開発中および検討中の科学ミッション：P2-138

DESTINY
工学実験：運用の自律化・効率化のモデル例

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>
山田隆弘 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)

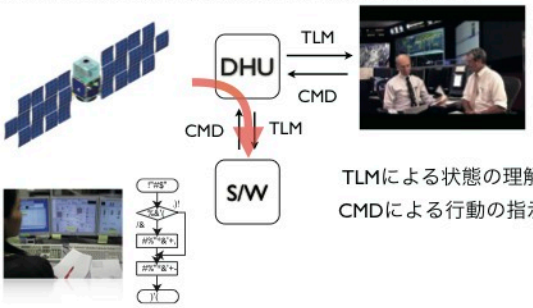
昨年に引き続き、このポスターでは小型科学衛星シリーズの標準衛星バスを使ったDESTINYミッションに対して、従来の宇宙科学研究所の宇宙システムでは実現不可能な宇宙機の自律化技術を提案します。
この提案では「自律」という単語の意味を、SF的な(楽しい)方向へ議論が発散することを避け、現実的な(どちらかと言えばワクワクしない)意味に限定し、深宇宙探査では避けられない「通信がとれない、ないしは非常に困難な状況下」での宇宙機の自律運用比率の向上を目的としています。

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

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)に変更します(右図)。

「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、です。

運用中に必要な「判断」がEOSで記述された通りにシーケンスの枝分かれに沿って実行されたなら、宇宙機は結果的に(従来の運用と比較して、劇的に)自律化されたことになります。

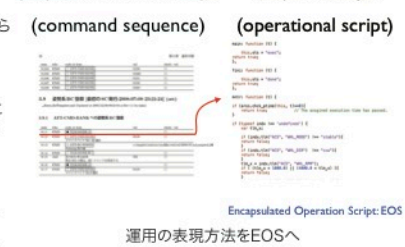
従来の自律化研究は、結局のところ「データから特徴量を抽出する」ことでした(認識、学習など)。
研究成果は「機能」として組み込み、良し悪しを確認することになります。

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「構造」に着眼点を置いているところです。

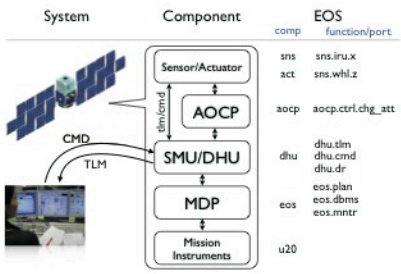
だから、EOSを使えば、過去に実施された自律化研究の成果の多くを「取り込め」ます。

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します。

そして、それらは「運用ごとに変更」されます。



運用の表現方法をEOSへ



コンポーネントに対応したEOS

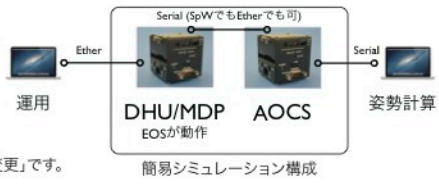
機能を作り込んで終わりではありません。
運用ごとに細かく調整できなければ、実際上実用には供せないからです。

2. この仕組を使った場合、例えば何ができるのか？

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。

EOSの優位性を示すために、現行の方法では不可能だけど、EOSで可能となるものの一例を紹介します。
それは、軌道上での「運用内容の動的変更」です。

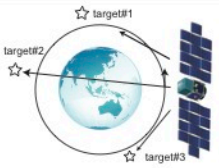


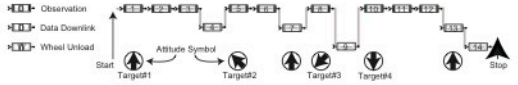
簡易シミュレーション構成

現状動作するEOSの一例

```
// Entry point
var ref_ang = 0.1*0.0174533;     // 0.1[deg] 目標姿勢との誤差設定角度
var ref_omg = 0.0174533;         // 1[deg/sec]の角度速度許容
var qref    = {0.707, 0, 0, 0.707}; // reference attitude  目標姿勢
load('util.js');                 // loading utility functions  計算数式のスクリプト
load('cnt_default.js');          // loading default control raw  よく使う制御ロジックのスクリプト
var eos = new Core();            // DESTINYシステムのオブジェクトと生成
eos.open();                      // そのオブジェクトを初期化（ここでコンポーネントI/Fを起動）
CMD eos.aocp.ctrl.chg_att.start(qref, ref_omg);  // qrefとなるようref_omgの角速度で姿勢変更を開始する
var ti; // 衛星時刻を代入する変数
do {
TLM  ti = eos.twai_ti();         // 衛星時刻が更新されるまで待つ。更新された時のTIを代入
TLM  eos.sns.update();           //  センサ系の数値をすべてアップデートする
  var q = eos.att.q();           //  姿勢値の値を読み取る
  eos.dbms.set(ti, q);           //  時刻タグと姿勢値とをDBに登録する
} while (util.dif_angle(util.qsub(q, qref)) > ref_angl);  // 目標姿勢値との差を角度計算し条件判定
CMD eos.aocp.ctrl.chg_att.stop();  //  制御を停止
CMD eos.dhu.dr.report(eos.dbms.query('average', q)); //目標設定値の近傍の姿勢だったのかをレポート
eos.close();                     //  オブジェクトを破棄する
// End of EOS
```

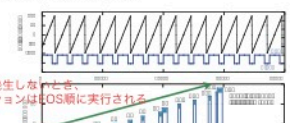
運用スクリプトのJavaScript表現の一例



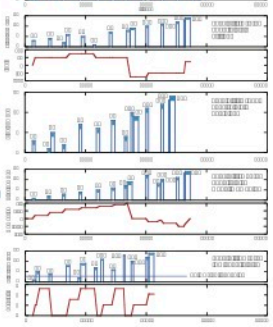


例として地球周回する衛星が、1)観測と2)観測データのダウンリンクと3)ホイールのアンローディングを実施する簡易モデルを考察します。基本的に順次実行するEOSを作成し、実行しますが、途中で異常イベントを発生させ、そのままでは運用が継続できない状態をシミュレーションします。運用は14個のEOSで構成され、搭載系に送信されました。以下が結果です。

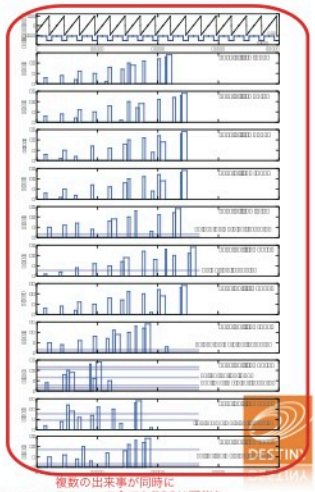
運用中にイベントが発生しない場合は、EOSは順番に実行される。



イベントが発生するとそのままでは運用系は継続できない。
通常は、運用は自動中止し、運用者が修正。



EOSの場合は、中止しないで、「可能な」運用を自律検索し、実行し続けることが可能。



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。

最後まで目を通していただいた方にはお礼を申し上げます。ありがとうございました。