開発中および検討中の科学ミッション：P2-182

DESTINY

工学実験：スクリプトによる自律運用に必要な計算機リソース

福島洋介 宇宙航空研究開発機構宇宙科学研究所<fukushima@isas.jaxa.jp>
川勝康弘(ISAS/JAXA)

宇宙機運用における効率化とは何をさすのだろうか

こんにちは、福島洋介です。宇宙研で助教をしています。(本人は近くにいるはずです。声をお掛けください)

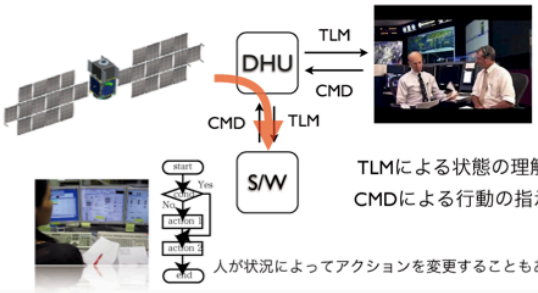
ここ数年続けてこのシンポジウムではDESTINYミッションの工学実験テーマの一つである「自律化」を紹介しています。 これまでのISAS科学衛星に「自律判断機能」を実装するためのベースとなる技術の提案です。とはいえ、この提案では「自律」という単語の意味を、SF的な(楽しい)方向へ議論が発散してゆくことを避け、現実的な(どちらかと言えばワクワクしない)意味に限定しています。それは、深宇宙探査では避けられない地球と通信がとれない、ないしは、非常に困難な状況下」での宇宙機の「自律運用比率」の向上させること。

運用の効率　実施する運用内容（やれること）——増やす / 運用に必要なコスト（費用・時間・作業）→減らす

運用の自律率　実施する運用内容（やれること）——増やす / 運用者（人）が考え・作業する量→減らす

要するに、運用者の作業量を減らすこと
(運用者よりも運用装置が、それよりも搭載計算機が作業を担えばよい)

自律運用の定義を確認してみましょう。
- その時点でのミッション計画の「目的を満足させ」るような「行動の選択肢を決定する」つまりコマンドを「選択する」ことです。
- そして、「システムの自滅にながらない」ように、宇宙機状態を監視することです。



TLMによる状態の理解
CMDによる行動の指示

人が状況によってアクションを変更することもある

1. 運用が「結果的に」自律化される

宇宙機運用の現実の作業内容を観察しますと、運用者は運用前に作成された「運用手順(電子化されている)」に沿って作業をしています。(その場で判断や思いつきは、まず(緊急時以外には)ないです)
自律化するには、まず運用手順の表現方法を「人が理解する」から「宇宙機搭載計算が理解する」に変更します。
要するに、コマンドのリストからプログラム(ここでは運用スクリプトEOS)に変更するわけです(右図)。

従来型のコマンド表記 (command sequence)　提案する表記 (operational script)

「高度化された運用システム」とか「エキスパートならではの支援」というものは、標準となるEOSにどれだけ枝分かれしたパスを内包させておくか、でしかない。

宇宙機においてEOSで記述されたシーケンス通りに枝分かれして実行されたなら、宇宙機は結果的に(従来の運用と比較して劇的に)必要な「判断をした」つまり、自律化されたことになります。

従来の自律化研究は、結局のところ「データから特徴量を抽出する」ことでした(認識、学習など)。
それらの研究成果は「機能部品」として宇宙機に組み込まれ、その機能の良し悪しが議論される。

この提案がそのような従来の自律化研究と違うところは、機能ではなく、機能を実現させる「仕組み」に着眼点を置いているところです。

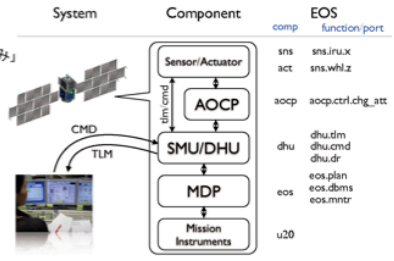
だから、EOSを使えば、過去に実施された自律化研究の成果の多くを「取り込め」ます。

EOSでは宇宙機を構成するコンポーネントごとに対応する「オブジェクト」を準備し、それらの振る舞いをスクリプトで記述すればよい。

そして、それらは「運用場面ごとに変更」される。

Encapsulated Operation Script: EOS

運用の表現方法をEOSへ



コンポーネントに対応したEOS

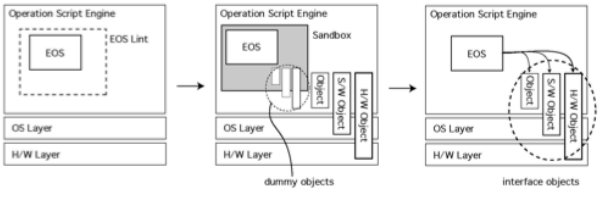
新しい機能を作り込んで、はい終わり、ではありません。
運用ごとに細かく調整できなければ、実際上実用には供せないからです。
衛星が運用される間ずっと新規機能をEOSで記述して宇宙機に送信し、試すことが続けられます。

2. 運用スクリプトは実用レベルにまでなりうるか？

運用手順を手続き型のプログラミング言語で表現し、それを宇宙機上で「実行」する。それができれば、運用のかなりの部分を自律的に実行できます。故障検知やIF-THENを含む運用も運用者の介在なしで実行できる可能性があります。

可能性の追求ではなく、実運用に使えるかどうか決める一つの基準は安全性です。スクリプトは運用期間中ずっと新しいものが作られ続けるので、DESTINY打ち上げ前に確認してない。スクリプトが悪さしないことをどう担保するのか？

方法は2つあります。一つはH/Wへのアクセスを伴う処理を一旦「大丈夫か確認」してから実施する方法。もう一つは、スクリプトが健全に機能しているかどうかを定期的に自己確認する方法。



オブジェクトアクセス時にWrapperをつかう方法

チェック用のファンクションをコールしてもよいが、スクリプトのevalやloadという組み込み機能を使って状況に合わせたチェック機能を実行時に自己改変することも可能(埋め込むことも可能)。

eval を使う方法

```
do {
 eval(stub_string_pos1);
 ti = eos.twai_ti();
 eos.sns.update();
 var q = eos.att.q();
 eos.dbms.set(ti, q);
 eval(stub_string_pos2);
} while (util.dif_angle(util.qsub(q, qref)) > ref_angl);
```

load を使う方法

```
do {
 if (check_pos1) { load('pos1.js'); }
 ti = eos.twai_ti();
 eos.sns.update();
 var q = eos.att.q();
 eos.dbms.set(ti, q);
 if (check_pos1) { load('pos2.js'); }
} while (util.dif_angle(util.qsub(q, qref)) > ref_angl);
```

3. 運用スクリプトを走らせられる計算機はあるか？

スクリプトエンジンとしてはT-Kernelで動作検証していますが、これがそのままDESTINYで動作するわけではありません。ミッションデータ処理用の計算機(MDPまたは単にOBCと呼ぶ)でスクリプトエンジンを実行させることになっていますが、それが可能かどうかは処理能力次第です。
ではどのくらいの処理能力があればよいのか？

実機用OBCについて詳細に検討することは現時点では難しいです。とはいえ、現実問題として可能なOBCの候補は少ないです。現時点では、候補となりうるOBCの処理速度を調べ、同程度の処理能力を持つ計算機上でスクリプトエンジンを含めた各種能力チェックを行い、ここでの提案が機上の空論化することを回避しています。

小型科学衛星バスで準備されているOBCや小型衛星ほどよしで採用されているOBCが候補となりますので、「同程度」のCPUボードでスクリプトエンジンを動作させています。しかし、CPUレベルとボードレベルでMIPSは異なりますし、キャッシュやメモリ、あるいはOSやライブラリによっても挙動が変わります。なので、あくまでも参考程度の情報にしかなりませんが、現時点で「能力的にスクリプトは動作不可能」となるものはありません。

OBCの処理能力

	System	Clock Hz	MIPS	OS
小型科学衛星バス →	HR5000S(SOI)	50M	80	
	HR5000	2G	320	
検討で使用 →	SH7727	60M	60	T-Ker
検討で使用 →	ARM920	180M	200	T-Ker
	SH4-7760	200M	360	Linux
小型衛星ほどよし →	HODOYOSHI-OBC(SOI)	62.5M	96	TOPPERS

宇宙機のOBCで利用できるOS次第というところがある。
TOPPERSへの移植は、まともなFileSystemがないので苦戦しています。

今後の活動

搭載系で実現可能なスペックの計算機と運用を模擬できる程度の宇宙機・外部環境シミュレータを準備し、EOS利用のメリットをより理解されやすい表現形式で紹介しつづける。

This document is provided by JAXA.