宇宙システムにおける上流工程仕様の妥当性確認技術

独立行政法人　宇宙航空研究開発機構　情報・計算工学センター
安全・信頼性推進部 上席開発員 技術領域総括

片平　真史

1 概要

宇宙システムに代表されるセーフティクリティカルシステムでは、要求された機能が諸条件の元で動作するという信頼性の向上のみではなく、安全性の視点でもその健全性の検証が求められる。特に、ソフトウェア開発において、仕様どおり動作することの保証は、従来の品質工学、信頼性工学に基づく手法により強化が可能であるが、要求仕様等の上流工程における仕様定義時の正確性の欠如、不完全性の残留、想定不足がシステムの安全性に脅威をもたらすことが多い。

宇宙分野でも、NASA 等の過去の事故事例からの教訓として、上流工程の仕様に関する問題に起因した事象が注目されている。JAXA（独立行政法人 宇宙航空研究開発機構）では、この解決のために、ソフトウェア IV&V（独立検証および妥当性確認）活動として、「モデル検査」や過去経験知に基づく「チェックリストベースレビュー」を導入している。

宇宙業界標準の開発プロセスとして、JAXA では、ソフトウェア開発標準（JERG-0-049）を定めており、その中で、妥当性確認を表 1 のとおり定義している。

図 1 に示すとおり、検証はあるリファレンスに基づいて評価されるものであるが、妥当性確認は、顧客や想定環境から抽出された上流要求に対し、開発全工程を通じてその成立性を確認するものである。ここでは、妥当性確認の視点で導入してきた「モデル検査」および「チェックリストベースレビュー」の取組みをその目的や期待効果とともに解説する。

表 1　検証および妥当性確認 (JERG-0-049 から引用)

用語	説明
検証	客観的証拠を提示することによって、規定要求事項が満たされていることを確認すること。
妥当性確認	客観的証拠を提示することによって、特定の意図された用途または適用に関する要求事項が満たされていることを確認すること。

2 取組みの目的

ソフトウェア開発の上流工程における要求仕様に関する問題に起因して、宇宙分野では以下の事故が発生している。

- NASA（アメリカ航空宇宙局）が 1998 年 12 月に打上げた火星探査機 MCO（マーズ・クライメイト・オービター）は、1999 年 9 月 23 日に火星軌道進入の際に、通信が途絶えた。事故調査の結果、予定されていた 140-150 km よりも低い 57 km の軌道で侵入してしまったことが主原因とされたが、これは、軌道モデルで使用されるソフトウェアの単位系の取り違え（メートル法のはずが誤ってヤードポンド法となっていた）により発生したとされている。**＜正確性の欠如＞**
- NASA が 1999 年 1 月に打上げた火星探査機 MPL（マーズ・ポーラー・ランダー）は、1999 年 12 月 3 日に、火星大気圏へ突入する際に、探査機からの信号が途絶え、行方不明となった。信号途絶の理由は現在でも不明であるが、最も可能性の高い原因として、着陸用の脚の展開時の衝撃を着陸と誤って判断したソフトウェアの問題とされている。また、この根本原因として、センサの振舞いに関する情報がソフトウェア要求仕様に欠落したと言われている。**＜不完全性の残留＞**
- ESA（欧州宇宙機関）が 1996 年 6 月 4 日に打上げたアリアン 5（Ariane 5）初号機は、打上げ後 37 秒後に予定していた飛行経路を大きく逸脱し爆発した。

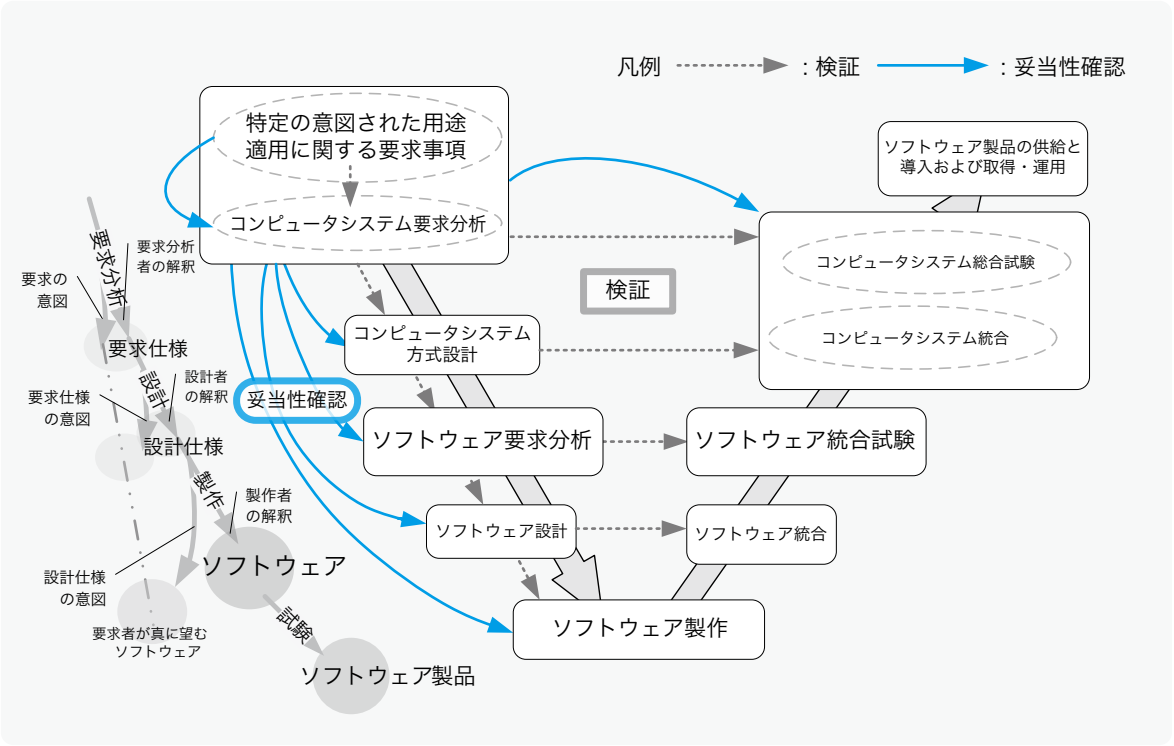

図１　検証と妥当性確認の概念図　(JERG-0-049 から引用)

事故調査の結果、一つの主要要因はアリアン４から再利用されている実行されないはずのソフトウェアが動作したためである。破壊に至った一つの副要因として、想定外のセンサの出力に対する処理が欠落していたことが挙げられている。**＜想定不足＞**

本事例で紹介する取組みは、これらの過去の事故・重大不具合の経験に基づき、ソフトウェア開発の上流工程における要求仕様等の仕様定義時に、正確性の欠如、不完全性の残留、使用方法・環境の想定不足を極力排除し、セーフティクリティカルシステムの安全性を向上させ、事故を未然回避することを目的としている。

3 取組みの対象、適用技術・手法、評価・計測

3.1. 取組みの対象製品と工程

ここでの取組みは、宇宙システムの中でも、変更が困難であり、また、短時間でセーフティクリティカルな判定が必要となる「人工衛星システム」や「ロケットシステム」に用いられる搭載ソフトウェアを対象として紹介する。

また、ここで紹介する取組みは、上流工程の仕様定義時の妥当性確認に限定しているが、実際の開発作業においてはそれ以外の工程にも適用されている。

3.2. 適用技術・手法

ソフトウェア開発の上流工程における要求仕様等の正確性の欠如、不完全性の残留、使用方法・環境の想定不足を排除するために、適用した技術・手法である「モデル検査」および「チェックリストベースレビュー」をここでは紹介する。この「モデル検査」および「チェックリストベースレビュー」は、技術・手法として、それぞれのデメリットを補完しながら使用している。通常 IV&V では、目的（観点）に対して、図２に示すように

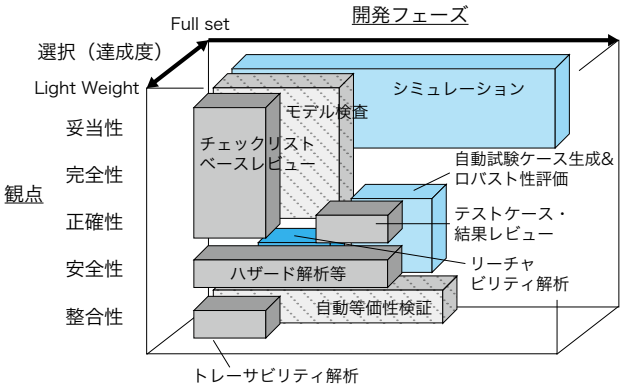

図２　IV&V 手法の組合せ・選択

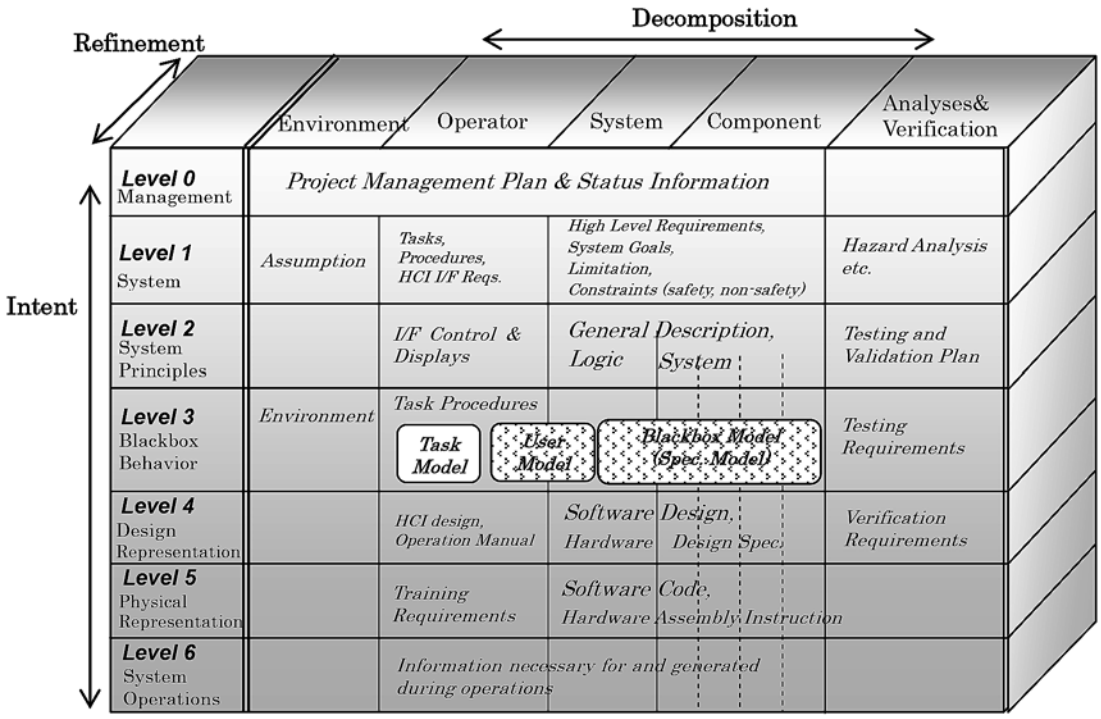

図３　Intent　Specification の構造

複数の手法を組合せ・選択し使用している。それぞれの技術・手法のメリット、デメリットを意識した適用が課題抽出を最大化するためには重要なポイントとなる。

モデル検査技術は、適切なモデル作成および検査目的を設定することにより、モデルやツールにより、複雑な事象を検査し、思いつきにくい想定シナリオを抽出、検証できる。

しかしながら、モデル検査技術のデメリットとして、比較的小規模な宇宙分野の搭載ソフトウェアであっても、ソフトウェア全体に対し、詳細な動作まで機能をモデル化し検証することは困難である。この解決のためには、モデル検査では困難な大局的な確認を行う必要がある。過去の事故・不具合や開発時の留意事項に基づいた経験則から作成されたチェックリストを用いて、人がレビューを行い、検査を比較的網羅的に実施している。このチェックリストベースレビューのデメリットは、検査対象の挙動が人により可読で、検討できる程度の単純な内容であることが点検内容の限界となることである。

モデル検査技術としてここで紹介するものは、適用実績のある手法のうち、以下の検査技術を代表事例として導入した時系列順に紹介する。

（1）SpecTRM（Specification Tools and Requirements Methodology）

（2）SPIN

また、チェックリストベースレビュー手法として、チェックリストの概要とレビュー方法について紹介する。

4 取組みの実際、および実施上の問題、対策・工夫

4.1. モデル検査技術

モデル検査技術は、システムのある性質に対しての「振る舞い」「状態遷移」等をモデル化することにより、ツール上で仕様内容をモデル検証することができる。すなわち、適切なモデル作成および検査目的を設定することにより、複雑な事象を検査し、思いつきにくい想定シナリオを抽出できる。

4.1.1. SpecTRM (Specification Tools and Requirements Methodology)

SpecTRM は、米 MIT（マサチューセッツ工科大学）Nancy G. Leveson が開発した方法論である。また、Safeware Engineering 社によりシステム／ソフトウェア安全設計支援ツールとして市販されている。SepcTRM は、形式的仕様記述 SpecTRM-RL（Requirement Language）により構成され、図３に示す Intent Specification で体系化された安全設計をガイドする統合化手法である。

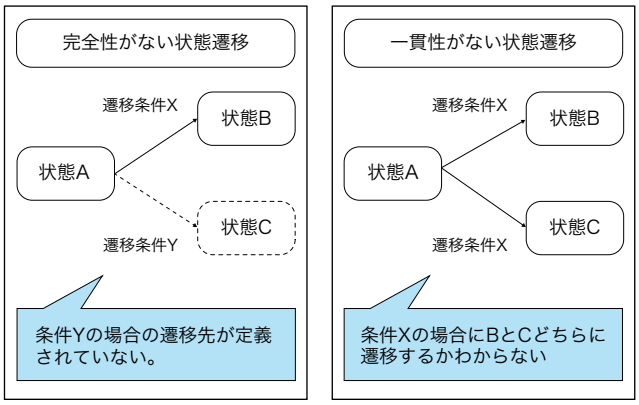

図４　SpecTRM のモデル解析機能

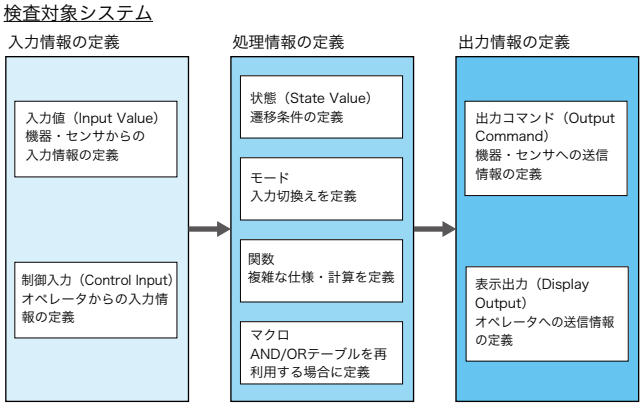

図５　SpecTRM-RL の６要素の関係

これらの手法は、過去の不具合事例の解析結果から必要となる安全上の考慮事項を網羅できるように構築されている。SpecTRM を利用することにより、Intent Specification に沿った対象システム／ソフトウェアのモデリング、モデル検査、シミュレーションを行なうことが可能となり、ガイドに従って安全設計を進めることができる。

モデリングおよびモデル検査を行うことにより、通常の自然言語による要求仕様のレビューでは判断することが難しい、仕様の一貫性（状態遷移先が複数発生することはないか）や、完全性（予想されない状態に陥ることはないか）の問題を抽出することができる。このモデリングおよびモデル検査は図２の Intent Specification では Level3 活動と対応付けられる。SpecTRM の特徴として、検査対象として選定した限定的な範囲で、システムが問題のある状態へ陥らないことを網羅的に自動検査し、上流工程からシステムの完全性・一貫性をモデル検査で検証しながら設計を進めることができる。

- 要求段階の抽象的な仕様から検査可能で、上流工程で問題を見つけることが可能
- ツールにより自動的に網羅的な検証が可能（他の方法で必要な検査式の設定がいらない）
- SpecTRM ツールは、Intent Specification Level3 にモデル検査として、以下の２つの自動分析用の「モデル解析機能」を持っている。（図４参照）
- 完全性分析 Completeness Analysis：状態遷移に関する遷移条件の抜けがないこと
- 一貫性分析 Deterministic Analysis：遷移先が一意でない遷移条件がないこと

SpecTRM モデルは、モデル記述言語 SpecTRM-RL で作成することになる。モデル作成は、システム開発におけるプログラミングに相応する作業と言え、モデル記述言語の文法等を学習する必要がある。検査モデルには、まずシステムの振る舞い（状態遷移仕様）を定義する必要があり、また状態遷移を表現するために、環境変化の表現、入力情報・出力情報等を文法に沿って定義する。

―SpecTRM-RL の６つのモデル要素―

（1）　入力（Inputs；機器・センサからの入力とオペレータ（主にディスプレイ）からの入力を区別して定義可能）
（2）　出力（Outputs；機器・センサへの出力とオペレータ（主にディスプレイ）への出力を区別して定義可能）
（3）　状態（States）
（4）　モード（Modes）
（5）　関数（Functions）
（6）　マクロ（Macros）

この６種類の各モデル要素の関係を図５に示す。対象システムの入出力情報や処理情報を識別し、各モデル要素でシステム仕様を表現し SpecTRM に登録していく。ただしモデル要素は全てを使う必要はなく、検証内容に応じてモデルを取捨選択することが可能である。

導入効果：

必要なシステムや人の動作をモデル化することにより、その完全性や一貫性を自動的に検査でき問題点を抽出、検証できる。完全性解析結果は、遷移先定義が無いことを意味するが、定義が不要なパターンも出力される可能性があるため、出力内容を分析し、各パターンの意味付けを行う必要がある。一方、一貫性解析結果は、遷移条件が一意に決まらない状態遷移があることを意味

するが、遷移不要なパターンも出力される可能性があるため、出力内容を分析し、各パターンの意味付けを行う必要がある。この結果、意味のあるとされたものは、「想定外・障害発生時のケース」に相当する。すなわち、この状態になった場合にシステムがどうすべきか決まっていないことを示しており、安全上の問題をもつものであるため、対応策を検討する必要がある。ここでは紹介しなかったが、安全性、一貫性の自動検査機能の他、入力値を与えて挙動の確認を行うシミュレーション機能もある。

SpecTRM を用いた検査では、システムの想定外・障害発生時のケース抜けを通常 1 システム辺り数十点の問題を抽出できる。また、対象システムの挙動と利用者の運用手順との不成立箇所の抽出にも利用され、手順・マニュアル開発に反映できる。

特徴：

- 様々な仕様を分析する際、モデルは比較的簡単にコピーができるため、コピーにより仕様を変えて自由度の高い分析を実施することができる
- 状態遷移や遷移条件等の情報が統一フォーマットで集約されるので、仕様の確認をする場合に設計書を見なくても必要な情報が得られやすい
- ツールがモデルの管理機能およびインポート・エクスポートを持っているため、過去に使用したモデルを蓄積し再利用する仕組みを用意すると、検査作業の効率化が図れる

課題：

以下のケースには向いていない。

- 複数の状態遷移が時間の経過とともに連携しながら発生するケース

（SpecTRM は状態遷移変化を伝搬し探索することが困難なため）

- 状態遷移の実行条件がシンプルなケース

（モデリング工数がかかる割には、複雑な条件に対する検査結果とならない）

4.1.2. SPIN

SPIN は、現 JPL Gerard J.Holzmann らが開発したモデル検査ツールである。この手法は、検査モデル作成に必要な情報（状態、実行条件、実行内容、外部イベント、制約条件等）が明確になっている工程が対象となる。特に、対象ソフトウェアが仕様どおりに稼働しない状態の検出、対象ソフトウェアの実行条件・状態遷移等に矛盾および抜けがないことの検証に向いている。SpecTRM に比較して、より仕様が具体的に定義された段階に導入しやすい。

この SPIN による検査には以下の入力情報を用いて作業を行う。すなわち、これらの情報が明確に定義できる段階から適用することが適切である。

■ システム機能仕様情報

- システム機能の処理内容
- システム機能の起動条件
- システム機能の処理フロー等

■ 検証条件情報

- システム異常状態の定義情報（エラー出力する状態）
- システム正常終了の情報等（正常終了でない状態を異常状態として使用する）

検査結果として、以下の結果を得ることができる。

■ 検証条件の成立可否

■ 反例情報（初期状態から検証条件を満たす状態までの経緯）

■ 状態遷移の環境情報（検証条件を満たした状態の、状態管理変数の値等）

検査結果として得られた反例情報を一つずつ実際に問題であるかどうかを評価しながら判定していくことになる。

導入効果：

比較的仕様が明確になってきた場合に導入すると、SpecTRM では検査が困難な、複数の状態遷移が時間の経過とともに連携しながら発生する問題を抽出することができる。

宇宙システムに適用を行い、設計の初期段階の仕様に対して危険な状態になるシナリオ（反例情報）を識別することができた。このシナリオは、安全上問題に繋がる可能性の高いものであり、対策検討につながった。

課題：

効果を上げるためには、反例情報の分析および検証条件の設定をある程度定型化しながら行う等の工夫が必要である。

4.1.3. モデル検査技術の導入上の課題と解決策

モデル検査技術の JAXA における導入事例から以下の経験則がある。

表2　ボイジャー・ガリレオチェックリスト

Interfaces
(1) Is the software's response to out-of-range values specified for every input?
(2) Is the software's response to not receiving an expected input specified? (That is, are timeouts provided?) Does the software specify the length of the timeout, when to start counting the timeout, and the latency of the timeout (the point past which the receipt of new inputs cannot change the output result, even if they arrive before the actual output)?
(3) If input arrives when it shouldn't, is a response specified?
(4) On a given input, will the software always follow the same path through the code (that is, is the software's behavior deterministic)?
(5) Is each input bounded in time? That is, does the specification include the earliest time at which the input will be accepted and the latest time at which the data will be considered valid (to avoid making control decisions based on obsolete data)?
(6) Is a minimum and maximum arrival rate specified for each input (for example, a capacity limit on interrupts signaling an input)? For each communication path? Are checks performed in the software to avoid signal saturation?
(7) If interrupts are masked or disabled, can events be lost?
(8) Can any output be produced faster than it can be used (absorbed) by the interfacing module? Is overloaded behavior specified?
(9) Is all data output to the buses form the sensors used by the software? If not, it is likely that some required function has been omitted from the specification.
(10) Can input that is received before startup, while offline, or after shutdown influence the software's startup behavior? For example, are the values of any counters, timers, or signals retained in software or hardware during shutdown? If so, is the earliest or most-recent value retained?

Robustness
(11) In cases where performance degradation is the chosen error response, is the degradation predictable (for example, lower accuracy, longer response time)?
(12) Are there sufficient delays incorporated into the error-recovery responses, e.g., to avoid returning to the normal state too quickly?
(13) Are feedback loops (including echoes) specified, where appropriate, to compare the actual effects of outputs on the system with the predicted effects?
(14) Are all modes and modules of the specified software reachable (used in some path through the code)? If not, the specification may include superfluous items.
(15) If a hazards analysis has been done, does every path from a hazardous state (a failure-mode) lead to a low-risk state?
(16) Are the inputs identified which, if not received (for example, due to sensor failure), can lead to a hazardous state or can prevent recovery (single-point failures)?

・強力なモデル検査の他、モデル検査の前のモデル作成段階で多くの仕様上の問題を抽出できる

また、モデル検査技術の導入上の課題と JAXA における解決策は以下のとおりである。

・モデル検査に依存し過ぎると大観的な重要な問題を見逃すことがあるが、チェックリストベースレビューなど他の手法との組合せによってこれを防ぐことができる。
・システムおよび環境をすべてモデル化することが困難なため、誤った絞り込みや抽象化を行うことにより、重大な問題を見逃すことがある。作業実施前に確認したいシナリオを分析し、関係する範囲を事前に仕様設定者と確認を行うことで回避できる。
・モデル化の段階および検査結果から正しく問題点を抽出するためには、対象システムのドメイン知識がないと誤った問題の抽出となるため、ドメイン知識を習得してから、またはドメイン知識を有する者と作業を行うことが望ましい。
・モデル検査はモデル化した範囲である程度網羅的に特徴抽出を行うことができる反面、連続系の制御則の様なシステムを検査することが困難である。特定の検査したい対象シナリオが明確になっている場合は、モデルを動作（シミュレーション）させてランダム入力に対する検証を行うことが有効である。

4.2. チェックリストベースレビュー手法

4.2.1. ボイジャー・ガリレオチェックリスト

JAXA が現在利用しているチェックリストの一つとして、1996 年に JPL（NASA ジェット推進研究所）Lutz が発表した Safety Checklist※1 がある。通称、ボイジャー・ガリレオチェックリスト（表 2）である。

Lutz は、1991 年に Jaffe, Leveson らが過去の事故データの分析結果に基づき提唱したセーフティクリティカルのクライテリアを参考にチェックリストを作成した。また同時に NASA の探査機ボイジャー（Voyager）およ

【脚注】
※ 1　Robyn R. Lutz "Targeting Safety-Related Errors During Software Requirements Analysis," The Journal of Systems and Software, Vol. 34, Sept, 1996, pp. 223-230.

びガリレオ（Galileo）で実際に発生した 192 件の不具合を詳細に分析し、チェックリストを検証した。このチェックリストを用いることで、192 件の不具合のうち 77% である 149 件は、要求分析段階で発見できたことが分かった。

4.2.2. JAXA 独自チェックリスト

4.2.1 項で紹介したものは、JPL の不具合分析の結果を活用したボイジャー・ガリレオチェックリストの導入例であるが、日本の宇宙業界でも同種の不具合が発生しており、これらの不具合情報や開発・検証の経験者の知見を体系化した JAXA 独自のチェックリストを整備、利用している。表３に姿勢制御系チェックリストの例を示す。

4.2.3. チェックリストレビューの導入効果

ボイジャー・ガリレオチェックリストや姿勢制御系チェックリスト等を利用してレビューを行うことにより、通常のレビューよりも過去の不具合事例・経験知に基づく確認ポイントを確実にチェックでき、問題ない場合も含み点検結果の記録を残すことができる。ボイジャー・ガリレオチェックリストでは、宇宙システムの適用を通じて、要求仕様上の問題点を通常 1 システムにつき数点の問題点が要求定義段階で抽出できる。タイムアウト時間の設定がなく、待ち続けてしまうケースや、信頼性の観点でバックアップ計算機に切替える場合にシステムの状態によっては、切替信号を送信されずにシステムが停止するケースが発見され、重大な事故の原因を要求段階で未然に抽出・解決できることが実証できた。また、姿勢制御系チェックリスト等の場合、単位系等の初歩的な間違えから想定外の故障時の処理機能の抜け等複雑な問題まで幅広いレベルの要求仕様上の問題点を抽出でき通常 1 システムにつき数十点の問題点が要求定義段階で抽出できる。２章で示した MCO の例の単位系の誤りもこのチェックリストを利用してチェックしていれば要求定義段階で発見できたはずである。JAXA では同様なチェックリストを５つのシステム分類に対し、準備しており、重大な事故の原因を要求段階で未然に抽出・解決できることが実証できた。

4.2.4. チェックリストベースレビューの導入上の課題と解決策

チェックリストを用いたレビュー活動は、過去の不具合事例・経験知に基づく確認活動が可能となるが、課題としては、問題点を正しく抽出できるか否かが、レビュー活動を行う作業担当者のスキルに依存してしまうことである。また、限られた時間の中ですべての確認項目を同様の深さで確認することは期待する結果が得られないことになる。

このチェックリストを用いたレビューの質・効率を作業担当者のスキルに極力依存しないように、JAXA では、それぞれのチェック項目に、詳細なチェック方法を補足説明している。以下に人工衛星姿勢制御系サブシステムのチェックリストをサンプルに解説する（図６）。

また、複数の作業担当者で共同して実施する場合は、

表３　姿勢制御系チェックリスト　（抜粋）

項目	内容	詳細	事例	対象文書
入出力データ	物理量データ毎に単位が明確に記述されているか？	搭載ソフトウェアの中では、角度は、ラジアン単位、度単位が混在して用いられているので、単位の扱いについては十分な注意が必要である。	設計基準に従って入出力データインタフェースは統一されているが、既開発品の利用により、機器毎に異なる場合がある。	データ仕様
ON/OFF 手順	ON/OFF 手順、および ON 時データ処理手順は明記されているか？	使用するセンサによっては、電源 ON 後、センサ信号を用いるまでにマスキング等の初期設定操作を必要とするものがある。	マスキングをしていない場合、誤った動作をしたことがあった。	要求仕様書
異常信号処理	異常信号（雑音）に対して処置策が施されているか？	パルス信号に雑音が重畳し、その結果、物理的に可能な回転速度範囲を逸脱していないかをソフトウェア側で絶えずチェックする必要があり、要求仕様に規定されているか？		要求仕様書
フォールトトレラント設計	モード別の FDIR の設計が妥当か？	故障耐性の設計として、モードごとの故障解析（FMEA、FTA）の有無、クリティカル故障の抽出、それに対する設計上の対応策（FDIR 機能）がとられているか？冗長構成をとるのが難しい、あるいは不可能な部分を除いて単一故障点が回避された設計となっているか？	故障を特定できるか？センサを使わないモードで FDIR が機能していないか？	要求仕様書

チェック項目、詳細確認ポイントをスキルに合わせて配分し、完了後に作業担当者間で確認を行うことで作業の質・効率を向上できる。

優先度：　高、中、低

　重要な機能に関わるチェック項目、あるいは問題が多く見られる項目の優先度

　難易度：

　易……ひとつの項目のチェックを、一人の担当者が15 分以内で行える

　中……30 分以内

　難…… 1 時間以上

　姿勢制御系チェックリストのチェック項目の構成と所要時間を表 4 に示す。

　実際の利用時には、対象となる宇宙システムの特徴・リスクおよび配分できる作業予定時間から、それぞれの装置に対して、優先度のレベルを選択し、点検項目を決定する。あらかじめ定義された難易度に合わせて、ボイジャー・ガリレオチェックリストと同様に作業担当者のスキルと照らしながら、作業分担を行っている。

5 達成度の評価、取組みの結果

　今回、適用した技術・手法である「モデル検査」および「チェックリストベースレビュー」の個別の導入の効果は、 4 章で述べたとおりである。ここでは、当初目的として設定した上流工程における仕様定義段階の 3 つの課題の改善状況について表 5 にまとめる。

例えば、チェック項目（2）

Is the software's response to not receiving an expected input specified? (That is, are timeouts provided?) Does the software specify the length of the timeout, when to start counting the timeout, and the latency of the timeout (the point past which the receipt of new inputs cannot change the output result, even if they arrive before the actual output)?

詳細確認ポイント：

- 安全上重要なソフトウェアに対し期待される入力値にタイムアウト時間が設定されているか？
- タイムアウト時間が設定されていない場合、対象のソフトウェアの挙動が安全であり、期待どおりに動作するか？
- 必要なハードウエア、インタフェースは存在するか？また、他の事象によりインタフェースが利用できないことはないか？
- タイムアウト時間はインプット、アウトプット側で定義・了解されているか？

・・・など。

図 6　人工衛星姿勢制御系サブシステムの補足説明例

表 4　チェック項目の内訳

優先度	高			中			低			計
難易度	易	中	難	易	中	難	易	中	難	
センサ A	6		1	5			1			13
センサ B	3	1	1	4			1			10
センサ C				4	2					6
センサ D	2						4		1	7
センサ E				1	1	1				3
アクチュエータ A	6	1					1	1	1	10
アクチュエータ B	3			4			3	1		11
全体システム	1	1	2			1	4	2	3	14
計	21	3	4	18	3	2	14	4	5	74
作業時間 (h)	5.25	1.5	4.0	4.5	1.5	2.0	3.5	2.0	5.0	29.25

表 5　上流工程における仕様定義の改善状況

上流工程の仕様の課題	改善状況
正確性の欠如	仕様中の単位誤記・正負／極性反転等の初歩的であるが重大な事故を引き起こす過誤等、記述の曖昧さに起因する複雑な挙動のチェックリストベースレビューにて多くの問題を早期に抽出できる。また、仕様書間の一貫性をモデル検査技術により確認することで、人がチェックしにくい複雑な利用シナリオ上の過誤を検出可能となる。
不完全性の残留	仕様書の記載が不足・欠落したことによるアルゴリズムや機能の欠落・欠陥については、完全性をモデル検査技術により抽出可能となる。しかしながら、本来のシステムに期待される正しい動きを理解・把握する人とともに、意図する挙動かどうか最終確認することが、より精度を高めることにつながる。また、仕様記述の際に欠落しやすい情報についてはチェックリストなどに含め、レビューにて点検することが更に効果を上げることになる。
使用方法・環境の想定不足	システムが動作する環境が全て網羅され仕様が設定されていることは少ないのが一般的であるが、特に安全確保のためには、使用方法・環境の想定を超えた場合でも安全な動作が期待される。環境や使用方法も含めてモデル検査を行うことで、想定しない危険シナリオ等をある値度抽出することができるが、今後の改良が必要である。