する機種を拡大し、衛星開発の低コスト化および高速開発に寄与することを目指す。

参考文献

[1] 吉田 誠, 深川 周和, 石井 明彦, 石島 義之, 艸分 宏昌, “静止衛星”DS2000”搭載用機器”, 三菱電機技報, Vol.85, No.9 (2011).

[2] Akira Chiba, Isao Odagi, Shinya Hirakuri, Takahiro Owaki, Shimpei Ogino, Kazuhiro Nishikawa, “SpaceWire network and communication technology in the MMX system”, International SpaceWire Conference 2022 (2022).