第5章

宇宙機器に学ぶ安全なソフトウェアの作り方

片平真史
石濱直樹

安全意識と誇りを持って設計・検証に取り組む

ここでは，宇宙分野における信頼性向上や安全性確保のための考え方と，宇宙航空研究開発機構（JAXA）で実施している安全のための取り組みについて解説する．（編集部）

宇宙分野にも幅広い種類のソフトウェアがあります．例えば，人工衛星（**写真1**）に搭載されるソフトウェアは，規模や機能の面で家電製品などに組み込まれるソフトウェアと似ています．そして，開発の課程において抱えている問題点も，ほかの組み込みソフトウェアとよく似ています．

一方，宇宙分野のソフトウェアは以前から高い信頼性や安全性を要求されてきており，その点でいろいろと工夫を重ねています．ここでは，宇宙分野において取り組んできた手法や考え方をベースに，ソフトウェアの安全性を高める方法について解説します．

写真1　国際宇宙ステーション（完成イメージ図）

1．信頼性向上と安全性確保の違いを理解する

人工衛星などの宇宙機器は，宇宙という特殊環境で使用する（例えば，宇宙放射線の影響により，コンピュータのデータにビット化けが発生する）ため，宇宙用に開発された耐放射線性能の高いCPUを搭載しています．ただし，宇宙用CPUを使用したとしても，ソフトウェアにおけるビット化けなどの影響を考慮して設計する必要があります．

ソフトウェアで故障耐性を確保するための手法として，Nバージョン・プログラミング（同じ処理を異なるアルゴリズムで複数実装しておき，並列処理して計算結果を多数決する）[注1]というものがあります．この有効性を報告する論文もある一方，開発コストの増加や，多数決で計算結果を選択することによって一貫性がなくなり，新たな問題を引き起こす可能性がある，などと指摘されています[(1)(2)]．

スペース・シャトルの設計においては，Nバージョン・プログラミングのほかに，それらを監視して異常を検知する役割を持ったコンピュータとソフトウェアを用意するなどの方法も採られています．

また現在は，安全にかかわる複数の制御系や監視系を，1台のコンピュータに搭載されたソフトウェアで処理する場合があります（右掲のコラム「宇宙ステーション時代になって認められたソフトウェアの冗長性」を参照）．このよ

注1：IEC 61508で紹介されている diverse programming の一つの技法を指す．

KeyWord　Nバージョン・プログラミング，高信頼性検証，FTA，IV&V，SPiCE for Space，TOPPERS/HRP，安全確保

うな状況では，リアルタイムOSにおける安全性確保が重要となります．安全解析や安全設計のためにOSの内部詳細情報を分析する必要があるため，宇宙航空研究開発機構（JAXA）では，リアルタイムOSの高信頼性検証プロセスを独自に策定するとともに，オープン・ソース・ソフトウェアであるTOPPERS OSをベースとして安全機能を強化したリアルタイムOSを開発しています（p.58のコラム「リアルタイムOSの信頼性を検証する」を参照）．

●「信頼性が高い＝安全」ではない

製品の信頼性が向上することは，出荷後のリスクを低減するために大変重要です．また，製品（ソフトウェア）をテストすることにより，製品の信頼性を向上できることも自明の理です．ところで，製品の安全に対しては，信頼性を向上するだけで十分なのでしょうか．

意図的に事故を起こすような製品を作る人はいません．準備された機能が想定外の動きをしたときや必要な機能が欠落していたときに，それが大きな事故につながります．従って，要求された機能が想定された条件のもとに動作するだけではだめなのです．事実，1996年に起きた，ESA（European Space Agency；欧州宇宙機関）のアリアンV型ロケットの事故（異常検知により自動爆破した）は，設定された機能は求められた通りに動作しましたが事故に至りました[注2]．

製品そのものにどのような潜在的危険があり，その製品に組み込まれたソフトウェアがどのように影響する可能性があるのかを，安全性という観点からしっかりと分析する必要があります．

宇宙分野においては，以下のような安全性の基本方針があります．

- どのような故障が二つ重なって発生しても，人を殺さない．製品を喪失しない

A. 直接原因となる制御系
機能そのものが危険な事象に影響する機能
ケース1） 機能不動作が人を殺す，負傷させる
- 必要なときに動作しない
- 動作中に不意に止まる
- 停止時に想定外の止まり方をする

ケース2） 機能動作が人を殺す，負傷させる
- 動いてはいけないときに動き出す
- 停止しなければならないときに止まらない
- 動き出すときに想定外の動きをする

B. 安全監視系
危険な事象を監視・検知し，抑制する機能

図1　ソフトウェアが関係する安全機能
ソフトウェアは製品に組み込まれることにより，危害を及ぼす可能性が発生する．特に，制御系や安全監視系の機能を担ったときに，安全にかかわってくる．

- どのような故障が（単独で）発生しても，人を負傷させない．製品の重要な機能を喪失させない

ソフトウェア安全性の規格も，この基本方針に従って整備されています．これらの前提の下，ソフトウェアの特性を考慮して，安全確保に取り組んでいます．

あえて言うまでもありませんが，ソフトウェアだけでは人を殺したりはしません．ソフトウェアがシステム（製品）に組み込まれて初めて，その危険性が発生します．では，ソフトウェアがどのような形で製品全体の安全性に影響するのでしょうか．具体的事象は，業種や製品によってまちまちですが，一般的には**図1**に示す機能について，安全を考慮する必要があると考えられます．

なお，宇宙ステーション計画における安全要求「SSP50038（宇宙ステーションのコンピュータ制御システムに対する安全要求）」[(3)]にも，**図1**のA（直接原因となる制御系）に相当する項目があります．Must Work Function（ケース1）とMust Not Work Function（ケース2）を識別し，安全要求を適用するように規定されています．

注2：当初の想定と異なる利用環境（再利用）で動作したため，事故に至った．もちろん，再利用時の検証不足も問題として指摘されるべきだ．

COLUMN　宇宙ステーション時代になって認められたソフトウェアの冗長性

スペース・シャトルを設計していた時代（1970年代後半～1980年代後半）には，1台のコンピュータ上のソフトウェア機能として二つ以上の冗長機能を実装することは，安全設計上，許容されていませんでした．しかし，コンピュータの能力向上などにより，宇宙ステーションを設計する時代（1980年代後半～現在）になってからは，1台のコンピュータ上のソフトウェアで二つ以上の制御系や監視系の処理を行うことが許容されるようになっています．

この際に重要なのは，ソフトウェアの処理の独立性（リアルタイムOSレベルまで含む）を確保することです．いかなる欠陥が発生しても，同時に二つ以上の処理に影響しないことを保証する必要があります．