博　士　論　文

宇宙システム開発効率化のための
知識の分子化による知識再利用手法に関する研究

東京大学大学院工学系研究科
航空宇宙工学専攻

髙橋　　亮平

概要

近年、民間の宇宙ビジネス拡大とともに、ロケット、人工衛星をはじめとする宇宙システム開発ミッション数は増加している。一方で、宇宙システム開発プロジェクトにおけるコスト超過、スケジュール遅延の問題、打ち上げ後の不具合によるミッション失敗といった問題は解決しない。その原因の一つとして、宇宙システムにおいて、どういう設計がふさわしいか、設計時に何を気を付けるべきか、試験で何を確認すればよいかといった、プロジェクトを支えるエンジニアの技術的な知識の不足があげられる。

解決方法として、過去のシステムの積極的な再利用が挙げられる。過去のシステムの再利用は、検討すべき設計事項、実施すべき検証事項を減らし、プロジェクトをシンプルにする。一方で、全く異なる目的や環境条件で、実施するミッションでは、システムの再利用を簡単に行うことはできない。それは、異なる目的、環境条件などのミッションコンテキストでは、単純に性能を発揮できなかったり、新たなコンテキストと創発し、不具合をはじめとする悪影響が発生するリスクがあるからである。別の方法として、システムを要素技術レベルまで分解し、知識としてまとめ、データベースにそれらを集約し、ユーザーが適宜それに検索をかけ参照することで、再利用することが挙げられる。しかし、この方法では、ユーザーが必要だと認識できる知識は検索し再利用される一方で、ユーザーが必要だと認識していない知識は見落とされ再利用されない可能性がある。これは、重大な不具合リスクの見逃しにつながり、コストやスケジュールの超過、軌道上不具合につながる危険がある。また、知識を誤ったコンテキストで再利用してしまい、思わぬ不具合につながるリスクもある。多くのプロジェクトでは、この課題を、経験豊富な専門家による技術レビューによって解決している。専門家は、単なる経験に加え、深い物理的な知識を持っており、必要な知識を把握し、どのようなコンテキストで知識を再利用できるかを判断することができる。一方で、専門家を外部から発見し、招集すること、および専門家に設計情報やミッションのコンテキストを伝達し、認識を共通化するのに莫大なリソースが必要となる。

この知識再利用の課題を解決するため、本研究では、異なるプロジェクト間で、エンジニアが必要な知識に関する認識がなくても、役立つ適切な知識を効率的に、かつ専門家のレビューのように効果的に再利用するフレームワーク「知識分子」を提案する。ここで「知識」とは、

特に宇宙システム開発プロジェクトに関わる不具合に関連する知識を指し、例えば、設計に対する不具合リスク、不具合の改修・防止方法や検証方法、それに関連するコンポーネントの機能や性能などである。また、知識分子は、知識が結合する条件を定量的な条件とグラフ構造で表すことができ、それらに基づき知識自身が自律的に、システムにおいて適合できる個所を特定できる手法である。また、このフレームワークを効率的に活用するために、以下も付随して提案を行った。

1. 知識分子の結合条件、宇宙システムのモデルを表現するためのオントロジー
2. オントロジーを活用し、知識分子と宇宙システムのモデルを結合するためのアルゴリズム
3. 結合した知識のうち重要な知識を識別するための知識評価手法
4. 知識分子を活用するツールの実装とプロセス

本研究では、この提案手法の検証を以下の 4 つの実験を行い、その実用性の検証を行った。

1. 簡易的な模擬衛星に対する知識再利用
2. 実衛星 EQUULEUS の知識を、開発中の ISSL6U に再利用するケーススタディ
3. 提案手法と、ルールベースのみを使った手法の比較実験
4. 実際にツールを利用してもらい、有効性を確認するユーザーインタビュー

なお、知識分子は不具合以外にも様々な知識 (要求に対して、満たすべき技術候補の知識、技術に関するコストの知識等）を取り扱えると考えている。一方で、網羅的に検証を行うことは難しいため、本研究では「不具合」の知識に絞り、より実運用等の例を想定し説明や検証を行う。最後に、提案手法の特徴を整理しつつ、残った課題の整理、およびそれらを解決する方法として考えられる方法をまとめた。

目次

図目次

表目次

第 1 章

研究背景

1.1　宇宙開発における課題と知識の関係

人工衛星や有人宇宙船、ローバーなど、宇宙システムを開発するプロジェクトにおいて、これまで多くの、コスト超過、スケジュール遅延、そして軌道上不具合の発生などの問題が発生してきた。例えば、NASA の新型超大型ロケットである Space Launch System (SLS) は、5 年以上のスケジュール遅延とそれに伴う人件費の増加を経験している [1]。 また、三菱重工業によって開発が計画された日本発の旅客機 Mitsubishi Space Jet は、旅客機の型式証明に関する要求が、設計時に把握されていなかったことから、設計の手戻り、開発の遅延が発生し、長いスケジュール遅延の末、プロジェクトの凍結にまで至った [2]。 サイズと重量を絞ることで、低コスト短期開発可能な CubeSat は、多くの大学研究室のプロジェクトや研究機関で活用されつつあるが、軌道上で全く起動や通信ができない結果に終わることが多い [3]。 このような問題は、過去数十年において、減るどころか逆に増加しているといわれている。例えば、アメリカ防衛システム開発において、1970 年代のスケジュール遅延率が 33% であったのに対して、現在は 63% まで上昇している [4]。 このような宇宙システム開発プロジェクトにおけるコスト超過、スケジュール遅延、軌道上不具合の原因として、技術的な不確定性、すなわち開発者の知識不足があげられる。De Weck らは、システム開発プロジェクトにかかわる不確定性をレイヤー化し、その中心の不確定性の一つとして技術的な不確定性を上げている [5]。NASA の調査によると、スケジュール・コスト遅延を発生する確率は、システムの複雑度に強く相関している [6] が、システムの複雑度は、システムを構成するサブシステム同士の依存関係だけではなく、それらを構成するサブシステム、およびサブシステムのかみ合わせの技術成熟度に依存する。これらの技術成熟度は Technological Readiness Level (TRL) と呼ばれる指標で管理され、TRL は開発する機関の知識レベルや開発経験に依存する。そのため、知識が不足していると、技術成熟度は下がり、スケジュール遅延やコスト超過を引き起こすシステムの複雑度は上昇する [7]。 実際の例として、NASA が Faster, Cheaper, Better のスローガンを掲げた 19070

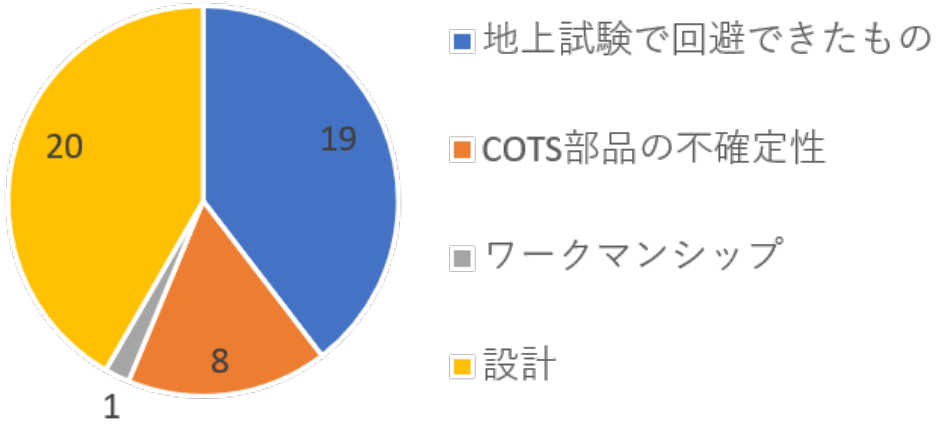


図 1.1　超小型衛星衛星の軌道上不具合の原因内訳 [9]

年代、火星周回衛星をはじめ多くの衛星で不具合が発生した。その原因の一つとしてプロジェクト間で知識が確実に共有する仕組みがなかったことが指摘されている。アポロ時代のような大規模プロジェクトを少数やる方針から、小中規模のプロジェクトをサイクル早く繰り返すようになったが、サイクルを早めるだけで学びを残す仕組みがないため、サイクルを速めても品質は向上することはなかったのである [8]。 また、超小型衛星においても、よりレッスンズラーンドを蓄積し、再利用できている機関ほどミッション成功率が高いという調査結果もある [9]。実際、超小型衛星の軌道上不具合の多くが、地上試験で回避できたもの、あるいは設計に起因するものであり、適切な技術知識があればこれらの不具合は回避できる (図 1.1 [9])。そのため、エンジニアが技術的な知識を十分に持つことはプロジェクトの成功のために非常に重要であるといえる。

1.2　システム再利用の有効性と問題点

前述の技術知識の不足を補う手段として、過去に構築したシステムをそのまま再利用する手法があげられる。過去に構築したシステムをそのまま再利用することで、そのシステムに起こりうる不具合などは、殆どすでに対策されているため、細かいリスクなどをエンジニアが見逃したとしても高い信頼性をもってミッションを実現できる。また、多少のサブシステム、コンポーネントを変更した場合でも、変更点とその影響範囲だけを管理することで、リスクを減らすことができるため、システムの再利用は行うことが望ましい。その場合、後述の Model-Based Systems Engineering (MBSE) のアプローチを利用するうことで、網羅的な変更点の波及効果調査を行うことができ、変更点に伴うリスクを防ぐことが可能である。[10]

しかしながら、ミッションの目的や使用環境が全く異なる場合、システムをそのまま再利用することはできない。そのような場合、システムを構成する技術を要素ごとにまで分解し、

ミッションの目的に応じて再構築するようなプロセス、すなわち知識の再利用プロセスが必要となる。

1.3　宇宙開発における従来の知識共有手段

開発を行う組織の技術知識の再利用する方法は現在複数存在する。具体的には下記の手法があげられる。

1. データベース
2. パブリックな Web サイトや掲示板
3. ドキュメント・モデルの蓄積、公開
4. 教育活動
5. オープンソース
6. 技術レビュー

1.3.1　データベース

NASA では、科学的データの解析方法を、組織内で共有できるように、目的や必要ツールへのリンクと共にデータベースにそれらの知識を保存している。近年では、単純なリレーショナルデータベースだけでなく、高度な関連性検索が容易なグラフベースのデータベースを使用している事例がある [11]。

1.3.2　掲示板、ポータルサイト

パブリックな Web サイトに、役立つ知識情報を載せ、知識を共有する手法がある。NASA の Space Reliabilty ほげ　の機関では、Space Systems Reliability Institute と呼ばれる web サイトを構築し、そこに Cubesat 開発にかかわる知識を公開することで、全世界の Cubesat 開発を共有可能にした [12]。

1.3.3　ドキュメント・モデルの蓄積公開

JAXA や NASA などでは、設計開発プロセスにおいて、宇宙システムの設計情報や、それに至るまでの要求分析、技術解析をドキュメントとして保存することを、システムズエンジニアリングプロセス、あるいは品質保証プロセスとして行っている [13]。 これらの文章は、次プロジェクトで同様のミッションが行われた際に、参照することで、有用な共有の手段となる。近年では、Model Based Systems Engineering (MBSE) と呼ばれる手法に基づき、設計情報を

グラフィカルなモデルですべて記述しすることで、認識の共有ミスを防止したり、あるいは機械解釈による再利用性の向上、トレーサビリティの獲得を実現している [14]。また、JAXA や NASA は、機関での宇宙機開発経験をもとに、各種作業標準を作成し、それを公開し、知識の共有を行っている [15]。

1.3.4　教育活動

University Space Engineering Consortium (UNISEC) と呼ばれる宇宙教育機関では、Cansat や HEPTA-sat と呼ばれる模擬衛星を用いて体験型授業を行うことで、学生に宇宙開発に関わる知識、さらには深いノウハウを広めている [16]。

1.3.5　オープンソースソフトウエア

オープンソースソフトウエア (OSS) は、宇宙開発に関わるソフトウェアを公開することで、多くの人がそのソフトウエアの構成などから知識を得るだけでなく、その開発を通して深い知識を得ることを可能にしている。宇宙開発における OSS の例として、NASA JPL が開発した Fprime と呼ばれる衛星ソフトウエアのアーキテクチャがある。機器同士の依存関係、IF をモデル化することで、通信に必要なソフトウエアを自動生成することができるなど、ソフトウエアコーディングに必要な労力の削減やコミュニケーションミスを縮小することが可能である [17]。

1.3.6　技術レビュー

技術レビューは、宇宙機のような複雑なシステム開発において重要なマイルストーンであり、設計から Engineering Model 開発プロセスへ移行するための Preliminary Design Review, Flight Model 開発へ移行する Critical Design Review などの、フェイズ意向を判定するゲート審査の役割果たす。レビューでは、経験豊富な多様な専門分野を持つエンジニアにより行われ、設計書や開発結果をもとに、検証計画が十分であるか、リスクは十分に識別、対処されているかを確認する [13]。レビュープロセスは、レビューワーと対話を行うことで、その過程でエンジニアが示す設計情報などに不足があれば、それを明らかにすることが可能である。[18] とともに、議論をすることで、過去の経験を複数組み合わせて、レビュー対象のシステムにあった新しい不具合リスクなどを発見することが可能性もある [19]

1.4　宇宙システム開発における従来知識共有手段の課題

従来手法により、プロジェクトで出てきた知識を拡散する、あるいはプロジェクトのために知識を集約することがある程度可能になっている。ここで、あるプロジェクトを成功に導くには、過去の知識を蓄積しシステムに合わせて適切に再利用することが重要になる。知識を蓄積するは、前述の手法のうち、「データベース」「Web ベース」「ドキュメント」などが主な方法となる。一方で、これらの手法は、知識を再利用する際、「プロジェクトに必要な知識が何か、をエンジニアが認知」し、その結果「検索する」というプロセスが必要になる。そのため、システムの構成から必要な知識が認知できない場合、蓄積した知識が再利用されないリスクがある。このような認知から外れてしまう知識が、不具合となり、スケジュール遅延やコスト超過の問題を引き起こす。すなわち、プロジェクトで宇宙システムを開発するエンジニアが、気づかない知識、例えば不具合のリスクや解決すべき課題が存在し、開発途中での不具合やそれに伴う手戻りを発生させることになる。

また、「教育活動」「オープンソースソフトウエア」は、ユーザーが知識を認知していなくても、その活動を通して、新たな知識を身に着けることが可能であるが、そこで身に着けた知識は必ず、エンジニアが関わるプロジェクトで利用できるとは限らない。

「技術レビュー」は、知識に対するエンジニアの認知をせずとも、プロジェクトに適した知識を再利用できる手段である。エンジニアが認知していない知識であっても、経験豊富なレビューワーが必要な知識を認知することで、不足した知識を提供することが可能である。また、人間の柔軟な発想により、複数の過去の経験を組み合わせてリスクを見つけたり、不足している情報をエンジニアに問い合わせながら知識を提供することができる。一方で、技術レビューは、専門家レビューワーを呼び集めることにコストが必要となる。そのため、頻繁に繰り返しレビューを実施することはできない。また技術レビューは、レビューワーにプロジェクトのコンテキスト、設計の意図や結果をくまなく伝える必要があり、対話に非常に労力を必要とすることが多い。さらに、レビューワーによっては、偏った視点や経験をもつエンジニアもおり、必要以上のリスク懸念や検証の不足が指摘されてしまうこともある。その結果、エンジニアとの議論が発散してしまい、プロジェクトの進行に影響が出てしまう [20]。このように必要なリソースの面で、技術レビューは課題がある。

1.5　不具合に関する知識の種類

不具合に関する知識を再利用しようとした場合、不具合知識が表現される形式を網羅的に扱える必要がある。不具合に関する知識は「定性的なコンポーネント、機能の関係」、「数値をもつパラメータに関連するもの」「不明確な知識」の 3 つの種類があると考えられる。例えば、衛

星内部の導体片が、接地されていないため帯電が発生し、最終的に放電とそれに伴う大電流が発生する、という不具合は、導体片が接地されていない、という定性的な関係で表現される。また、温度が 150 degC 以上の環境下で、モーターが動作しなくなったという知識は、温度に関する数値が含まれたパラメータに関連する知識である。また、原因は不明だが、I2C を使用した伝送回路で通信が途絶する、という不具合に関する知識は、完全な原因が不明確な知識となる。そのような場合でも、人間は、過去のシステムと現在のシステムの類似性から、過去の不具合が起こりうるかもしれないと類推することができる。そのため、効果的に不具合知識を再利用する際は、これらを適切に再利用する必要がある。

1.6　本研究の目的

従来の知識利用手法では下記の課題が存在する。

- データベースなどのユーザーが検索を行う手法は、ユーザー自身が必要な知識とその再利用可能なコンテキストを認知する必要があり、抜け漏れや誤った知識再利用を行ってしまう危険がある。
- 教育活動やオープンソースの活動では、開発のコンテキストにあった知識を再利用することができない。
- レビューなど人を介した知識再利用は、複数の知識の組み合わせや設計情報の補完など深い知識利用が可能である一方で、認識をすり合わせるのに非常にリソースがかかるうえ、属人性が高くなってしまう

これらの課題により、知識を活用した効率的な宇宙システム開発は難しくなっている。そこで本研究では、これらの問題を解決する知識再利用手法を提案し、効率的な宇宙システム開発の実現を目指す。具体的には下記の機能を持つ知識利用手法を提案する。

1. ユーザーが知識を認知しなくても必要な知識を提供できる
2. コンテキストに合わせて様々な知識 (定性的な関係、パラメータ、不明確な知識) を適切に再利用できる
3. レビューのようなコミュニケーションコストを必要としない効率的な知識利用ができる
4. 不足した情報の補完、複数の知識の組み合わせなど効果的な知識利用ができる。
5. 宇宙システムに適用できるために、機械学習のように大量のデータが無くても利用できる。

なお、本研究では宇宙システム開発における「不具合」に関する知識を主な題材に、提案手法の説明、検証を実施する。(設計そのものやシミュレーションに関する知識分子の応用については、Appendix D に示す)

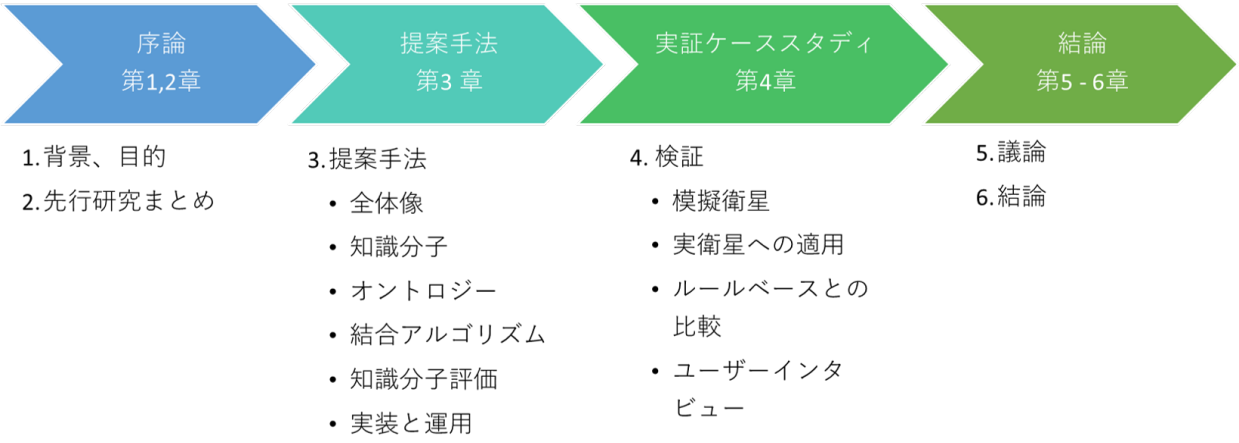


図 1.2　本論文の構成図

1.7　本論文の構成

本論文は大まかに、第 2 章で先行研究をまとめた後、第 3 章から第 8 章まで、提案手法について説明する。第 9 章から第 12 章では、提案手法の有効性を示すために行った一連の実験とその結果を示す。第 13 章、第 14 章では、実験結果についてまとめ、本研究の特徴、課題を述べる (図 1.2)。

詳細には、以下のように構成されている。

- 第 1 章では、背景となる宇宙システムの知識利用の問題点をまとめ、本研究の目的を提示した。
- 第 2 章では、知識の再利用性を高める先行研究についてまとめ、課題の整理を行う。
- 第 3 章では、目的、先行研究の課題を解決する提案手法を示す
- 第 4 章では、本研究の有効性を示すために行った実験について、その内容と結果を説明する。
- 第 5 章では、提案手法とその検証結果をまとめ、提案手法により得られる効果と課題について議論を行う。
- 第 6 章では、総合的な提案手法の結論を述べる。

また、本研究に課される要求と、それに対する実装方法、説明場所を表 1.1 に示す（要求については、第 3 章で述べる)。

なお、Appendix には以下の内容を示す。

表 1.1 提案手法に対する要求と実装方法、説明箇所の対応

要求	提案手法での実現方法	備考	説明 (章)	検証 (章)
必要な知識をユーザーが認識しなくても獲得することができる	知識分子と結合アルゴリズム	ユーザーが知識を認知しなくても、システムに結合する知識が推論される	3.1	4.1
コンテキストを踏まえて様々な知識 (物理関係、パラメータ、不明確) が適切に再利用できる	知識分子の Condition Graph による結合判定	使用される部品、機能、属性とそれらの構造など、定性的な情報をもとに結びつける	3.2, 3.4	4.1
	知識分子の Analysis の結合による結合	パラメータをもとにした解析により知識を結びつける	3.2	4.3
	知識分子結合アルゴリズム (クエリ言語 or 類似度ベース)	類似度によって曖昧な原因の知識を再利用することができる	3.4	4.3
不足した情報の補完、複数の知識を組み合わせが可能である	Analysis によるパラメータの取り込み、解析	ユーザーに対して不足した情報を問い合わせる	3.2	4.3
	知識分子の Operator による複数知識による推論	複数種類の知識を組み合わせ、複雑な知識を再利用可能にする	3.2	4.2, 4.3
効率的に知識を再利用できる	知識分子評価手法	大量に結合する知識から、有用な知識を人間が識別できるようにする	3.5	4.2
	Excel ベースのインターフェース, 不具合知識定義、運用プロセス	知識やモデルを効率的に定義できる	3.7	4.4
大量のデータを集めることが難しい宇宙領域で適用できる	Condition Graph と知識分子結合アルゴリズム	機械学習のような大量の学習データは必要ない	3.1	4.1

- 付録 A では第 4.2 章で利用した EQUULEUS の知識分子一覧を示す
- 付録 B では、データマイニングの手法を用いて知識分子を一般化し、得られた知識を洗練化する手法を提案する。
- 付録 C では、自然言語処理を用いて、知識分子を、自然言語によって書かれた不具合に関するレポートから自動生成する手法を提案し、現状得られている結果を示す。
- 付録 D では、不具合に関する知識再利用以外の応用先について、その事例と期待される効果をまとめる。

第 2 章

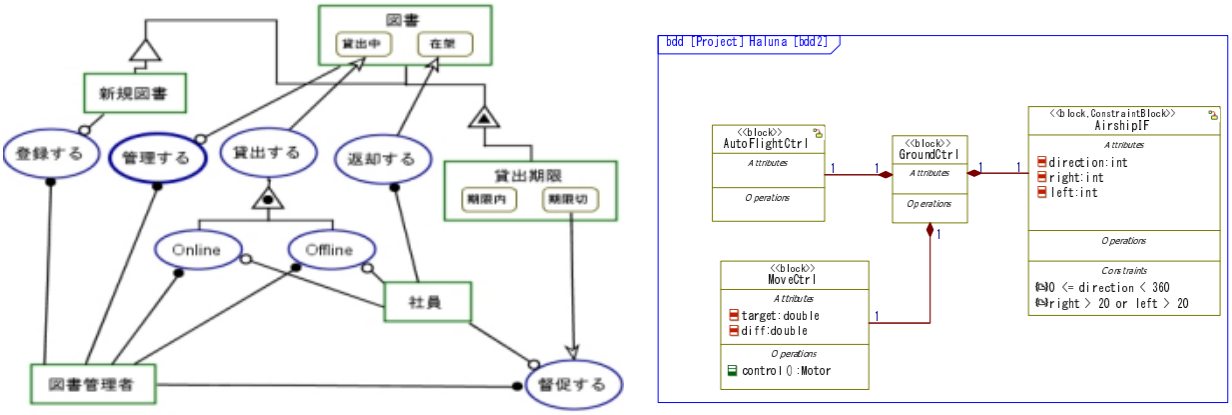
先行研究

本章では、知識を再利用する、もしくは再利用性を高める先行研究について調査した結果をまとめる。

2.1 Model Based Systems Engineering

複雑なシステムの設計開発に必要なプロセスを適切に実行し、決められた予算・スケジュールで品質の高いシステムを開発することを目的とする Systems Engineering と呼ばれるアプローチが、航空宇宙の分野では広く用いられている [13]。 Systems Engineering において、プロジェクトに関わる情報をステークホルダー間で共有、管理する媒体は非常に重要である。ここでのプロジェクトに関わる情報とは、システムの設計情報だけでなく、そのもとになる要求、要求の階層関係、要求に対する仕様や検証項目の対応関係、技術間のインターフェス、技術の検証方法など、システムの構築に関わる全ての情報を指す。従来、この管理媒体は、自然言語といくつかの図、表を用いたドキュメントで管理されてきた。自然言語で記述されたドキュメントは人間の理解が容易であり、記述に必要な知識や能力への要求も少ない一方で、機械解釈が難しく計算機を用いた処理が容易ではない。そのため、自動的な設計整合性の確認や設計変更の波及効果の確認を計算機を用いて行うことはできず、何か設計変更が起きた際は、人が逐一整合性を確認し、マニュアル的に変更を行う必要がある。このようなマニュアル的な整合性確認には、ヒューマンエラーが高い確率で発生してしまう。そのため、ドキュメントベースの管理の場合、開発途中でシステムの技術的な整合性が取れなくなってしまうリスクが大きい。

この問題に対して、Model-Based Systems Engineering と呼ばれる手法が考案、開発されてきた [14]。Model-Based Systems Engineering では、設計情報を自然言語のドキュメントに保存するのではなく、「モデル」と呼ばれる、機械解釈可能で、かつグラフィカルに設計情報を可視化し、人間が理解できる一貫した形式で保存することを提案している。特定の記法に沿って、設計情報を記述することで、複数のステークホルダーで認識の齟齬を起きづらくすることが可



(a)　　(b)

図 2.1　各言語で記述されたシステムのモデル例 (a) OPM [21] (b) SysML(内部ブロック図)[22]

能である。また機械解釈性により、設計変更のトレースを自動的に行うことが可能となる。

下記の図 2.1 は、Object Process Methodology (OPM)、および SysML と呼ばれるモデリング言語を用いてシステムの設計情報を記述した場合の例である [21][22]。Model-Based Systems Engineering において、モデルの記述方法、言語は複数種類が提案されている。それぞれのモデル言語は、システムの描写に必要な性質をそろえているものの、表現が行える観点や学習のしやすさに差異がある。OPM は、オブジェクトとプロセスの関係を階層的に記述することができ、ダイアグラムの種類も単一であるため、学習が容易である。一方で SysML は、OPM よりもさらに多様な表現が可能で、要求の階層化やシーケンスにおけるシステムのふるまいの変化を記述することなどが可能である。一方でダイアグラムの種類は多くなり、学習コストは高い。

ダイアグラムによる直感的な理解のしやすさと、機械解釈が可能なことは、知識再利用の面でも機能を発揮する。過去のプロジェクトで開発されたシステムの設計を再利用する場合、設計情報の記述者と設計再利用者で情報の解釈が異なると、その認識齟齬による不具合リスクが存在することになる。その点、MBSE を用いて記述された情報は、グラフィカルな情報により、認識齟齬のリスクは低減されると考えられる。また、過去のミッションとアーキテクチャがほとんど変わらず、一部分の設計のみを変更するようなミッションの場合、MBSE の機械解釈性を活用することが可能となる。新しいミッションでの変更されたインターフェース、要求をインプットとして、機械的な処理により、変更点の波及効果を網羅的に検索し、必要な追加解析、仕様の変更の必要性を網羅的に検索することが可能である。特に、システムにおいて不具合の温床となるコンポーネントのインターフェース仕様は、このような自動化が威力を発揮する（図 2.2）。これにより、新しく構築するシステムのアーキテクチャが既存アーキテクチャに近ければ、MBSE をもとに、過去の技術を利用し設計プロセスを効率的に行うことが可能で

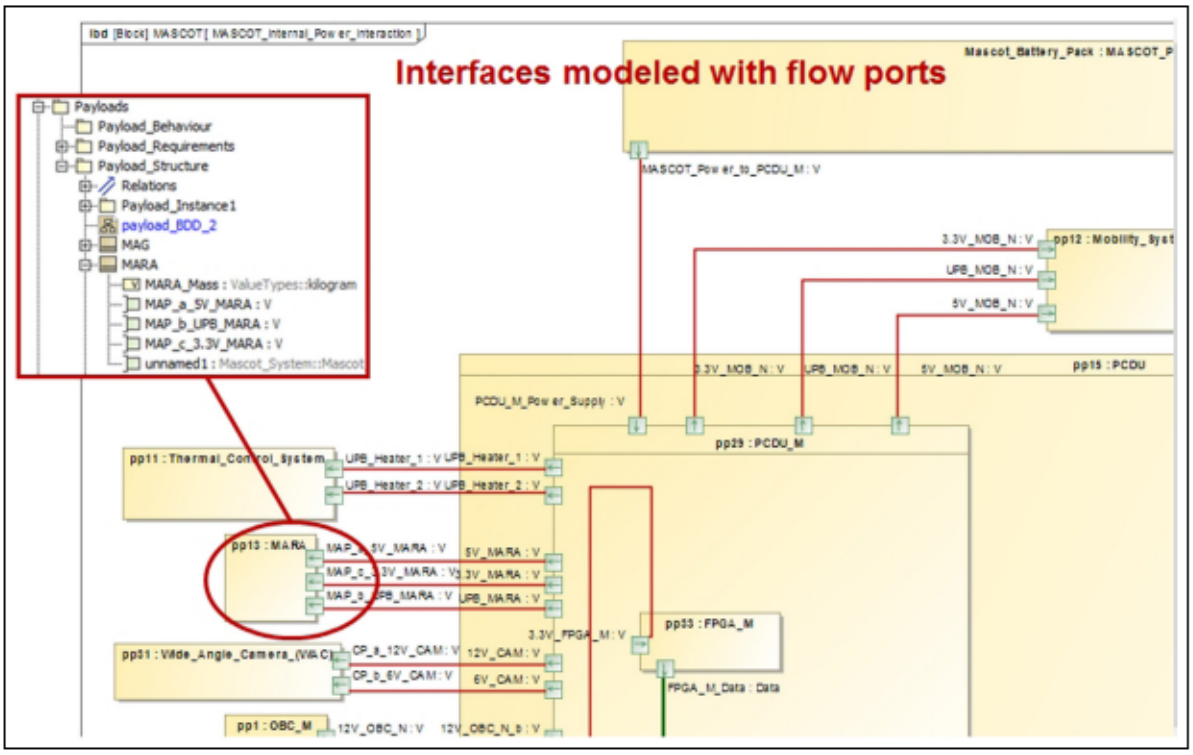


図 2.2　MBSE を用いたインターフェースコントロール [10]

ある。[10]

2.2　MBSE の課題とオントロジーの活用

前述のとおり、MBSE では、モデルを用いた効率的な設計情報の再利用が可能になっている。一方で、モデルに記述する詳細な説明は、自然言語で記述されることになり、結局再利用できる範囲が限られてしまうといった課題が存在する。また、モデルで使用する単語、表現方法のばらつきも、ドキュメントより少ないとは言えどある程度存在してしまっている。そのため、異なる部門やツール間でモデルを再利用できなくなってしまっているという課題がある。この課題に関して、近年では Model-Based Systems Engineering での記述方法をより厳密にオントロジーとして規定する取り組みがなされている。ここでオントロジーとは、概念の抽象、具体、所属などの関係を、機械解釈可能な形で明示的に定義する方法である。JPL では、CAESAR と呼ばれるプロジェクトの中で、よろい厳密なモデルを規定し、複数のツール同士を結合することで効率的な情報の再利用を実現しようとしている [23] (図 2.3)。また、Foundry Furnance と呼ばれる Concurrent Engineering ツールでも同様に、オントロジーを定義、使用することで、ツール間やプロジェクト間でのデータ再利用を効率的にしている [24]。

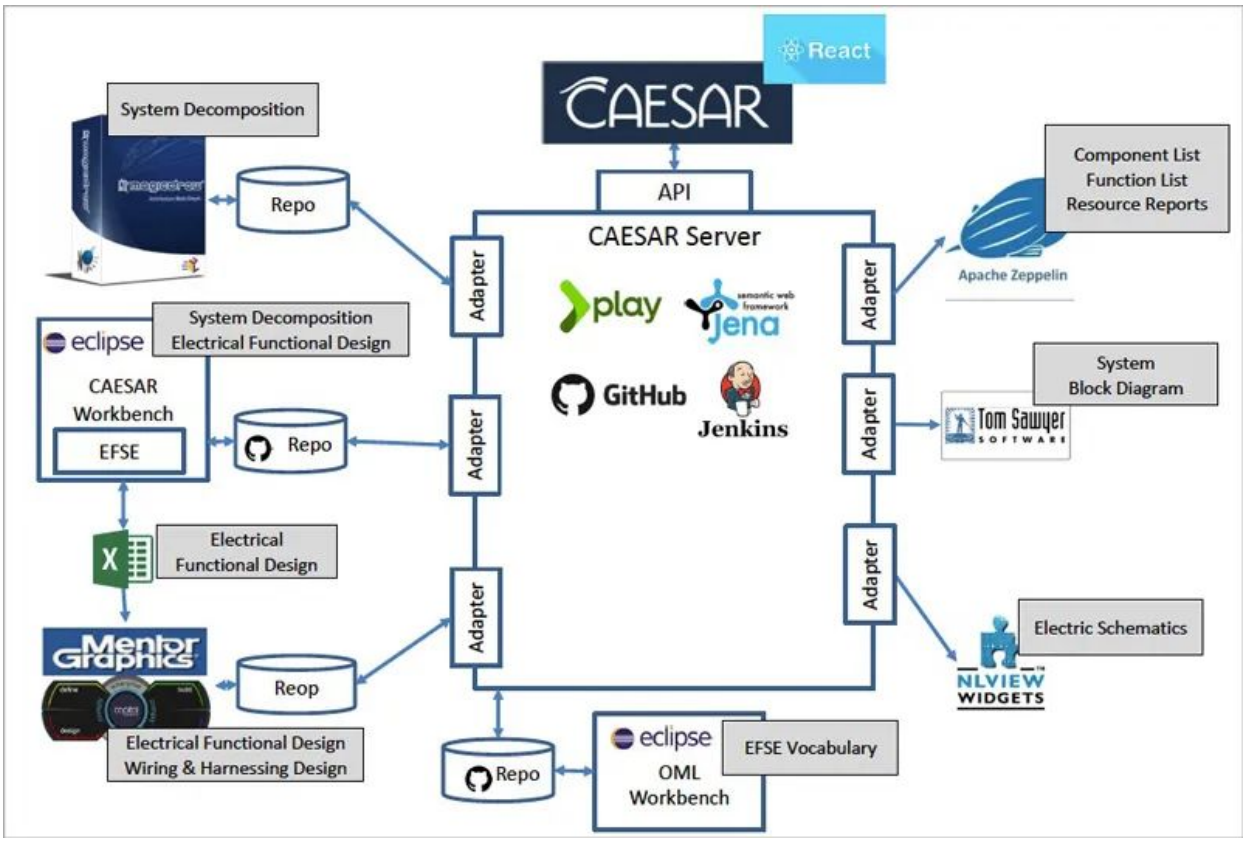


図 2.3　CAESAR 概要図 [23]

2.3　MBSE による知識再利用の課題

MBSE とオントロジーの利用により、エンジニアによる解釈のバラつきの防止や機械解釈性が向上し、設計情報の再利用性は向上した。一方で、MBSE は設計情報を現在のプロジェクトでどう記述、管理するかが焦点になっており、知識再利用に関しては下記の機能を有していない。

- （別プロジェクトでのエンジニアが）認知するべき知識を認知させる機能
- どのような条件であれば記述された設計情報や知識を再利用してよいかをエンジニアに提示する機能、あるいはふさわしい条件での設計や知識を再利用を促す機能

これらの機能は、従来の MBSE 手法では考慮されず、達成されていない。上記の機能がないがゆえに、オントロジーにより再利用性が向上したモデルがうまく活用されないという危険がある。そのため、過去に検討で考慮した不具合リスクやそれに対する対策なども、再利用されない可能性がある。そのため、これらの過去の知識に対する認知や、再利用にふさわしい条

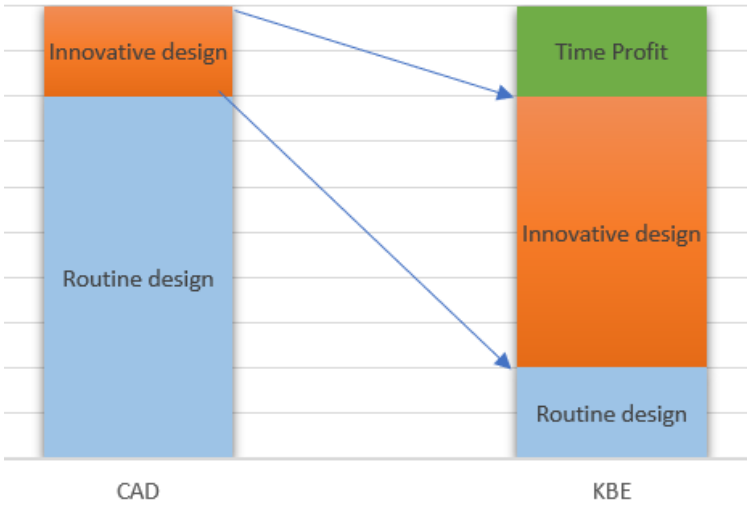


図 2.4 KBE による設計開発プロセスへの効果 [26]

件、コンテキストを提示する機能を実現し、MBSE で蓄積されたモデルを利用性を上げる必要がある。

2.4 Knowledge Based Systems

Knowledge Based Systems (KBS) とは、人間の知識を知識ベースと呼ばれるデータベースに保存し、設計や製造、意思決定、解析、診断などの人間の活動を、インタラクティブなプロセスでサポートするシステムを指す。Expert Systems や Computer Aided Design (CAD) をはじめ、多くの手法が研究されてきた [25]。

近年、ユーザーニーズの移り変わりの激しい市場において、産業は大量画一生産から、大量個別生産にシフトしている。ユーザーニーズの多様化や変化に適応するために、KBS を用いた設計開発 Knowledge Based Engineering (KBE) が有効であると考えられている。その理由として KBE には以下のようなメリットがある。

1. 過去の知識を利用し設計の自動化などが実現できる
2. 自動化により設計開発にかかる時間を短縮することが可能である
3. エラーを早期に発見し、製品を最適化できる
4. 縮小した時間でエンジニアは創造性の高い領域に集中できる

単純な CAD システムに比べると、KBE によるシステム開発は図 2.4 のように、スケジュールの短縮や、創造性に貢献するとされている。KBE はこれまで多くの分野に適用されてきた。例えば、[27], [28] では、KBE を用いた船の設計プロセスの保存、意思決定の自動化が提案さ

れている。航空宇宙では、[29] が KBE の手法を適用し、航空機の胴体パネルの設計プロセスを効率化し足り、[30] が胴翼航空機の設計の自動化を実現している。

KBE の手法は以下のパターンに分けられる。

2.4.1　パラメータベース

パラメータベースの KBE では、システムがパラメータの値を読み取り、不等式や Fuzzy Logic に保存した知識に基づき、診断や設計などの自動化を行う手法である。例えば [31] では、患者から測定したデータをパラメータとして入力として、冠動脈疾患となっている確率を、知識として入力した計算式から求める手法を確立している。また、[32] では、電気水中ポンプの故障診断において、Fuzzy Logic を使用し、知識を関数に落とし込み、運用データから故障モードの特定を可能にしている。[33] では、システムの故障リスクとその波及効果、影響度を洗い出す Fault Mode and Effect Analysis (FMEA) という活動において、その故障を特徴づけるパラーメータの動きを同時に記載しておくことで、後にパラメータの挙動から不具合と波及効果を推論できるようにしている。

2.4.2　オントロジーベース

オントロジーベースの手法では、単純な文章として記載していた過去の知識の検索性を向上し、再利用する手法が提案されている。例えば、[34] では、産業機械に関連する単語について、オントロジーを構築し、それを用いて過去において起きた仕様調整の問題を残すことで、別の仕様調整の際の使用文書から関連する過去の問題事例を取得しやすくした。また、[35] では、造船業における不具合レポートを自然言語処理で解析し、オントロジーを用いて構造化することで、単純な文字列検索よりも文書の検索性を向上した。　[36] は、システムの故障に関するオントロジーを構築し整理することで、システムにおける内部部品の構造、関係をもとに不具合の波及効果の推論や、原因の推論を可能にした。

2.4.3　ケースベース (Case-Based Reasoning (CBR))

CBR とは、現在の要求に当てはまる、過去の類似の事例を取得し、設計支援や診断を行う手法である。基本的なプロセスは図 2.5 のようなプロセスであり、オントロジーによるマッチングやパラメータベースの計算を用いて、過去の事例と類似度の計算を行い、類似度の高いものをユーザーに提供する。例えば [37] では、要求をオントロジー用いて記述し、それに結び付く過去の事例を取得することで設計支援を実現している。また、[38] も同様に、設計問題をオントロジーで記述し、その設計問題に対して最も類似度の高い過去の設計を取得し、ユーザーの設計支援を実現している。

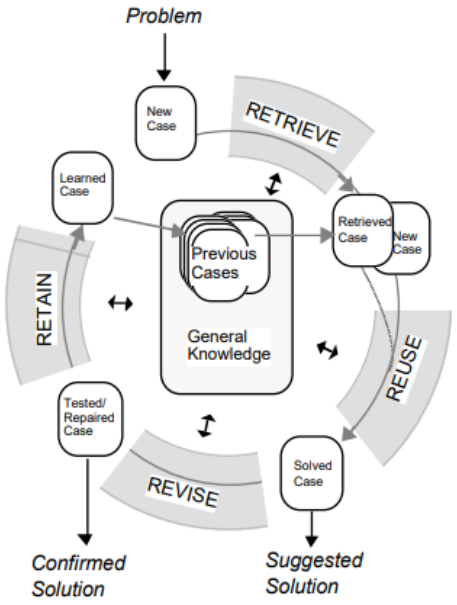


図 2.5 Case Based Reasoning のサイクル [39]

以上のように。KBE では様々な形で知識を活用し、人間の活動を支援する研究がなされている。一方で、パラメータベースの手法は、知識となるルールを追加、変化することを想定しておらず、また想定する開発対象のアーキテクチャも画一的である。オントロジーベースの手法や、ケースベースの手法は、定性的な概念の関係のみを扱うものが多く、パラメータと組み合わせより深い推論や自動化を実現できていない。また、複数種類の知識を組み合わせて、新しい知見を得るといったことも想定されていない。

2.5 Recommendation System

Knowledge Based System 以外にも、ユーザーに知識を与えるシステムとして、Recommendation System があげられる。Recommendation System は、Web 上の商品販売サイトなどにおいて、カスタマーが情報を発見することを支援したり、購買に関してサジェスチョンを行うことが可能である [40]。実装方法としてユーザー単体の過去の購買履歴のみに基づき、制約条件やルールによって提案する方法、複数人の購買履歴から、同じような購買行動をとる人の購入品を提案する方法、地域や年齢から無差別に提案する方法などがある [40]。近年、Amazon や Facebook を中心に、大量のデータを持つ e-commerce サイトでこれらの手法は機械学習の手法を用いて発展を遂げている [41][42]。Amazon では、AutoKnow と呼ばれる手法により、商品の特徴量を自動で学習し、商品とユーザーの関係が形づくる Knowledge Graph を補完することで推奨を行っている。これらの手法は、精度を上げるために非常に多くの試験ケース、データが必要となる。

2.6　先行研究のまとめ

これまで述べた先行研究の特徴を表 2.1 にまとめる。表の行方向に手法を、列方向に評価軸を並べている。なお、評価軸は、「過去のプロジェクトに出てきた様々な知識を、現在のプロジェクトに再利用する」という目的からブレイクダウンされる要求から決定した (要求については 3.1.1 章で詳細を述べる)。なお、1 行目の要求を先行研究が満たしていれば○、満たしていなければ×、制限付きであるが実現できる、あるいは不明な場合は△としている。

まず、MBSE 手法はオントロジーを用いてツール間での再利用性等が向上してきている一方で、ユーザーがどのモデルが再利用可能か、その条件は何かを認知する必要があるという欠点がある。データベースへの知識の蓄積も同様に、ユーザーへの知識の認知が必要とのなる。一方で、KBS で用いられるパラメータベース、オントロジーベース、CBR は、ユーザーの入力に対して、知識が自動でアクションを起こすため、知識に対する認知は必要ない。一方で、パラメータベースの推論の場合、定性的な機能やコンポーネントの関係性は考慮していないものが多く、適用先のシステムのアーキテクチャが異なる場合に使用できないと考えられる。また、パラメータの不確定性については、計算式で考慮できるものの、対象のアーキテクチャに対する柔軟性は低い。逆にオントロジーベースの推論では、パラメータまで踏み込んで定量的な関係まで表現できる手法は調査した限りではない。Case-Based Reasoning では、一部定量的なパラメータの近さを考慮して過去の事例を取得する研究は提案されているものの [38], 複雑なパラメータ解析までは考慮していない。Recommendation System は、柔軟に知識を際量でき。ユーザーの労力は少ない一方で、ニューラルネットワークを学習し、利用できるまでに必要なデータ数が非常に多くなってしまう。

本研究では、これらの先行研究に対し、知識を記述労力は若干必要なものの、ユーザーの知識認知を必要とせず、さらに不確定性や知識の曖昧さ、定量的なパラメータと、機能、コンポの関係性をすべて考慮できるフレームワークを検討する。また、事例がすくなく、知識蓄積に時間がかかる宇宙システム開発で使用できるため、データが少なくても使用可能なフレームワークを提案する。

また、プロジェクト間での効率的な知識再利用において重要な、下記の点が先行研究では検討されていないため、それらについても検討、考慮する必要がある。

1. 再利用できる知識とともに、宇宙システムにおけるどの部分（コンポーネント、機能) に、知識が再利用できるかを提示すること。
2. システムの規模、データの蓄積数に応じて、大量の知識が再利用される可能性があり、重要な知識をそこから識別できるようにすること。
3. 設計に関する知識や不具合に関する知識等を組み合わせて、複雑な知識を再利用する

表 2.1 先行研究との比較表
(1 行目の要求を先行研究が満たしていれば○、満たしていなければ×、制限付きであるが実現できる、あるいは不明な場合は△としている)

手法	知識に対する認知が必要ないか	不確定性や曖昧さを含んだ知識の定義が可能か	定量的パラメータに関する知識が記述できるか	機能やコンポの概念基づき知識が記述できるか	大量のデータを集めることが難しい宇宙領域で適用できる	知識蓄積の労力
MBSE	×	×	○	○	○	×
データベース	×	×	○	○	○	○
パラメータベース	○	△	○	×	○	×
オントロジーベース	○	×	×	○	○	×
Case-Based Reasoning	○	×	△	○	○	×
Recommendation System	○	○	△	○	×	○
提案手法	○	○	○	○	○	×

こと。

特に 3 は、提案手法が、専門家のレビューで行われるプロセスを置き換えるうえで重要であると考えられる。単に不具合の知識だけを提示するのではなく、与えられているコンテキストや設計情報を深堀り、詳細化し、エンジニアが見えない知識を再利用する必要がある。

第 3 章

提案手法

3.1　提案手法の概要

3.1.1　機能要求の確認

第 1.6 章に示した通り、本研究の目的は、宇宙システム開発の効率性および信頼性を上げるために、ユーザーの認知やコンテキストの誤りのない効率的かつ効果的な知識再利用を実現することである。この目的を達成するために満たすべき要求を整理すると表 3.1 のようになる。

要求 1 については、データベースやドキュメントのような、エンジニアが「何が必要か」を認知する必要が無いようにする必要がある、という要求である。 要求 2 については、コンテキストに対して適切な知識を再利用するという目的を、定性的な関係と定量的な解析、また不明確な知識などを包括的に扱うことができる必要がある、というものである。また、要求 3 は、従来専門家のレビューのように、設計情報の深堀りや詳細化を行いつつ不具合に関する知識を提供し、複数の知識を組み合わせて複雑な知識を再利用できることを指す。要求 4 は、他の要求を満たす手法が、宇宙システム開発プロジェクトの中でリーズナブルなリソースで実施できる必要がある、という要求となる。そして、要求 5 は、宇宙システムで適用できるため、機械学習のような事前学習を必要としないようにする必要がある、という要求となる。この提案手

表 3.1　本研究の提案手法に対する要求.

要求 1	必要な知識をユーザーが認識しなくても獲得することができる
要求 2	コンテキストを踏まえて様々な知識 (物理関係、パラメータ、不明確) が適切に再利用できる
要求 3	不足した情報の補完、複数の知識を組み合わせが可能である
要求 4	効率的に知識を再利用できる
要求 5	大量のデータを集めることが難しい宇宙領域で適用できる

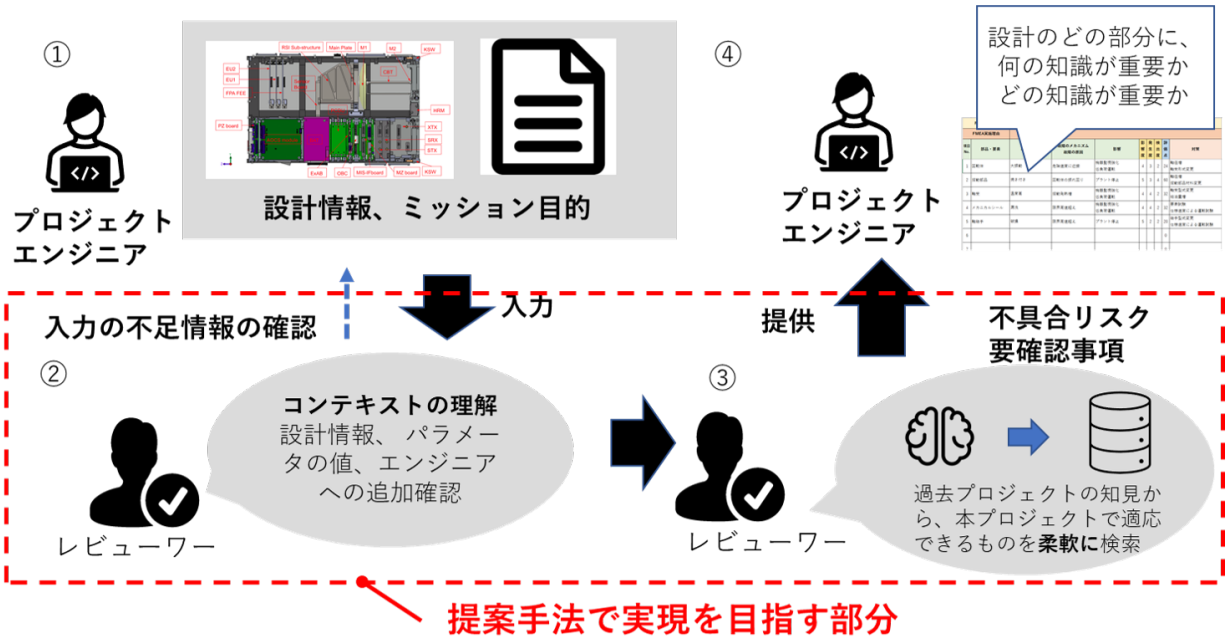


図 3.1　提案手法により実現する知識利用の概念図.

法が知識提供を行うコンセプトを図 3.1 に示す。

3.1.2　提案手法の全体像

ユーザーが知識を認知していなくても、コンテキストを考慮し、かつ属人性を排除して知識を再利用するため、本研究では次のようなアイデアを利用する (図 3.2)

- 人間が、必要な知識を検索するのではなく、人間がコンテキストを入力し、それに対して知識側が自律的に、今のコンテクスト (ユーザーの立場、宇宙機設計) から再利用可能な条件に当てはまる部分を検索し、適合すればユーザー前に現れる。

ユーザーが探すのではなく、知識が自律的に表れることによって、ユーザーの認知や属人性の問題は解決される。さらに知識側がユーザーの様々なコンテキストを把握し判断できる機能を持つことによる、コンテキストに適合した知識をユーザーに提供することができる。また、専門家とのコミュニケーションも必要ないため、効率的な知識再利用が可能になると考えられる。

上記のアイデアを実現する手段として、以下の手法を提案する。

1. 知識格納のフォーマット (知識分子)
2. 衛星設計を記述するための基礎となるオントロジーと、それを用いた宇宙システムの情

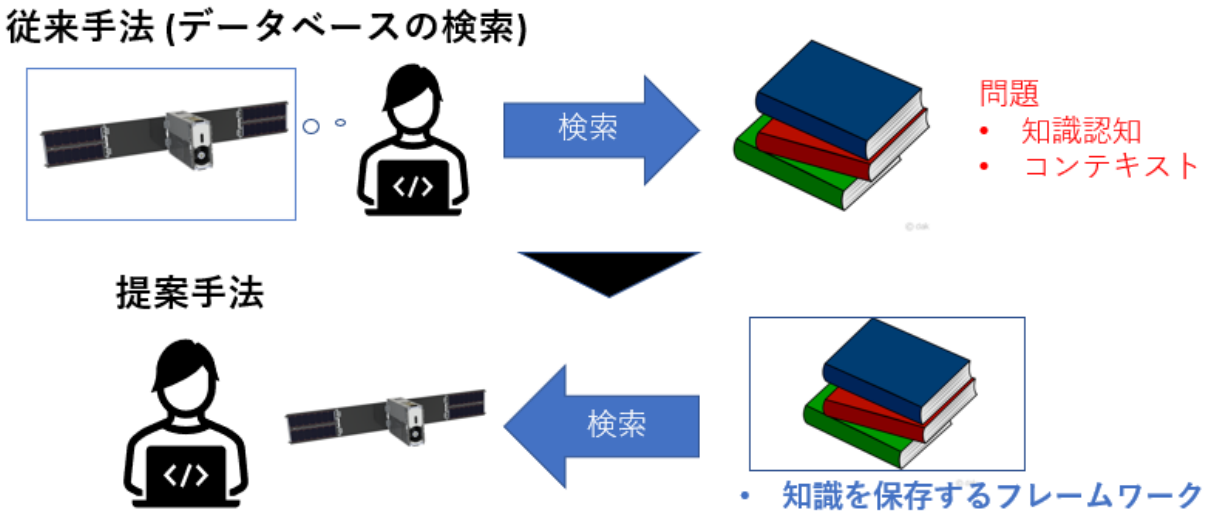


図 3.2　提案手法のアイデア.

報のモデル化

3. 構築した宇宙システムのモデルに知識分子を結合するエンジン
4. 統合した知識のうち、重要な知識を識別するアルゴリズム
5. 以上を活用し、知識を再利用するプロセス

各要素については、第 4 章から、第 8 章で詳しく述べる。

3.1.3　提案手法による知識提供の流れ

提案手法により実際に宇宙システム開発プロジェクトにおいて、必要な知識をエンジニアが再利用するフローとしては下記のようになる。

1. 開発を目指す宇宙システムの設計情報や要求情報を、オントロジーを用いてモデル化する (以降、このモデルを宇宙システムモデルと呼ぶ)。
2. 構築した宇宙システムに関するモデルを、提案するアルゴリズムを保有するシステムに入力する。
3. これまで登録された知識が、知識分子として宇宙システムのモデルに結合する。
4. 結合した結果、どの場所にどのような知識分子が結合するかをユーザーに提供する。

これを図に示すと図 3.3 のようになる。いかに実際にあるプロジェクトで知識が発生し、本提案手法により再利用し活用するまでの具体例 (図 3.4) を示す。

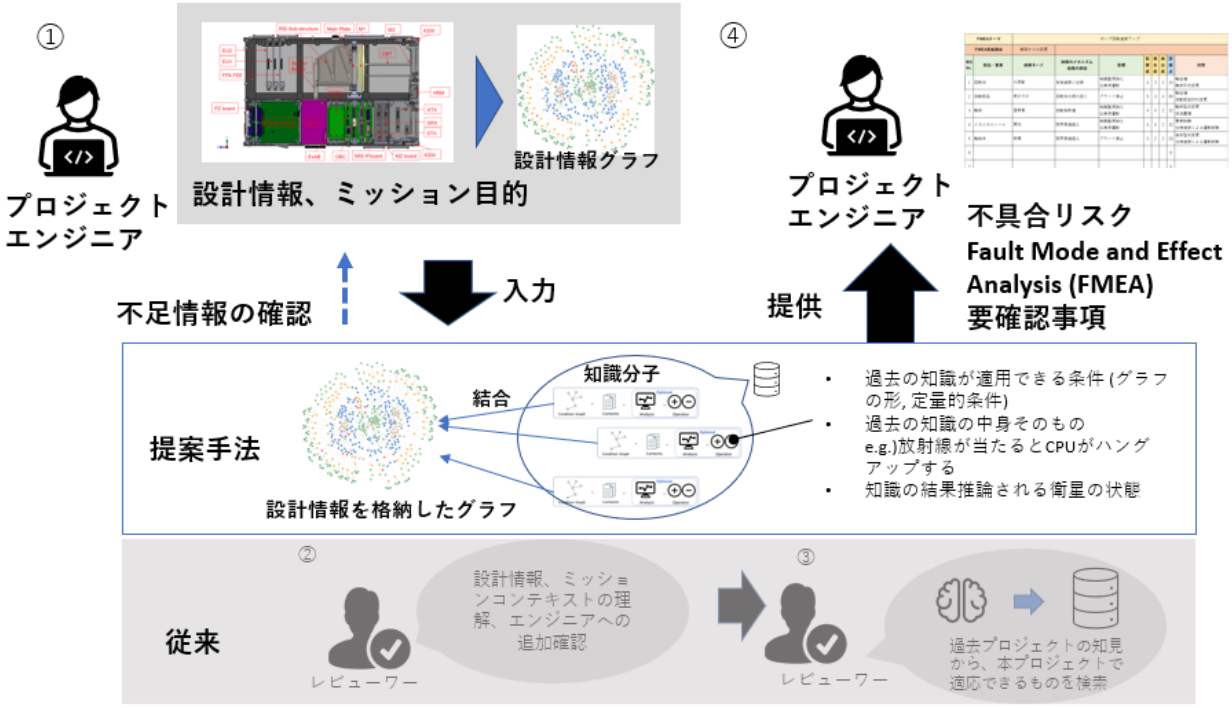


図 3.3　提案手法により実現する、知識提供プロセス

3.1.3.1　プロジェクト A での不具合発生

プロジェクト A で、衛星の開発途中にハーネスが断線する不具合が発生した。原因を調査すると、ハーネスを流れる電流量が、ハーネスの直径によって決定する定格電流量を上回っていることが原因と分かった。この不具合から得た教訓として、「設計時に電流量が、ハーネスの定格を上回っていないか確認すべき」ということがまとめられた。

3.1.3.2　知識の知識分子への落とし込み

この不具合に関する知識を再利用し、今後別のプロジェクトで確実に生かすために、知識分子に落とし込むことを検討する。知識分子では、知識が当てはまる条件として、定性的な概念の関係と、定量的なパラメータの関係を入れ込める。今回定性的な条件として「ハーネスに電流が流れていること」、定量的な条件として、「ハーネスの太さから計算される定格を、電流量が上回っている」として記述した。再利用される知識の中身として、「不具合：定格を上回る電流により、ハーネスが焼き切れる、対策：電流量の太さを確認」を記述した。

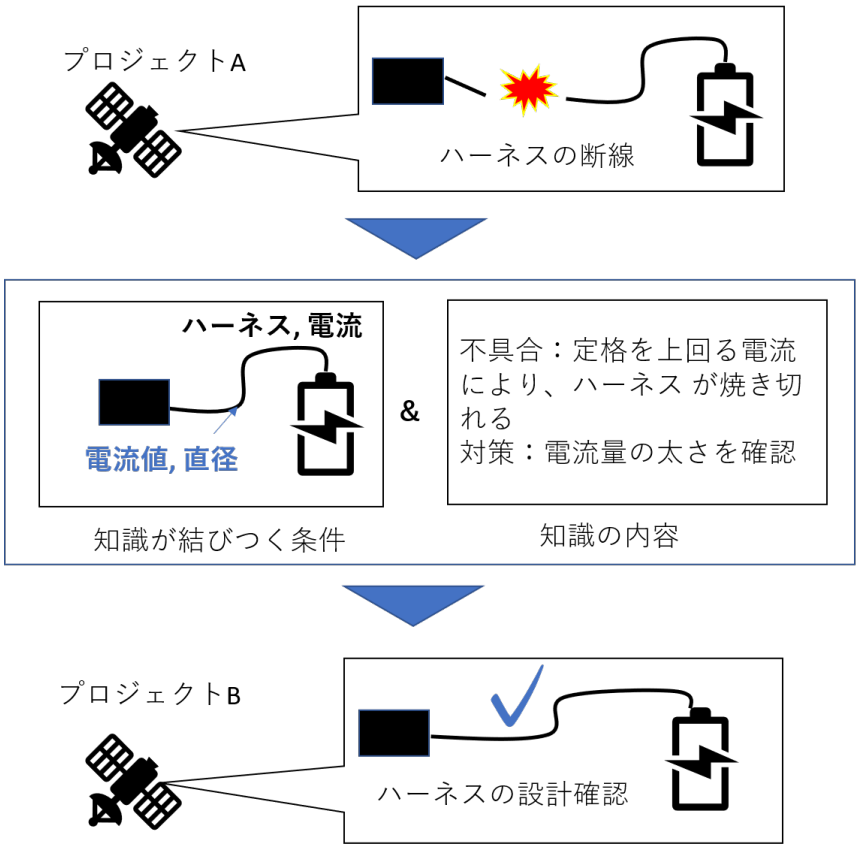


図 3.4　衛星においてハーネス断線の不具合が起きた際の、提案手法による知識再利用プロセス.

3.1.3.3　別プロジェクト B での設計リスク確認

プロジェクト B では、設計フェイズが一回りし、設計した宇宙機に故障リスクがないかを確認するべく、Fault Mode and Effect Analysis の整理を実施した。この時、設計の全範囲を、見落としがないかを確認するために、知識分子を用いた不具合に関する知識の再利用、それによる設計不具合リスクの確認を実施した。その際、衛星を構成するコンポーネント、コンポーネントのつながり、電流、電圧値、ハーネス太さを定義した衛星全体のモデルを入力した。

3.1.3.4　知識の再利用

プロジェクト A でのハーネスに関する不具合を、知識分子に落とし込んでいたため、プロジェクト B での知識分による設計リスク確認により、「リアクションホイールと電力分配器の間のハーネスに、定格を上回る電流が流れて断線する可能性がある」ことが発見された。確認した所、確かに断線するリスクがあることが分かったので、設計を修正した。

以上のようなプロセスで、本研究で提案する知識分子により、宇宙システム開発プロジェク

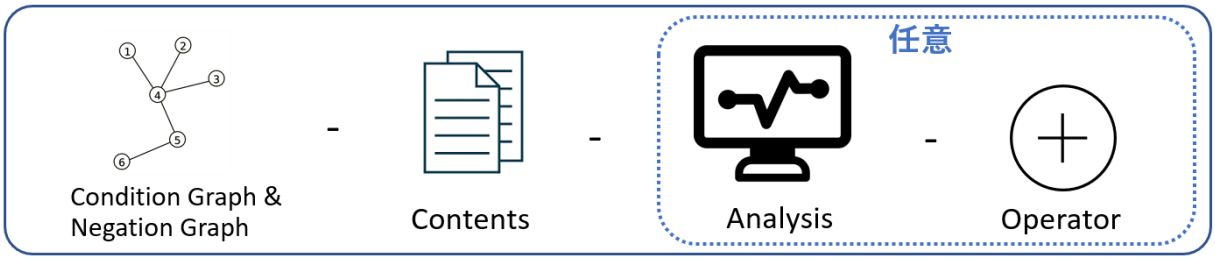


図 3.5　知識分子の構成.

トにおいて出てきた知識は再利用される。

3.2　提案手法 知識分子

3.2.1　知識分子の概要

本研究では、宇宙システム開発プロジェクトに生成される知識を保存するフォーマットとして、知識分子を提案する (図 3.2.1)。知識分子は下記のような特徴を持つ。

1. 知識が開発中のシステムに当てはまるかの条件を、Condition Graph として、機能や構成を表すグラフ構造で保存できる。
2. 知識の内容そのものを Content に保存することができる
3. Analysis と呼ばれる定量解析を行う部分を持っており、パラメータの値など定量的な値をもとに知識が当てはまるかを判定できる。
4. Operator によって、知識分子が結合した際、結合先となるシステムのグラフに変化を与えることができる。これにより、故障知識を用いた故障の波及効果等を推論することが可能である。

なお、知識分子において Analysis と Operator は知識分子を構成するうえで任意であり、必要に応じて設定することが可能である。

以下の節では、知識分子の各要素について詳細に説明を行う。

3.2.1.1　Condition Graph

知識分子の Condition Graph は、知識が現在のプロジェクトに適合できるかどうかの条件を、プロジェクトで開発するシステムの構成要素や、それらの機能、パラメータと、それらの関係をもとに定義する部分である。グラフのノード、エッジは、後述のオントロジーを用いて記述され、知識分子の結合アルゴリズムによって、類似の形状を持つ宇宙システムモデルと結合を

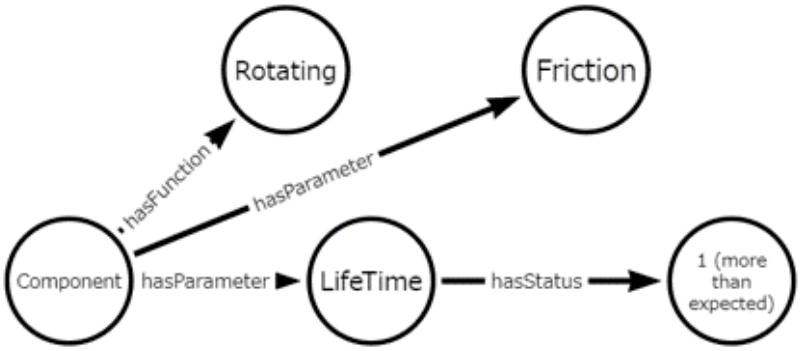


図 3.6　Condition Graph の例.

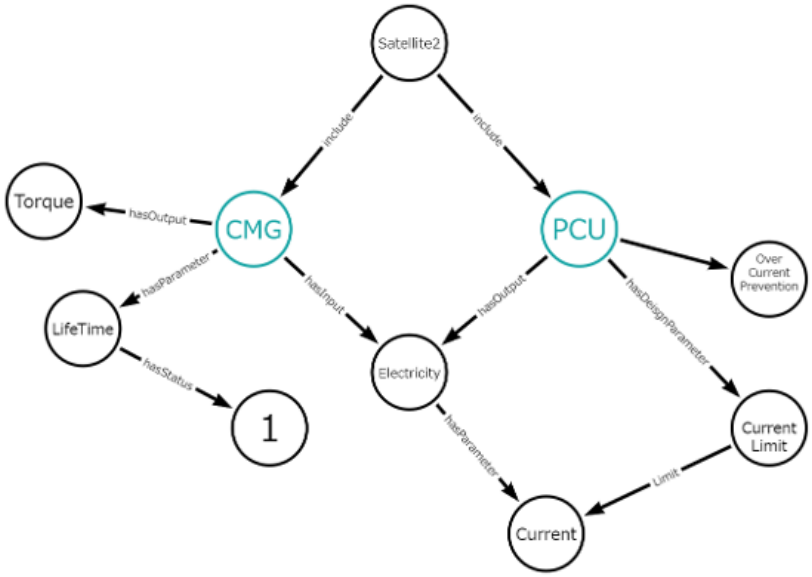


図 3.7　CMG が PCU から電源供給されている場合の知識分子の Condition Graph

行う。以下の図 3.6, 図 3.7 に簡単な例を示す。

図 3.6 で示されるグラフが Condition Graph である場合、宇宙システムに含まれるコンポーネントが以下の条件を満たすとき、知識が結合するという条件を表している。

1. コンポーネントが、“Roatating” (回転機能) を有する
2. コンポーネントが、“Friction” (摩擦) というパラメータを有する
3. コンポーネントは、“LifeTime” (寿命) というパラメータを有する
4. “LifeTime”が 1(期待以上の値) というステータスになっている。

上記の条件を満たす状況として、人工衛星が Reaction Wheel を搭載している状況が考えられる (Reaction Wheel と呼ばれる姿勢制御装置は、一般的に回転機能をもち、これらのパラメータが重要となる)。また別の例を 3.7 に示す。

表 3.2　Concent に書かれる内容の例

ID	知識の種類	Content
1	故障リスク	Reaction Weel (RW) 型番 00A を磁気センサー型番 RM9300 の傍で回転させると磁気センサーにおおきなバイアスが乗る
2	設計に関する基礎知識	Power Control Unit には電子回路と半導体が含まれる
3	検証方法に関する知識	バッテリー寿命検査にはバッテリーの漏洩試験をする

1. 衛星が、Control Momentum Gyro (CMG)、Power Control Unit (PCU) を保有している
2. PCU から、CMG へ電源共有されている
3. コンポーネントは、“LifeTime” (寿命) というパラメータを有する
4. そのほか周辺パラメータの定義

上記の条件は、CMG を行って姿勢制御を行うような衛星に適合できると考えられる。また、知識分子は、知識が結合しない条件を Negation Graph としてグラフで表現することも可能である。もし、前述のグラフが Negation Graph として知識分子に定義されていた場合、Condition Graph にどのような結合条件が書かれていても、CMG を持つ衛星には知識は適用されない。以上のように、

- システムを構築するコンポーネント
- システムでやり取りされている媒体
- システムにかかわるパラメータ・機能

をノードとし、

- コンポーネント同士の関係
- パラメータ・機能の

をエッジとするグラフで、知識の結合条件を表したものが Condition Graph (or Negation Graph) である。

3.2.1.2　Content

Content は、知識分子が提供する知識そのものであり、一般的には自然言語で記述される。例えば、不具合リスクに関する知識に関しては、Condition Graph に当てはまるような宇宙システムにおいて起こりうる不具合の内容を自然言語で書いたものとなる。表 3.2 に Content に含まれる内容の例を追加で示す。なお、一つの結合条件で、複数種類の知識内容を再利用、結

合させたいときは、Content 複数種類定義することが可能である。

3.2.1.3 Analysis

Condition グラフは知識分子の結合条件を、機能・実体の構造により表現できる一方で、グラフ中に含まれるパラメータを用いて不等式や等式により定量的に結合条件を表すことはできない。また、後述する Operator により、宇宙システムの情報となる宇宙システムモデルに、変化を与える際も、パラメータの値に応じて知識分子が変更するパラメータの値を解析結果により変えたい場合がある。

例えば、宇宙機において、コンポーネントの消費電力、送信電力に合わせて、それらの電力を受け渡しに必要な配線の太さの最小値は決定され、それ以上の太さを持っていない場合は焼き切れるリスクがあるという知識がある。この場合、コンポーネント間でやり取りされる電力のパラメータと、配線太さのパラメータをもとに、リスクがあるかどうかを判定する必要がある。

このような知識の結合条件を表現するため、知識分子は Analysis と呼ばれる部分を持つ。Analysis は以下の内容を記述する。

- 関数の入力となる、Condition グラフ中に記述されたパラメータ
- 解析を行う関数。出力として、次の Operator に使用するパラメータ値、知識の適合性の判定結果の二つを持つ
- Operator に使用する関数の出力結果

下記に例を示す。

例 1

- 入力: 二つのコンポーネントの温度
- 解析: 二つのコンポーネントの温度差から、移動する熱量、移動方向を出力
- 出力: 二つのコンポーネントの熱量の変化

また、Analysis は、Condition グラフから値を受け取らずに、ユーザーからパラメータの入力を問い合わせることも可能である。これにより、後述する Operator と組み合わせることで、知識が結合する宇宙システムのモデルにおける情報を補完することが可能になる。下記に例を示す。

例 2

- Condition Graph: コンポーネントとコンポーネントがねじで構造的に締結されている
- Negation Graph: ねじサイズの情報が定義されている
- 入力: なし

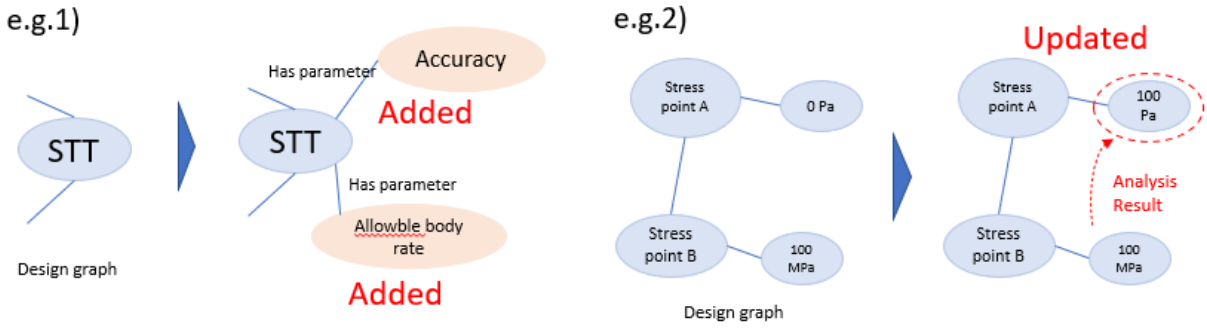


図 3.8　Operator の例 (左: Star Tracker (STT) へのパラメータ定義、 右: コンポ圧力の伝搬)

- 解析: ユーザーからねじサイズの入力を要求
- 出力: コンポーネントネントの締結に使われているねじの情報を追加する

3.2.1.4　Operator

知識分子は、前述の Condition Graph、Analisys で判定を行い、結合した箇所と、Content をセットで記録し、ユーザーに提供することで、ユーザーに知識提供を行うことができる。一方で、複数の原因が連鎖することで発生した不具合に関する知識など、複雑な知識を再利用する場合、Condition Graph にそのまま表現しようとするとグラフが巨大化してしまう。巨大化した Condition Graph は、正確な記述が難しいだけでなく、適合するような宇宙システムの数は少なくなり、再利用性が悪くなる。例えば、宇宙システムにおけるある個所が、機能や実態の結合条件によって故障した際に、その故障がさらに特定の条件によって波及し別の故障を引き起こすような知識は、条件が二重になり、表現が難しくなる。そこで、このような複雑な知識を表現できるようにするため、ある知識が結合した結果や、結合した知識の内容から導かれる宇宙システムの状態の更新を、宇宙システムモデル側に残せる機能を提案する。その機能を担うのが、知識分子における Operator となる。

例えば、Operator により図 3.8 の e.g.1 のように設計グラフ上にパラメータの定義を追加することが可能である。この例では、宇宙システムの設計情報を表す宇宙システムモデルに STT(スタートラッカ) があることを条件とし、結合した先に、Accuracy や Allowable Angular Rate の定義を追加する Operator となる。パラメータの値は、事前に知識分子に定義しておくこともできる一方で、Analysis により結合先の別のパラメータから計算したり、結合時にユーザーへの入力を求めたりすることが可能である。e.g.2 では、Operator により、パラメータを更新する例である。あるシステムにおけるコンポーネントにかかる圧力が伝搬する様子を、Operator により表現している。

3.3　提案手法 オントロジー

3.3.1　宇宙システムのモデル化

本研究の提案手法では、知識分子の Condition Graph、およびそれと結合する宇宙システムモデルを記述する必要がある。また、計算機を用いた知識結合のために、機械解釈可能な形でモデル化する必要がある。その際、宇宙システムをモデルとして表現するための語彙や文法が、知識分子で記述する結合条件 (= Condition Graph) と異なってしまうと、機械的に結合することができないため、それらを統一的に記述できるようにモデル化する必要がある。

そこで、本研究では宇宙システムモデルと知識分子で共通で用いられる基盤となるや概念の関係を、既存研究を参考に、オントロジーとして定義する。

3.3.2　オントロジーとは

オントロジーとは、哲学の分野に由来する言葉で「存在とは何か」を問う存在論を表す言葉である。一方で、人工知能の分野では、オントロジーは「概念の明示的記述」と呼ばれ、対象世界のとらえ方に関して計算機と人間で共有できる形で定義する方法を表す。本研究では後者の定義でオントロジーという言葉を用いる [43]。

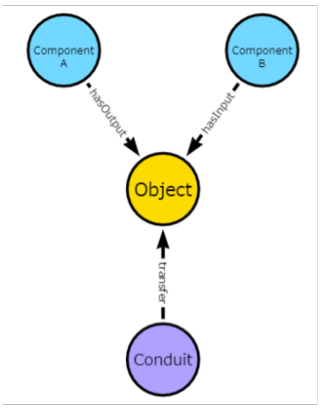
人工知能の分野において、オントロジーは、以下の二つで構成される。

- 対象世界を世界を説明する「概念」
- 概念間の「関係」

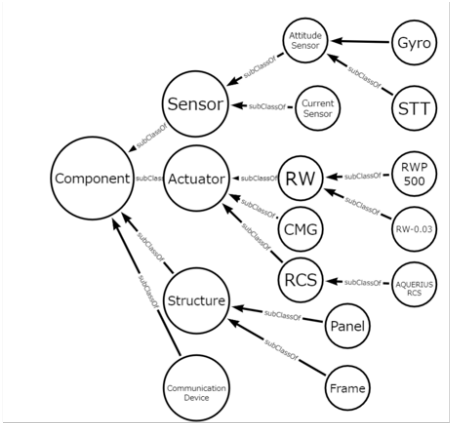
上記の例として、例えば、人工衛星の世界では「人工衛星」と「はやぶさ」という概念があり、それらは「抽象→具体」の関係がある。これを「はやぶさ is-a 人工衛星」と明示的にデータとして定義し、データとして利用するのがオントロジーである。上記のような例以外でも、定義される概念とそれらを結ぶ関係には表 3.3 のようなものがある。

表 3.3　オントロジーにおける概念と関係

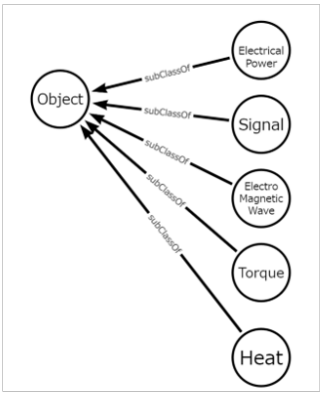
ID	概念	関係
1	上位概念 or 下位概念	is-a 関係
2	部分概念	part-of 関係
3	属性	attribute-of 関係
4	公理	その他の関係



拡張デバイスオントロジー



衛星オントロジー



物理機能オントロジー

図 3.9　本研究で定義したオントロジーの種類と概要図.

3.3.3　オントロジーへの要求

本研究では、オントロジーを用いて、宇宙システムをモデル化し、知識の結合条件を表現することを目指す。知識分子の結合条件として記述できる範囲は、宇宙システムにおけるあらゆる知識が再利用できる程度であることが望ましい。宇宙システムは、基板や素子レベルの非常に細かいレベルから、システム全体レベルで成立性を検討する必要があり、宇宙システムのあらゆる知識を利用できるようにするためには、多様な粒度を表現できる必要がある。また、宇宙システムは、複合領域を組み合わせたシステムであり、多くの異なる分野から (時には、ロボットや自動車など、宇宙以外のシステムから) 知識を集める必要がある。この前提を踏まえた場合、使用するオントロジーには下記の要求を満たす必要がある。

1. 宇宙機システムの多様な知識の粒度を柔軟に表現することができる。
2. 多様な分野からの知識を受けれることができるようにする必要がある。

3.3.4　本研究で提案するオントロジー

上記の要求に対して、本研究では下記の方針でオントロジーの構築を行った (図 3.10)。

- 拡張デバイスオントロジーをベースとした、物体とその入出力によるシステムの表現
- 宇宙システムにおける部品、コンポーネントなどの概念の階層関係
- 拡張デバイスオントロジーで用いられるオブジェクトの種類や、コンポの機能の関係

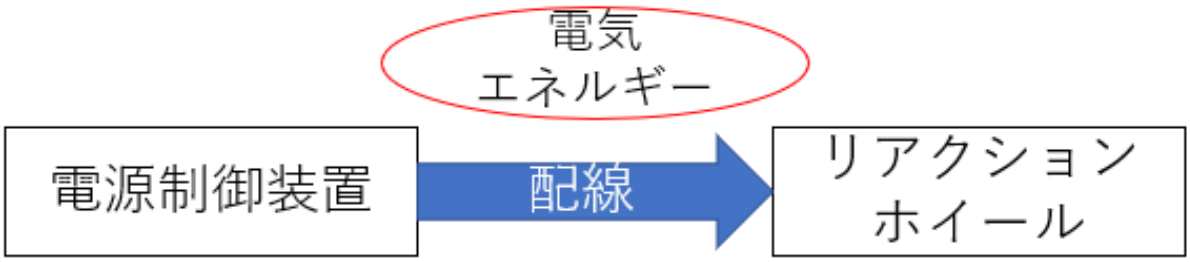


図 3.10　拡張デバイスオントロジーを用いて表現した場合の機器の接続表現例.

拡張デバイスオントロジーとは、来村らによって提案されたシステム記述のオントロジーの一つである [44]。このオントロジーではシステムを、入力と出力をもち、それらの変換を行うデバイスとそれらの組み合わせとしてとらえるオントロジーである。人工衛星をはじめとする宇宙システムは、部品やコンポーネントを組み上げて構築し、その際それらコンポーネントの入力、出力の挙動を踏まえて設計することが多いため、このオントロジーでの表現が容易である。またそれに加え、部品の内包関係を表現することで、システムの階層関係を柔軟に表現することができ、要求の一つである粒度の柔軟性を確保できることがある [45]。

さらに、拡張デバイスオントロジーでは、デバイスとともに、それらでやり取りされる「対象物」を概念としてとらえ、対象物が導管を通ってやり取りされるといいう表現を行う（図 3.10)。

入出力の「対象物」を基礎的な物理現象、工学的に必要な概念 (エネルギー、電気、情報など）にすることで、多様な分野でも共通に知識を表現することが可能となる。先行研究となる [45] では拡張デバイスオントロジーを姿勢制御なども含めて表現できるように拡張し、設計支援を提案している。

この拡張デバイスオントロジーのみを用いた場合、宇宙システムは物理的な関係をもとにモデル化され、知識の結合条件もそのような「コンポーネント・部品とそれらの入出力関係」を条件に記述することになる。これは、異なる分野で共通に知識を再利用することを可能にする。一方で、知識が適合する条件が、物理的なやり取りだけでは記述することが難しい場合がある。例えば、以下のような知識の例が存在する。

1. Gyro センサーは一般的に、バイアス推定試験をする必要がある。
2. あるコンポメーカ A から供給されたリアクションホイールと、コンポメーカー B から供給される電源制御基板は、組み合わせると電源がつかなくなるという不具合が、ごくまれに発生する (その原因は明らかにされていない)

1. の方は、センサーの一般的な試験方法に関する知識である。上記の知識の条件は、「Gyro セ

ンサー」であるが、これはどのコンポーネントなら適合するかを拡張デバイスオントロジーだけでは表現することができない。例えば、Fiber Optical Gyro センサーに、この知識が適用できるかを計算機が解釈することはできない。Gyro センサーという概念の下位の概念に具体的にどのようなセンサー (型番、種類) が含まれるかを表現したオントロジーを定義する必要がある。

2. の方では、前述と同様、リアクションホイールと電源制御基板に関する下位概念を定義する必要がある。それに加えて、「コンポーメーカー」という概念、コンポーメーカーとコンポーネント間の「供給者の関係」を定義する必要がある。これらの概念も拡張デバイスオントロジーに基づく、物理的なコンポーネント間の関係では表現することができない。

前述のような問題点があるため、本研究では、宇宙システム、およびプロセスにおける概念の階層関係、プロセスと物体の関係を表す表現を、新たに宇宙システムオントロジーとして独自に定義を行う。これにより、物理的な関係では表現できない、宇宙システム開発特有の知識の再利用を可能にする。

なお、注意点として、宇宙システムの部品、コンポーネントの概念の階層関係は定義する一方で、これらの概念間の付属関係や接続関係は、可能な限り拡張デバイスオントロジーに基づき表現する。これは、宇宙システムオントロジーとして、宇宙内部に閉じる関係の概念を定義すると、他の分野（例えば航空機、ドローンなど）の知識を再利用することが難しくなる、また、客観的に見て正しい知識を表現することが難しくなる、などの問題点があるためである。他分野からの知識の統合性と、知識の客観性を上げるため、物理的、あるいは機能的な関係を基にする拡張デバイスオントロジーをベースに宇宙システム、知識分子の結合条件を記述する。詳細は、3.6 章で説明を行う。

3.4　提案手法 知識結合アルゴリズム

3.4.1　知識結合アルゴリズムへの要求

3.3 章のオントロジーを用いて、知識分子の Condition Graph と、知識分子結合先の宇宙システムに関する情報を、計算機に入力することが可能である。今、知識分子は、知識の結合条件を Condition Graph, Negation Graph, Analysis の形で、計算機が理解できる形で保持している。ここから知識分子を活用し知識を再利用するためには、入力された宇宙システムが、知識分子に示された結合条件にマッチ部分を持つかを判定し、Content となる知識をユーザーに提示するアルゴリズムが必要となる。また、知識分子に定義される Operator を用いることで、複数の知識を用いて複雑な知識を再利用するアルゴリズムが必要である。

また、人間であるユーザーが、知識分子により知識が再利用される際に、なぜその知識が結

合したのか、どの部分に結合したのかがわからないと、効率的な知識の再利用を行うことができない。よって、知識が結合した際は、結合条件、知識の中身とともに、宇宙システムのどの部分にその知識が当てはまるかを知らせる必要がある。

さらに、宇宙システム開発に関わる知識の結合条件は、いつも厳密に指定できる訳ではない。例えば、ある条件で不具合が頻発し、その原因がわからないときは、ひとまずその状況に近い条件では、故障が起こるリスクがあるとして知識を保存したい場合がある。その場合、Condition Grpah に記述した要素、機能の関係に厳密に一致するものだけでなく、それに近しい構造をもつ宇宙システムがあるかどうかを判定することが求められる。また、結合条件などにおいて、同じ条件を表す場合でも若干の表記ゆれなどが存在するリスクがある。そのような、厳密に知識分子の条件に基づき結合を判定するのは難しい状況においても、知識を再利用できるようにするため、近しい状況であれば結合を判定することが求められる。

以上のような観点で、提案する知識結合アルゴリズムへの要求は下記の通りになる。

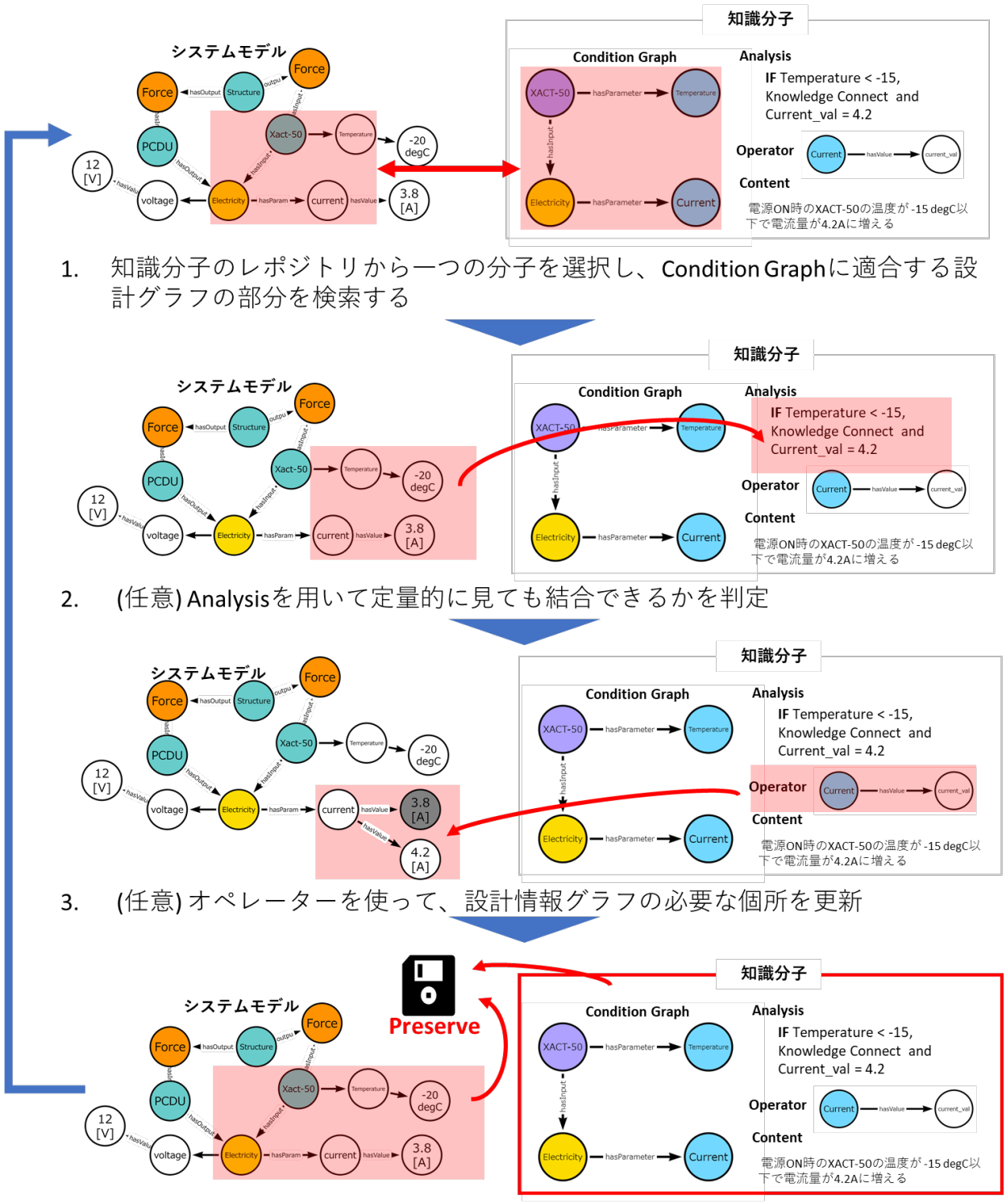
1. 計算機に入力される宇宙システムモデルと、知識分子の Condition Graph, Analysis を活用し、知識分子が適合するかを判定する。
2. 知識分子の結合が判定される場合に、宇宙システムモデルのどの部分で結合が発生するかをユーザーに示す。
3. 知識の結合条件が曖昧さや不確定性を含み、記述が厳密に規定できない問題を解消するため、多少条件に一致しなくても条件に近しい箇所で結合できるようにする。

3.4.2 知識結合アルゴリズム

前述の要求を満たすようなアルゴリズムを本研究では提案する。アルゴリズムの概要を図 3.11 に示す。この図に示したように、提案手法は大きく分けて 4 つのパートに分割することができる。

1. Condition Graph, Negation Graph による結合箇所の判定
2. Analysis による定量的結合
3. Operator による作用
4. 結合した箇所の情報とともに、結合した知識分子の Content の保存を行う

上記のプロセスは、知識分子を保存したデータベース中の知識分子一つ一つに実行される。1 の段階で知識分子が結合しないと判断された場合は、すぐに次の知識分子に移行し、上のプロセスを実行する。知識分子において、Analysis, Operator の定義は任意であるため、これらが定義されていない場合は 2, 3 はスキップされる。アルゴリズムの詳細のフローチャートを図 3.12 に示す。このフローチャートに示すように、知識の結合を行う際は、最大のループ数を定



1. 知識分子のレポジトリから一つの分子を選択し、Condition Graphに適合する設計グラフの部分を検索する
2. (任意) Analysisを用いて定量的に見ても結合できるかを判定
3. (任意) オペレーターを使って、設計情報グラフの必要な個所を更新
4. 結合場所とOperatorの結果、知識分子のIDを、出力ファイルに追加

図 3.11　知識結合アルゴリズムの全体像

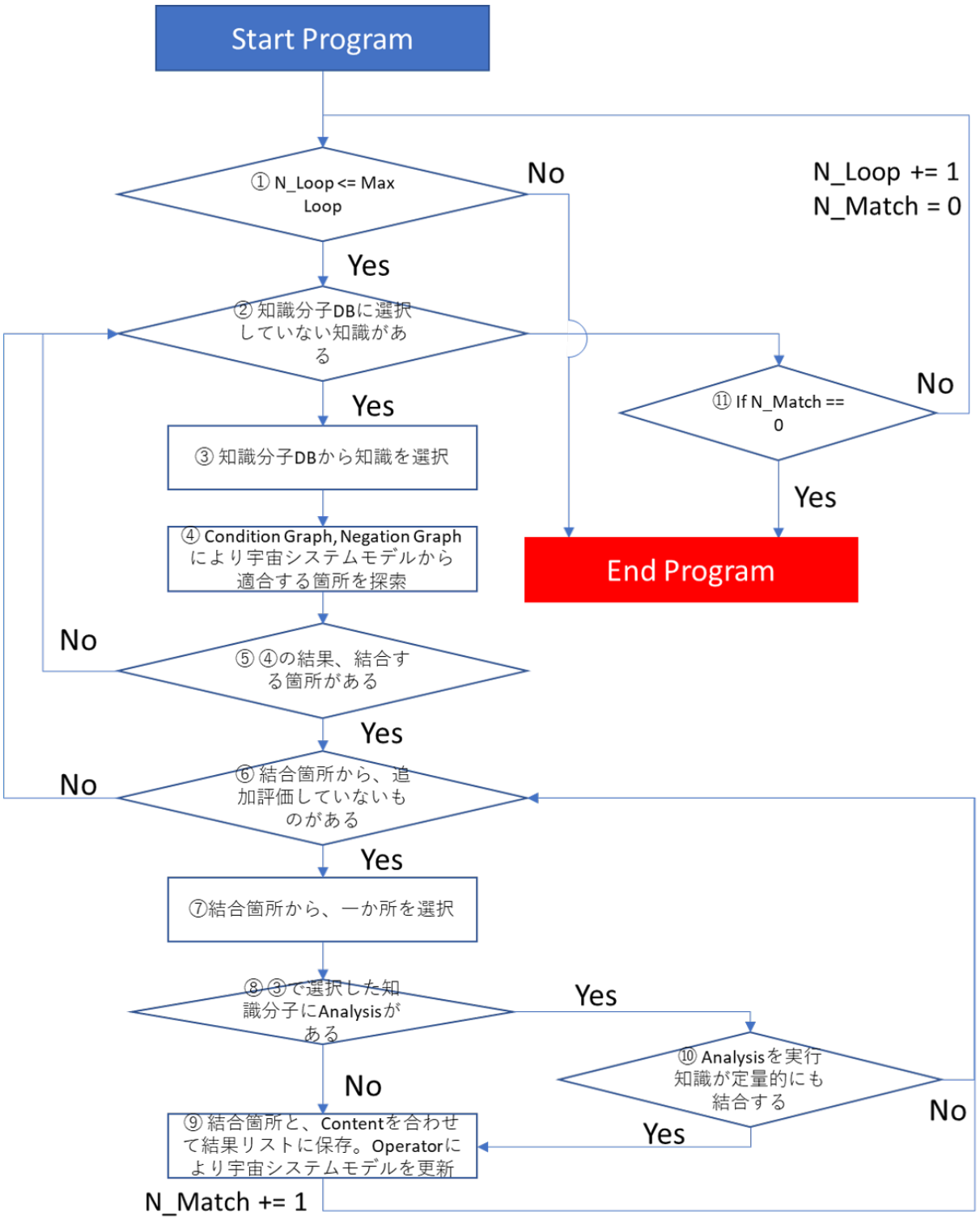


図 3.12　知識結合アルゴリズムフローチャート

義する。そして、知識の結合が起きない、もしくはループ以上の回数、繰り返し知識を結合させるまで、知識分子の結合を続ける。ここで、すべての知識に対して 1 回づつ結合の判定を行うことを、1 ループと定義する。

Operator の機能により、宇宙システムモデルの情報は、知識の結合のたびに更新され、前のループでは結合しなかった知識が結合する場合がある。(例えば宇宙システムにおける故障の波及効果推論などでは、ある知識分子による故障状態が Operator によりモデルに加えられ、それを Condition Graph や Analysis で判定する知識分子が新たに結合する)。そのため、ループを繰り返すことで、より充実した知識の結合を行うことが可能となる。一方で、推論が非常に膨大になってしまい人間が理解できる範囲を超えてしまうリスクがある。また Operator 同士が干渉し、無限ループが発生するリスクがあるため、ループの回数を事前に設定することを提案手法では行う。

本研究では、知識の結合条件を厳密に規定できない場合でも、知識の結合を実行できるよう、知識分子の結合判定において二つの手法を提案する。

3.4.2.1　SPARQL ベース手法

一つ目の方法は、Resource Description Framework (RDF) で定義されたグラフ構造データに対して問い合わせを行う SPARQL と呼ばれる言語を用いる方法である。RDF 形式のデータは、主語 (subject), 述語 (predicate), 目的語 (object) の 3 つの要素 (Triples) から成り立つデータを複数保持しており、主語、述語がノード、述語がエッジになることでグラフ構造を構築する [46]。SPARQL はこの RDF グラフのうち、条件に合った構造を（主語、述語、目的語の組み合わせ）を複数を瞬時に検索し、出力することができる [47]。また、その条件も、主語、述語、目的語の組み合わせ (Triples) で記述することができる。そのため、グラフ構造を持つ Condition Graph、あるいは Negation Graph を、SPARQL による問い合わせに変換することで、宇宙システムモデルから結合場所を検索することが可能である。また検索言語であるため、宇宙システムモデルを RDF ファイルに変換すれば、「結合が起きるかどうか」だけでなく、「どの箇所で結合が起きたか」を探索することが可能である。この検索方法は、計算コストがかからず実装もシンプルとなる。一方で、厳密に条件に合致したグラフのみを検索することから、グラフの記述が揺らいでいたりすると、知識を結合することができないというデメリットがある

3.4.2.2　類似度ベース手法

別の方法として、本研究では RDF グラフの類似度を計算する手法を用いる方法を本研究では、提案する。プロジェクトで入力された宇宙システムモデルの部分グラフ（全体モデルの一部分）と知識分子の類似度を測定し、適合の判断、適合箇所の特定を行う。類似度計算では、Zhang らが提案した手法を用いる [48]。この手法の特徴は以下の通りである。

- グラフを構成する各ノード、エッジが、オントロジーにおける概念の上位/下位の関係上距離が近いと、類似度が上昇する。
- 類似したグラフのエッジ同士の接合関係が近いと、さらに類似度が上昇する。(接合関係が異なると類似度は下がる)

これにより、例えば下記のように、知識の曖昧さを踏まえて適用範囲の判定が可能になる。

- RW-003(リアクションホイールの 1 つの型番) において、電流値の設計値のミスにより、起動時の過電流が許容量を超える不具合が発生する。
- 知識分子の条件としては「RW003 をグラフに含む」
- SPARQL ベースの結合方法では、別の RW type-A には知識は適合されないと判定される。一方で、同様の設計ミスは類似のコンポーネントでは発生すると考えられる。
- 類似度ベースの結合判定を用いることで、概念の関係上近い RW の別の型を含む場合でも、結合し、不具合リスクを提示できる。

なお、単純な類似度計算 [48] のみを用いると、類似度の測定結果、宇宙システムモデルの複数個所で知識が適合可能な場合、一か所のみがランダムに選択される。そのため、別の部分グラフへの結合を見逃す可能性がある。そこで、類似度を計算する際、探索され結合判定した類似する部分グラフを保存していき、二回目以降に類似度計算を行う際は、保存した部分グラフへの類似度評価は悪くなるように、類似度計算の重みを付けるようにした。これにより、複数回類似度計算を実施することで、同じ知識で宇宙システムモデルの複数個所への知識分子の結合を可能にした。

以上のように、本研究では二つの知識分子の結合方法をベースに、知識分子と宇宙システムモデルのマッチングを行い、知識の再利用を行う。なお、二つの手法は以下のようなメリットデメリットが存在し、知識の性質や知識を利用したい状況に合わせて、方法を切り替えることが望まれる（表 3.4)。

なお、提案手法の実装では、どちらの結合判定方法が望ましいかを、知識ごとに入力しておくことができるようにしている。知識の結合条件が厳密に判明しているかに応じて、この望ましい結合判定を指定することが可能である。(例えば、原因がはっきり判明していない不具合に関する知識は類似度ベースを指定する)。 知識利用の際は、この知識ごとの望ましい推論方法を利用することを基本とし、一方で全知識分子を SPARQL ベース、あるいは類似度ベースで推論するよう強制することも可能なようにしている。

また、本提案手法のアルゴリズの計算コストのオーダーは、$O(N \times M \times I)$ 程度となる。N はレポジトリに保存してある知識分子の種類の数、M はシステムモデルに含まれるエッジの数、I は推論の際のイタレーションの数である。レポジトリの知識分子一つづつにたいして、結合の判定をしていき、Iteration で結合が判定されなくなるまで結合の判定を行うので、その計算

表 3.4　Concent に書かれる内容の例

手法	特徴	使用場面
SPARQL ベース	厳密に条件に合致した所に結合できる 実装がシンプルで計算コストが軽い	知識が適合する条件が、物理的に 説明がつくなど、明確な場合
類似度ベース	厳密に条件が一致しなくても、 類似なら結合できる 計算コストがかかる	知識が結合する条件が 曖昧さを含む場合

コストは $N \times I\times$(1 つの知識分子結合の計算コスト) となる。Operator によりイタレーションが無限に続く場合計算コストは発散するが、そのような可能性を防止するため、事前に上限を設定する必要がある。また (1 つの知識分子結合の計算コスト) は、最悪ケースでシステムモデルのグラフ一つ一つに知識分子のエッジのマッチングを取っていくコストになるので、知識分子 Condition Graph の大きさ× システムモデルのサイズとなる。この時知識分子の Condition Graph の大きさは、システムモデルに対して小さいと考えられるため、オーダーとしてはシステムモデルのサイズ M が支配的となる。よって計算コストは $O(N \times M \times I)$ 程度である。そのため、知識数が増えたり、モデルのサイズが増えると、どうしても計算コストが増大してしまう。知識分子自体にラベル化を行い、必要なものを抽出してから推論を行うなどの工夫が、対策として考えられる。

3.5　提案手法 知識の評価

前述の知識分子とその結合アルゴリズムにより、保存した知識の結合を行い、エンジニアがプロジェクトに必要な過去の知識を再利用することが可能となる。この時、データベースに保存された全ての知識分子に対して、知識の結合判断が実施される。多くの知識が保存されている場合、多量の知識が結合してしまい、重要な知識に対するエンジニアの理解、判断が難しくなり、効率的な知識利用ができなくなる可能性がある。アルゴリズムにおける結合の上限回数を制限することで、結びつく知識の数は制限できるものの、それでは重要な知識が見逃されてしまうリスクは残る。

結合する知識分子全てが、それを提供されるエンジニアにとって、有用であるわけではない。例えば、エンジニアが電源回路に関して多くの知識を有している場合、認識している電源回路に関する知識を提供していても、認識の通りであることが確認できるだけであり、あまり価値を感じない。(ただし、このような場合でも、例えば電源回路周辺で、ほかの分野との interaction の関係で見逃している知識を提供した場合、それに対しては価値を感じる可能性はあると考えられる)。あまり価値のない知識があふれてしまった場合、エンジニアにとって、

重要な知識が見逃されてしまう可能性がある。例えば不具合リスクの知識を提供する場合、気づいたときに早めに対処すべき重要な不具合リスクが、すでに対処済みのリスクに関する知識に埋もれて見逃されてしまう危険がある。

このような問題に対して、本研究では、結合した知識の中から重要な知識を識別手法を提案する。これにより、効率的な知識再利用を目指す。エンジニアにとって重要な知識を識別できれば、エンジニアが見逃してしまう可能性は少なくなると考えられる。

3.5.1 関連研究

Data Mining の研究では、大量のデータから重要な特徴を、傾向 (= 知識) を特定するマイニングを実施する。その際、大量の知識が発見されてしまう可能性があるため、それらの重要度を識別する方法が研究されている。指標は客観的な知識と、主観的な知識の二つの知識の分野がある [49]。客観的な指標とは、マイニングされた知識の数や、頻度などの指標がある。しかしながら、それらは多数の人間に共通して価値となる可能性があるものの、個人に対する価値は保証されていない。一方で主観的な知識の指標は、ユーザーにとって有難いかどうかという指標であるため、ダイレクトにユーザーへの価値に繋がる。

主観的な指標の例として、Actionability, Unexpectedness という指標が Data Mining の分野では広く使われている [50]。Actionability とは、知識をもとに人が何かアクションを取れるかどうかという指標である。一方で、Unexpectedness とは、人が予測した知識、あるいはすでにある知識と、発生した知識がどれくらい異なっているかという指標である。Actionability は、実際に知識を用いて現実を改善する可能性が高いという点、Unexpectedness は人のバイアスを修正するという点で、ユーザーにとって価値となる。

3.5.2 提案手法

本研究では、このような Data Mining をもとに、Actionability, Unexpectedness という二つの指標を、知識結合結果に付与する方法を提案する。さらに独自に、Interest level と呼ばれる指標を追加し、ユーザーがほしい知識の分野、観点にどれだけ近い知識かを評価できるようにする。 いかに知識の評価手法を定義する。

3.5.2.1 Actionability

前述のとおり、Actionability とは、知識をもとに人が何かアクションを取れるかどうかという指標である。本研究では、宇宙システムを開発しているエンジニアに知識を提供することを想定しているため、その文脈に適した指標の計算方法を検討した。まず、本研究では、提案手法の利用者は宇宙システム開発を行っているエンジニアであるため、Actionability とは「開発

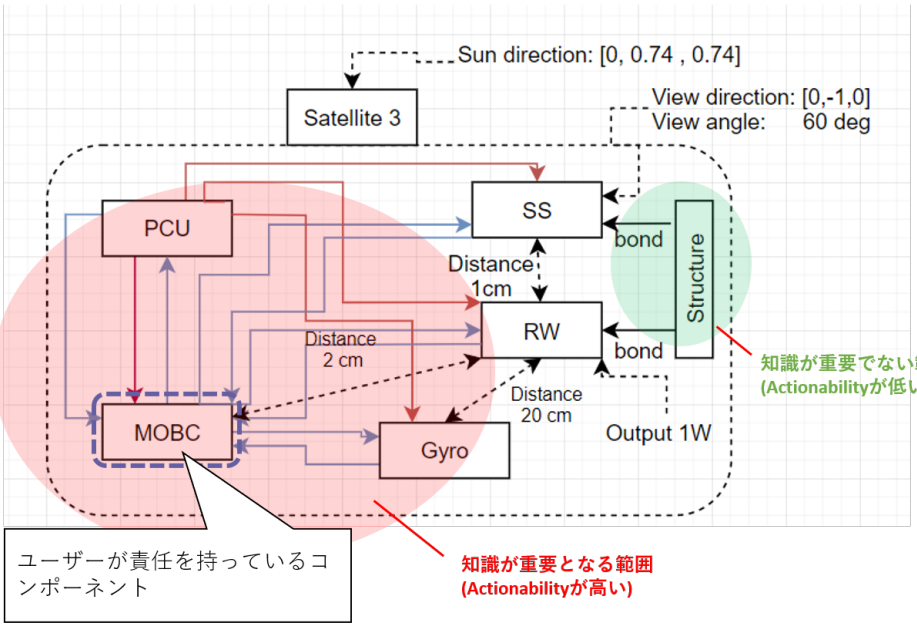


図 3.13　Actionability 評価のイメージ。システムブロック図において、ユーザーが責任を持っているコンポ、あるいはそのインターフェース周辺に結び付く知識は重要となる

するエンジニアにとって実行可能な知識かどうか」、という評価に置き換えられる。この開発エンジニアの実効性は、以下の要素で評価できると考えられる。

- 知識に定義されているアクションにかかるコスト、工数が、そのプロジェクトのリソースにマッチしているかどうか
- 知識に定義されているアクションが、エンジニアの開発責任範囲かどうか

前者に関しては、実行する宇宙システム開発プロジェクトへの依存が強く、プロジェクトのリソースの入力が必要となるとともに、そのプロジェクトにおけるアクションがかかるコスト・工数を見積もる必要がある。そのため、利用時にユーザーへの記述の負担が大きくなる可能性が高い

後者について、宇宙システム開発プロジェクトでは、エンジニアはシステムを構築するサブシステム、もしくはコンポーネントによってグループ分けされ、その範囲に責任を持って開発を行うことが一般的である。そのため、エンジニアは責任を持つコンポーネント、サブシステム、そしてその周辺 (他のコンポーネント、サブシステムとの IF) にかかわる不具合リスク、検証項目、設計に注視する必要がある。すなわちユーザーの責任をもつコンポーネント群に対し、そこにどれくらい近いかが重要となる (図 3.13)。知識が設計のどこに結びつくかは前述の知識結合のアルゴリズムで取得できるため、ユーザーの責任を持つコンポーネント群を入力することで、この「近さ」は簡単に計算を行うことが可能である。

以上のような検討の結果、本研究では、ユーザーへの負荷を抑えて簡単に利用できるよう、

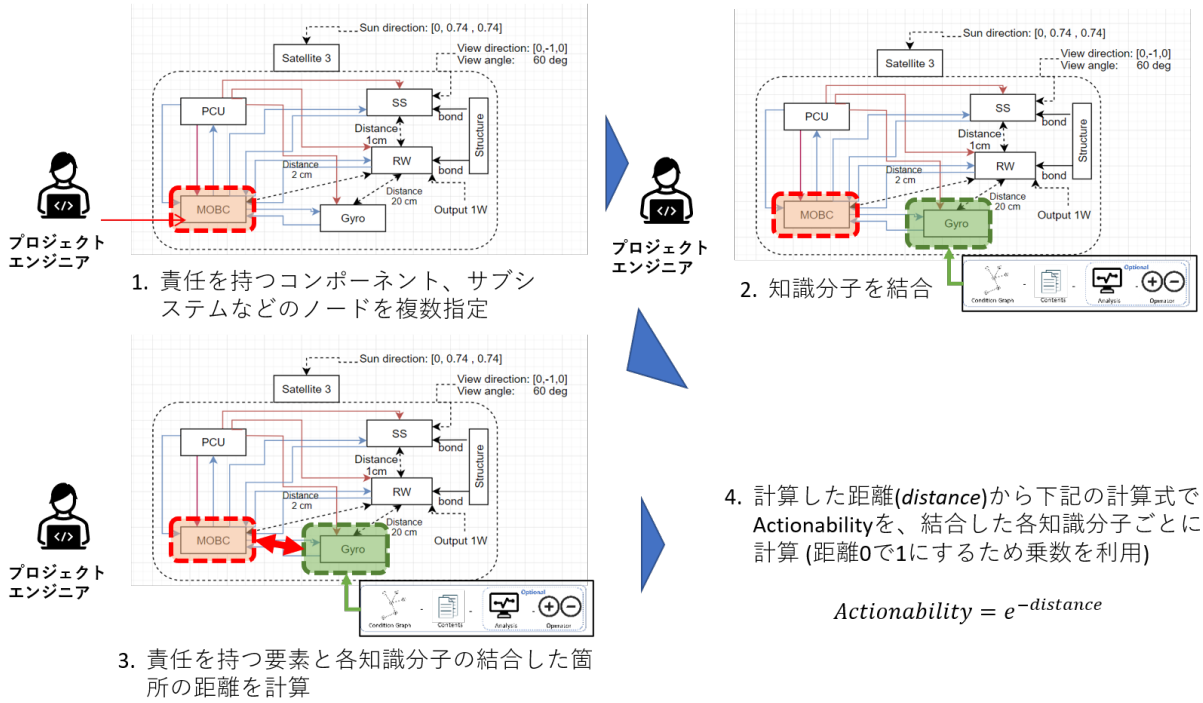


図 3.14　Actionability の評価手順

以下のように Actionability を測定する方法を提案する。

測定方法

本研究では Actionability は以下のように測定する。

1. ユーザーの責任範囲となるコンポーネントを、システムグラフの構成要素から複数入力する
2. 知識分子を結合する
3. 知識の結合場所を取り出す。ユーザーの責任範囲の各要素と、知識結合箇所のグラフ上の最短距離をダイクストラ法で計算する。
4. 図 3.14 4 の計算式で、最短距離から Actionability を計算する。Actionability を知識分子に付与する

3.5.2.2　Unexpectedness

前述の通り、Unexpectedness とは、人が予測した知識、あるいはすでにある知識と、発生した知識がどれくらい異なっているかという指標である。宇宙システム開発においても、エンジニアが持っていない、あるいは見落としている知識を提供することは、エンジニアにとって有

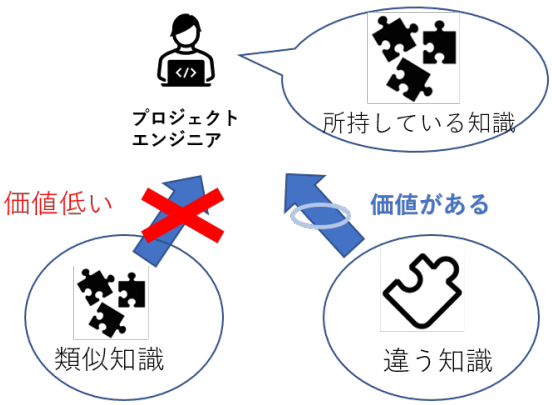


図 3.15　Unexpectedness のイメージ

意義である。逆に、すでに持っている知識に関しては、その価値は下がると考えられる。知識分子を利用る場合、知識はその記述方法に自由度があるため、その自由度も踏まえてユーザーエンジニアが持つ知識に類似する知識は、ユーザーにとっての価値は下がる可能性がある。一方で、ユーザーが認知している知識とは大きくなる知識は、ユーザーが見逃している観点や知識の分野の可能性が高いと考えられる (図 3.15)。

本研究では、この考えに基づき、ユーザーにすでに認知している知識を入力してもらい、それ対して結びついた知識分子のグラフ構造の類似度を計算することで、この値を計算する。グラフ構造の類似度は、RDF グラフの類似度計算を使用する [48]。

測定方法

本研究では Unexpectedness は以下のように測定する。

1. ユーザーの認知している知識を、知識分子 DB の中から複数選択する
2. 知識分子を結合する
3. 1 で選択した知識と 2 で結合した知識分子の類似度を計算する。
4. 図 3.16 の 4 の計算式で、類似度から Unexpectedness を計算する。Unexpectedness を知識分子に付与する。

3.5.2.3　Interest Level

宇宙システム開発において、エンジニアのレビューは、効率化のために、レビューをする範囲や観点を絞り、その範囲や観点にふさわしい専門家を招集することが有効である [51]。特に、必要な観点を整理し、プロジェクトのエンジニアがリスクが高いと感じる観点に、豊富な知識を持った専門家を招集することが重要である。これにより、エンジニアの不足した知識を、効率的に収集することが可能となる。このレビュープロセスにおける有効手段を、知識分

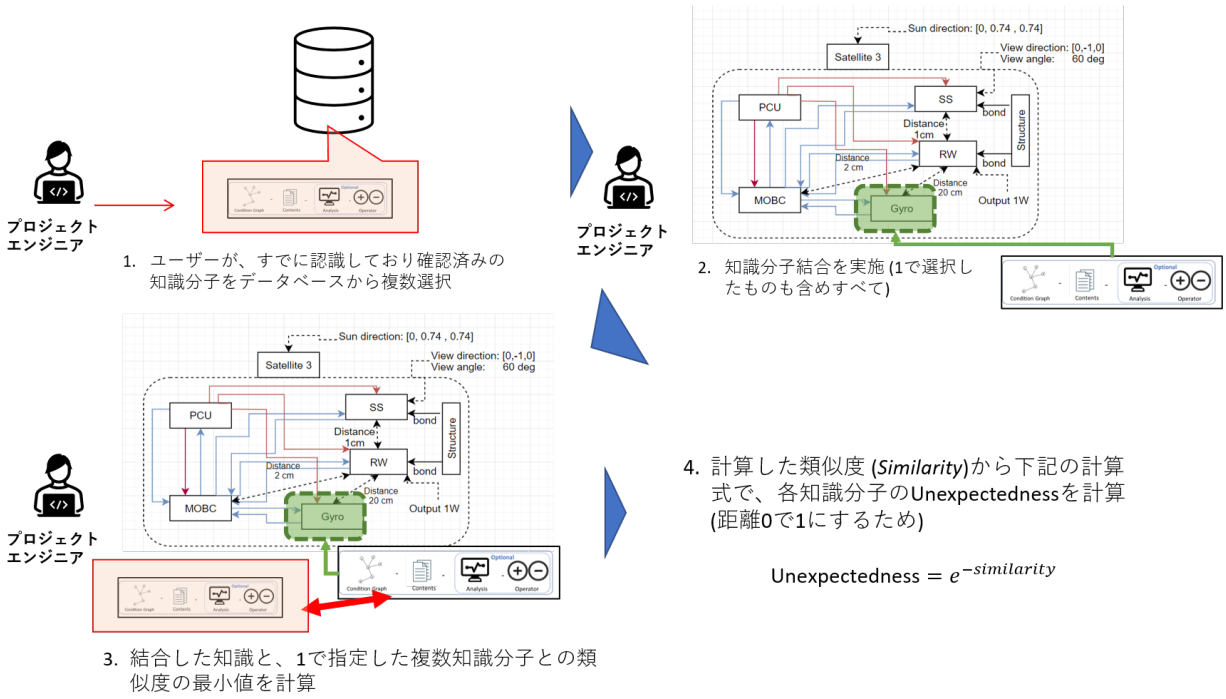


図 3.16　Unexpectedness の評価手順

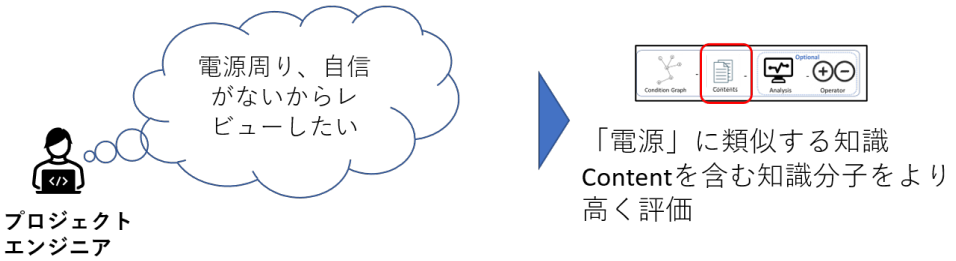


図 3.17　Interest Level のイメージ

子に適用することを考え、本研究では Interest Level という指標を定義した (図 3.17)。

Interest Level はユーザーが興味を持つ、不足しているという観点を単語や文章で入力し、それに近い概念を Content に含む知識分子に高い評価を与えるという指標である。本研究では、自然言語処理で反転している Word2Vec [52] および Word Mover's Distance [53] を利用し、ユーザーの入力した観点と、知識分子に入力された Content の距離を計算することで、この値を計算する。詳細は次のとおりである。

測定方法

本研究では Interest Level は以下のように測定する

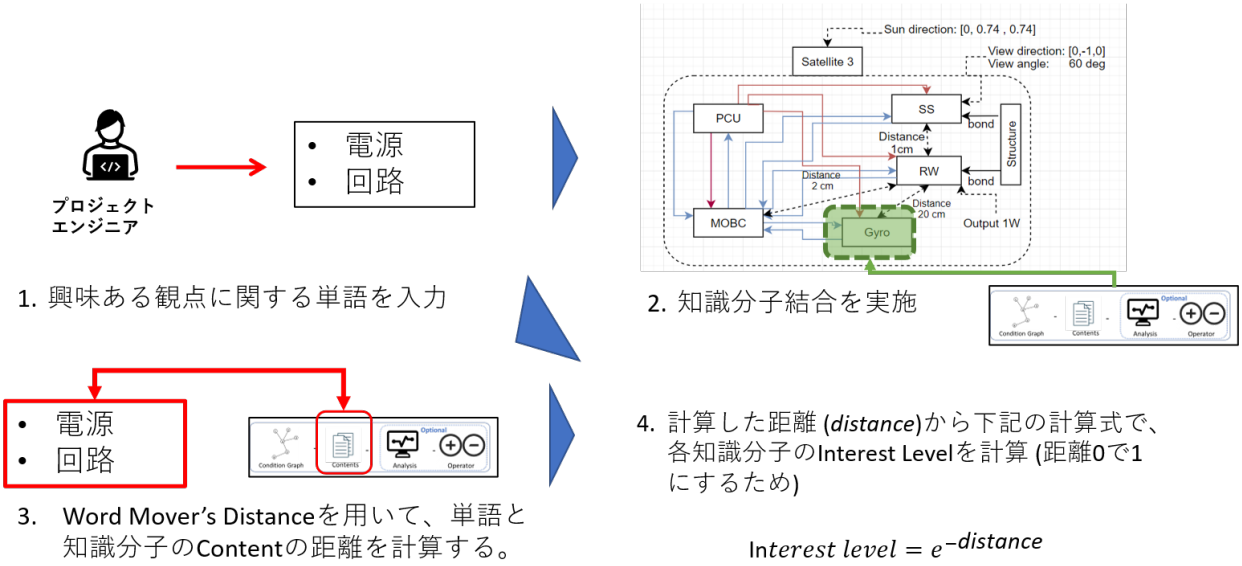


図 3.18　Interest Level の評価手順

1. ユーザーが興味ある観点、概念の単語を複数入力する
2. 知識の結合を実施する。
3. 入力した単語と選択した知識の Content の距離を計算する
4. 距離をもとに、図 3.18 の 4 の計算式で、Interested Level とする。評価を知識分子に付与する

3.6　実際の運用

本章で提案した知識評価は、実際に知識分子を結合する際、必ず実施する必要はなく、知識分子数が多い場合や、網羅的な確認よりも効率的なレビューが必要な場合に有効になると考えられる。また、知識の重要度評価指標を複数組み合わせて総合的に重要な知識を判断したいケースも存在する。そこで本研究では、後述の知識分子活用のツールにおいて、知識評価の有効無効か、有効の場合、各評価指標の有効化、無効化を指定できるようにした。なお、知識分子の評価は、知識の結合後に同時に実施するようにした。

3.6.1　提案手法を利用できるツールの実装

知識保存のフレームワークとしての知識分子、および結合アルゴリズムを用いることで、プロジェクト間の知識再利用を実現可能となる。一方で、実際にプロジェクトで、効率的にこの手法を活用するためには、多くのユーザーが利用しやすいインターフェース、データファイル

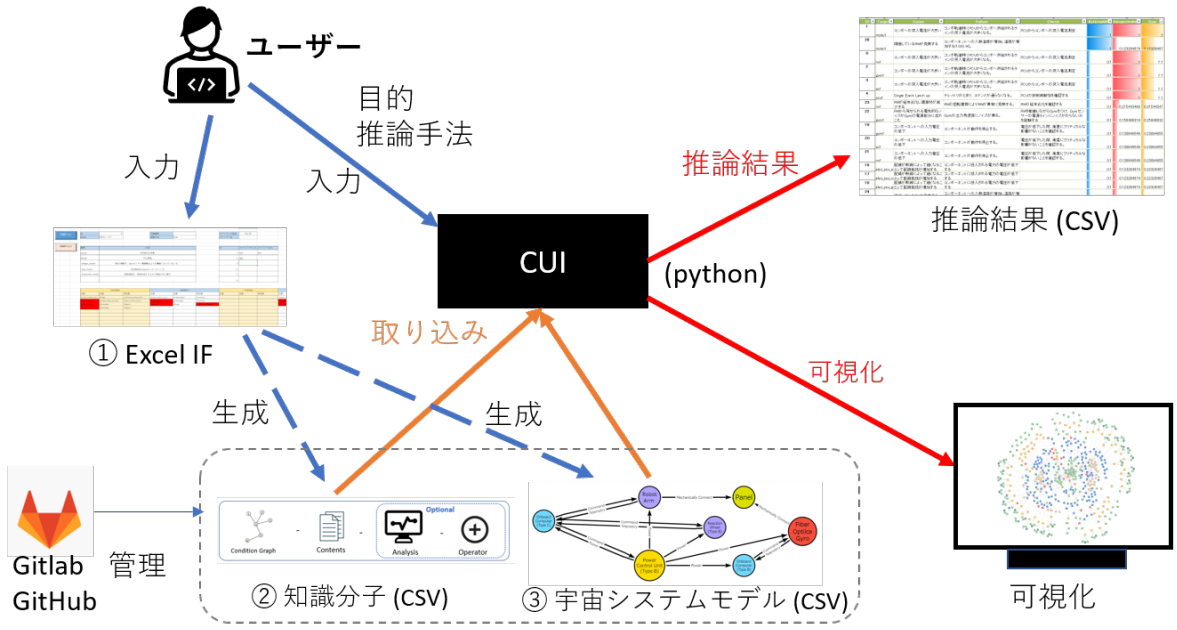


図 3.19　実装したツールの全体像

を整備する必要がある。そこで、本研究では、実際にそのようなインターフェース、データファイルを、エクセルなど利用者の多い汎用的なツールを用いて実装した。

ツールの実装の際には以下の要件を考慮した。

1. OSS をベースに誰もが利用でき、ツールを改良できる
2. 知識の増加や宇宙システムモデルは、追加、改良しやすいようバージョン管理ができる

上記の要件を考慮し、結果として図 3.19 のようなツールを実装した。

まず、知識分子と宇宙システムモデルについては、csv をマスターデータとする。csv はテキストデータなのでバージョン管理ツール (git) での管理が容易であり、変更履歴の保存や、変更時の意思決定、レビューを実施することが可能である。csv では、知識分子やシステムモデルの記述は直観的に行うことが難しいため、マクロにより Excel で読み込み記述をサポートする。これにより、誤ったオントロジーに基づく記述をエラーでユーザーに知らせたり、ソートによる整理を容易にすることが可能である (図 3.20, 3.21)。そして、Character User Interface (CUI) でユーザーからの目的、推論方法の入力に従い、知識を宇宙システムモデルに対して結合できるようにしている。

その際、宇宙システムモデルは SPARQL に基づき推論できるように一度 RDF ファイルに変換される。知識の結合を行う CUI への入力としては、以下のようなパラメータがある。

- 知識結合の繰り返し回数の上限

図 3.20 実装した知識編集の Excel インターフェース。知識 Content の種類、内容、Condition Graph, Analsis の Input Output を直観的に入力できる。オントロジーにマッチしていなかったり、記述に一貫性がない場合は色を変えて警告が出される。なお、Analysis のスクリプトは別途プログラムファイルで編集を行う)

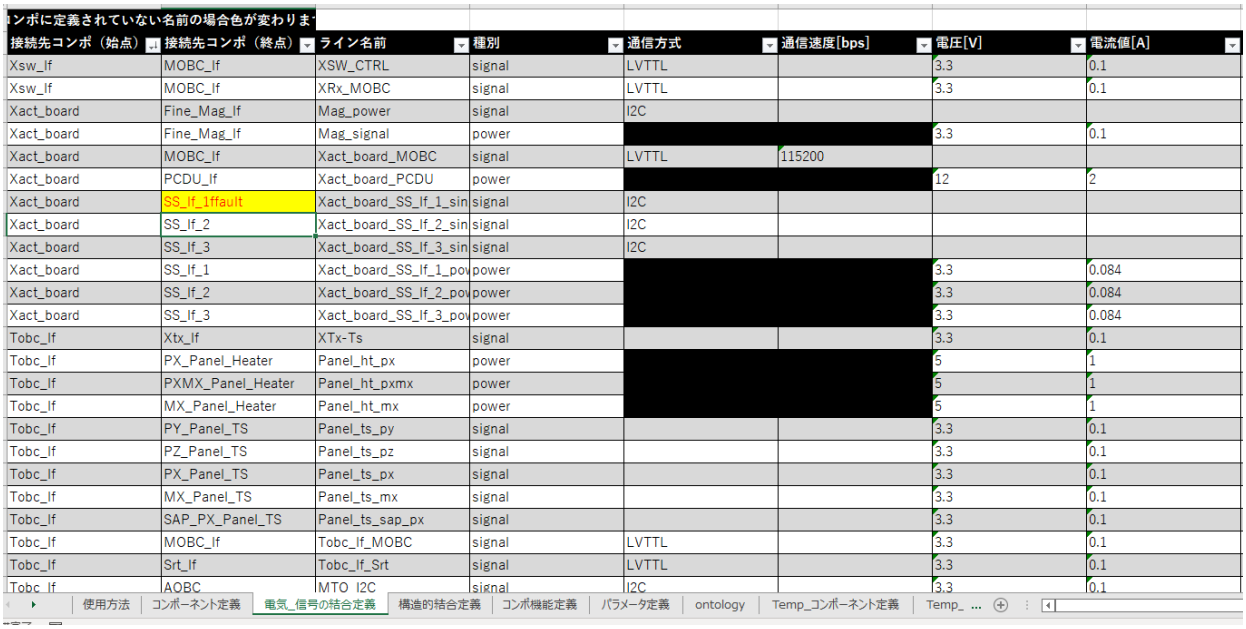


図 3.21 実装した宇宙システムモデル編集の Excel インターフェース。システムの要素となる部品、モジュールを定義したり、その包含関係、電気的、構造的依存関係、機能とパラメータを定義できる。記述に一貫性がない場合は色を変えて警告が出される。

- 利用する知識分子の種類 (FMEA（故障リスク、対策）, 設計モデル補完、Verification)
- 知識の結合を判断する際の結合アルゴリズム (Sparql ベース、類似度ベース)
- 知識評価の実施の有無とその引数 (既に所有している知識、ユーザーの責任範囲、興味を持っている概念）

これらは設定用の json ファイルで設定できるようになっている。

また、補足機能として、宇宙システムモデルのグラフとしての可視化や、付録 C に示す自然言語処理による知識分子の自動定義を実行することが可能である。

3.6.2 再利用性の高い知識分子の定義方法

本章では、プロジェクトで知識を登録する際のプロセス、特に不具合が発生した際の、不具合に関する知識を保存するプロセスについて説明する。まず、知識登録の全体概要を図 3.6.2 に示す。前述のツールに加えて、ここで述べるプロセスに従って知識を保存することで、確実かつ効率的な知識運用が可能となる。

1. 原因説明

まず不具合が発生した際は、その不具合の原因を、物理および機能の粒度で細かく説明することを試みる。不具合は、波及効果を踏まえて複数の要因がつながることで発生することが多いので、その場合は、根本的な原因（ヒューマンエラー、経年劣化、外部からの予想外な入力、設計値のミス）まで明確にする必要がある。ここでいう「物理および機能の粒度」とは、

- 衛星システムを構築するコンポーネントや部品、設計の「種類」

ではなく、以下のような要素を用いた説明である。

- コンポーネントがどういう材質、特性か (生存温度範囲、大きさ、重量)
- どのような物理量 (熱、電気、力) をやり取りしているか
- コンポーネントが持つ機能は何か (例えば、電源検知、角速度検知、トルク生成)

このような粒度で説明しておくことで、他分野でも共通する機能、特性を持つシステムに知識を適用することができ、再利用性の高い知識として保存することが可能である。ただし、「インターフェース設計に対するエンジニア同士の認識齟齬による不具合知識」など、ヒューマンエラーが原因の場合は、物理的な粒度での説明は不可能であるため、そのままコンポーネント名や部品の「種類」に対して、発生したと説明することを許容する必要がある。また、物理的な原因が、そのプロジェクトのリソース的に究明することが難しい場合は、コンポーネントの種類のみを記録し、次のステップへと進む

2. 知識分子の定義

1. で行った不具合の説明を元に知識分子を定義していく。その際、不具合の連鎖に関する説明を一つ一つ、原因、結果に小分けにしていき、それらのペアごとに知識分子を定義する。原因は、宇宙機システムの設計に由来する部分は Condition Graph として条件を保存しつつ、状態が異常の場合は Analysis としてパラメータやステータスの値を条件として保存する。物理的な原因が明らかな場合は、「コンポーネントの種類」を Condition Graph から排除し、入出力物の関係、機能のみの要素を Condition Graph に入れる。物理的な原因が明らか出ない場合は、3 で述べる。また結果として引き起こされる状態は Operator として保存していく。細かく原因と結果に分けることで、知識の再利用性が高まるとともに、複数の知識を組み合わせて人間が予想できない推論を実施することが可能になる。

3. 原因が明らかでない不具合知識の定義

全ての物理的原因が明らかである場合、2 のように知識を保存する。一方で、ヒューマンエラーや原因不明の不具合の場合は、物理的な要因を説明できない。その場合、厳密に知識の利用範囲を記述することは難しい。この場合、Condition graph は下記のように記述する。

- 不具合が発生したコンポーネント、部品の種類を可能な限り具体的に指定する (例えば Gyro センサータイプ A で不具合が発生した場合、Gyro センサーを条件にするのではなく、タイプ A を条件とする)。そして、結合条件を類似度として指定する。

このように知識の範囲をできるだけ具体化することで、過剰に知識が別の設計に結合することを防ぐことができる。一方で、知識を推論する際に、類似度による推論を用いることで、類似の箇所に不具合が発生するリスクを予見することに用いることが可能である。このように、推論の際に工夫をすることで、知識の結合が操作できるため、再利用性が向上する。

4. イタレーション

1 - 3 のプロセスを繰り返す。csv で知識を記述したら、git への commit,　必要であれば、既存知識へのマージ、それに対する レビューを実施する。

以上のような方法で知識を推論することで、知識の再利用性、利用時の効率性を保つことが可能である。そのほかのパターンも含めて、表としてまとめたものを表 3.5 に示す。

この表に従いすべてが場合分けができることが理想である。しかしながら、どのように不具合の原因を説明するかは厳格に規定しきれないため、最終的には人間の管理、判断が必要となる場合もある。時にはあるエンジニアが行った不具合の原因説明が十分でなく、知識として誤ってしまうリスクもある。

これに対して、前述のように Git を用いたバージョン管理、Github, Gitlab といったマネジ

表 3.5 不具合の原因と Condition Graph の記述方針

ID	パターン	生成する知識分子の Condition Graph 記述方法
1	不具合 F の原因が、コンポ type A の持つ物理的性質、機能 B で説明できる場合	Condition Graph は「Compo has-Function B 」として記述し、type A を条件として記述しない。
2	不具合 F の根本原因は不明で、コンポ type A がある条件 B になると、意図しない状態になる場合	Condition Graph は「Compo is-a type A, Compo has-status B 」として記述し、type A を条件として記述 (Status は Analysis での定量的条件としても OK)
3	不具合 F の原因が、コンポ type A の特殊な機能、性質周りの設計ミスの場合	Condition Graph は 「Compo is-a type A」とする
4	不具合 F の原因が、コンポ type A の持つ一般的な機能、性質 B に関連して起きたヒューマンエラーの場合	Condition Graph は「Compo has-Function B」として記述し、type A を条件として記述しない
5	コンポの種類に一般的にみられる性質、設計による正常な動作が原因の場合	Condition Graph は「Compo is-a type A 」または「Compo subclassOf type A」として記述

メントツールを利用したレビューシステムを導入することで、誤った知識と、誤った知識記述方法を減らすことが重要であると考える。このようなツールを利用することで、不具合原因の考え方、整理の仕方、知識の記述方法のノウハウが蓄積し、幅広い範囲の人間に共有されるシステムを構築することが可能である。逆に言えば、本手法では知識の記述というノウハウは必要となってしまう。しかしながら、そのノウハウは個別の宇宙システムの不具合知識に対してメタなレベルであり、宇宙システムの不具合そのものに関する多様なノウハウを蓄積するよりは、はるかに容易に獲得できると考えられる。

(あるいは、Appendix C に示すような、自然言語処理を用いて知識分子の生成を自動化を行うことで、知識の信頼性の高い再利用は難しいものの、人間の属人性が入る余地をなくし効率的に知識分子を定義していくということも選択肢の一つであると考えられる。)

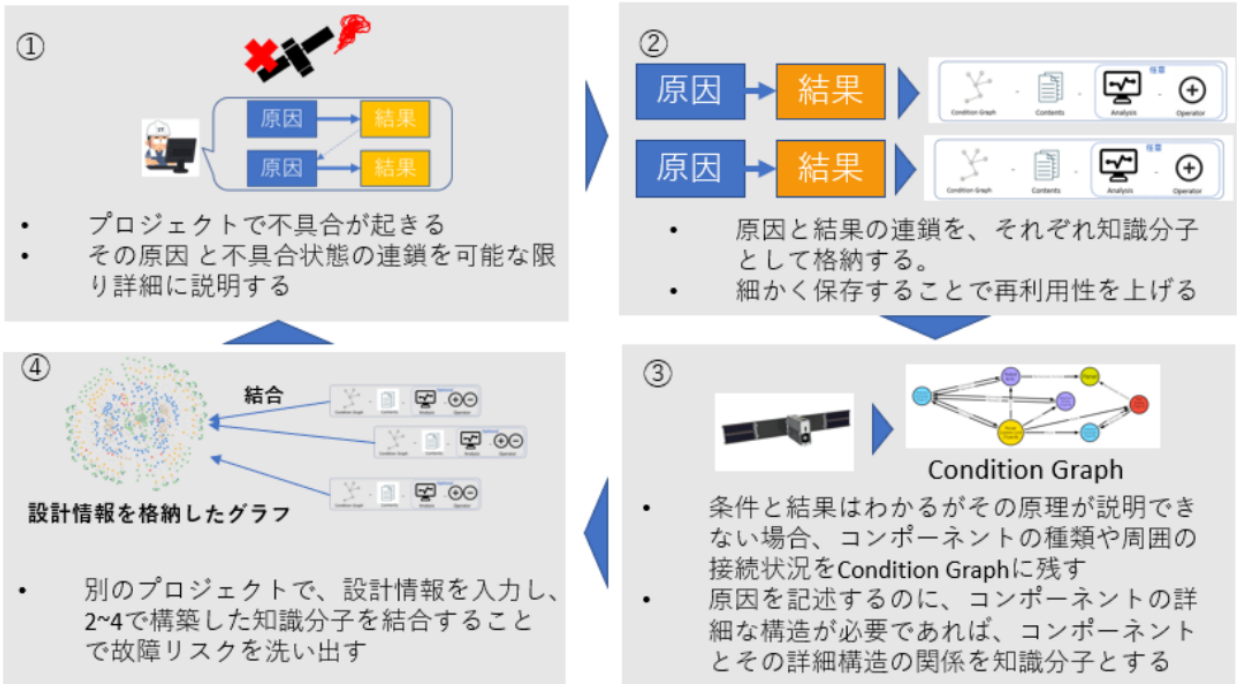


図 3.22　知識分子の獲得、再利用のサイクル

第 4 章

提案手法の検証

本章では、提案手法の有効性を示すために行った検証実験とその結果について説明する。

4.1　検証 1 模擬衛星

本章では、簡易的な衛星開発プロジェクトを想定し、模擬的な衛星モデルの構築を行い、それに対して提案手法の適用を行う。これにより、本研究の提案手法が、異なるアーキテクチャを持つ衛星プロジェクトでも、そのコンテキストに合わせて自動的に知識が再利用でき、不具合リスク等を指摘できることを示す。衛星モデルは、過去のプロジェクトの衛星モデル、現在の衛星プロジェクトの二つを用意し、異なるアーキテクチャとし、過去のプロジェクトで発生した「故障リスク」に関する知識がどのように再利用されるかを示す。

図 4.1, 4.2 に、過去の衛星開発プロジェクトにおけるモデル、および現在の衛星開発プロジェクトにおけるモデルの二つを示す。それぞれ、3.3 章で示すオントロジーに基づき記述されており、構成している要素 (コンポーネント) とそれらの接続方法は異なる。

4.1.1　知識想定をする過去のプロジェクト

過去の衛星でプロジェクトでは、下記のような不具合が発生したとする。

1. 低温時、衛星姿勢制御用の Reaction Wheel (RW) Type A の電源を ON した際に、Power Control Unit (PCU) Type B の過電流シャットダウン機能により電源 ON されなくなる。

2. RW の機械的振動が Gyro センサーに伝搬し、ノイズ源となり Gyro センサーによる正確な角速度の測定ができなくなる。

上記の不具合は、軌道上で衛星が正常な動作をできなくなり、ミッションの失敗につながる危

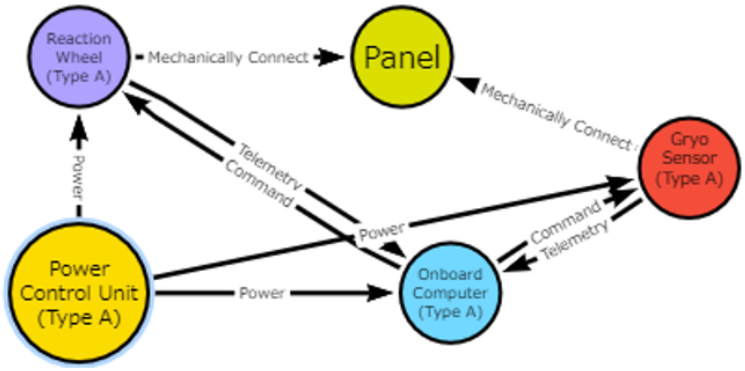


図 4.1　模擬衛星 1 (過去のプロジェクト)

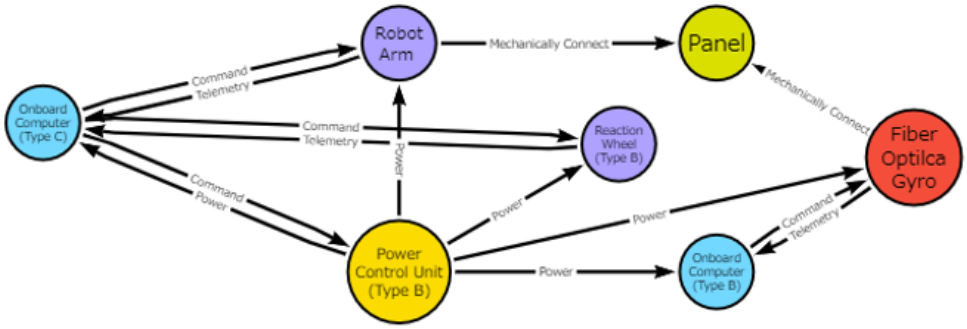


図 4.2　模擬衛星 2 (現在のプロジェクト)

険性がある。そのため、次のプロジェクトではこのような失敗が起きないよう設計上の工夫をしておく必要がある。従来、このような失敗知識は、不具合データベースに保存されるが、次ミッションの時に、その不具合が検索されない可能性がある。また、人がこの失敗を記憶していても、実際の設計時には忘れてしまったり、担当者が変わり同じ轍を踏む可能性がある。これに対して、本研究ではこの知識を知識分子に保存し、計算機によって知識を次プロジェクトのエンジニアに伝達することで、知識の抜け漏れを防ぐ。

4.1.2　知識の分子化

本章では、過去のプロジェクトで発生した不具合 (不具合 1, 2) を、知識分子に落とし込むプロセスについて説明する。

4.1.2.1　不具合 1

不具合 1 (RW の過電流シャットダウン) の原因は以下の形で原因が究明されたとする。

1. RW Type A の過電流は低温で大きくなり、5A になる。
2. RW の過電流値が、Power Control Unit の過電流閾値を超える。

また、この不具合から見いだされる教訓としては、それぞれ、

1. RW type A の過電流値を温度変化した際の値も確認しておくこと
2. 過電流閾値が、過電流を容易に上回る値になっていないかを確認すること

があげられる。

原因 1 とその教訓、原因 2 とその教訓を分けてそれらを知識分子に落とし込む。原因 1 は、RW type A 固有の話であり、他の RW でも同様の傾向があるかは判別することができるほど、物理的な原因が明らかになっていない。そのため、知識分子に落とし込む際は、RW type A を要素として入れ込む。状況としては、RW に入力されている電源に関する不具合であり、RW type A の温度が関係するため、それらの存在も知識分子の Condition に入れ込む。これを Condition Graph にすると、図の通りになる。また、Content に原因の内容、教訓を入れこむ。一方で、原因 2 は、過電流機能を持つ機器から電源を供給される場合に、普遍的に起こりうる不具合である。そのため、知識分子に落とし込む際は、コンポの種類は入れず、機能として過電流シャットダウンの機能を持ち、二つのコンポで電流のやり取りをしていることを Condition グラフにする。結果を表 4.1 に示す

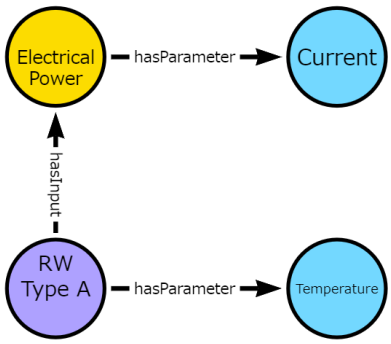
4.1.2.2 不具合 2

不具合 2 (Gyro の振動ノイズ) は下記の原因が究明されたとする。

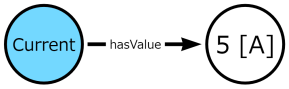
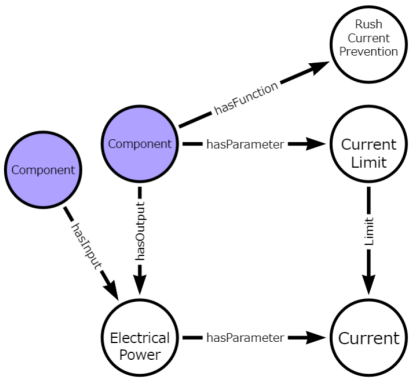
1. RW が機械的な振動を発生させた
2. 振動が、結合する構造物を伝搬した
3. 構造振動が、Gyro センサーに拾われた

これらは、それぞれ知識分子に落とし込むことが可能である。原因 1 は、内部に動くような部品全般に当てはまることから、内部に動く部品を持つコンポ全般に当てはまるとして、知識分子化できる。原因 2 も同様で、コンポが構造的に接合している限り、振動が伝搬するのは物理的に正しいため、一般化した形で知識を保存できる。最後の要因に関しても、構造振動が、角速度を取得する機能を持つ機器一般に当てはまるため、それを Condition グラフとして保存できる。教訓も合わせると、知識分子は表 4.2 のようになる。

表 4.1　不具合 1 に関する知識分子

<table>
<tr>
<td>1</td>
<td> 
</td>
<td>Content
故障: RW TypeA の温度が低いとき、電流値が 5A 程度まで行く

Analysis
Status(Temperature) == -1
(Temperature が正常時より値が低い)

Operator
 
</td>
</tr>
<tr>
<td>2</td>
<td> 
</td>
<td>Content
故障:過電流シャットダウンにかかる
設計確認: 設計上必ず閾値が大きいかを確認する

Analysis
Value(Current) ＞ Value(Current Limit)
(電流値が閾値より大きい)

Operator
なし</td>
</tr>
</table>

4.1.3　知識の適用

定義した知識分子を、今度は新しいプロジェクト（図 4.2) に再利用し、そのプロジェクトの設計において気を付けるべき不具合リスクを抽出する。実装に関しては下記のように実装した。

- オントロジーの記述: Ontology Web Language (OWL) をもとに記述
- プログラムの実行: Python
- 使用モジュール : RDFLib, Owlready

知識の結びつきは RDFLib モジュールにある関数を用いて、SPAQLE ベースの方法で行った。また、衛星のモデルに関しては、図 4.2 に現れている要素に加え、基本的なパラメータ (電流、温度、電圧、形状)、内部の部品の存在 (Robot Arm が内部に動く部品を持つ) を定義した。さらに、パラメータの異常、正常に関しては、-1 (正常値以下), 0 (正常値), 1 (正常値以上、もしくは単に異常) として表し Operator の効果を表した。

4.1.4 結果

知識を結合した結果を表 4.3 に示す。いくつかの知識分子が、入力した設計情報に結合し、故障リスクと設計における確認事項を提示している。まず、RW, Robot arm において、過電流シャットダウンの知識が指摘されている。これは入力した電流値や、過電流の閾値から判定された結果である。また、「動く部品を持つコンポが振動を生成する」という知識が、RW だけでなく、Robot Arm に結合し、その故障リスクが提示されている。これは知識分子生成の際に、故障の条件を一般化し、物理的および機能的な条件に落とし込んため、リスクが提示されたと考えられる。実際、このようなリスクは存在すると考えられるため、一般化が正しいことがわかる。また、FOG にノイズ出力のリスクが想定されているのも、同様に機能的な条件に不具合の条件を落とし込んだため発見されたリスクである。こちらも、想定されるリスクであり、正しい一般化がなされているといえる。最後に、宇宙システムグラフに知識分子が結合する様子を図 4.3 に示す。知識分子が、適切な箇所に化学分子のように結合している様子がわかる。以上のように、本研究の提案手法が、異なるアーキテクチャを持つ衛星プロジェクトでも、そのコンテキストに合わせて自動的に知識が再利用可能が示された。

4.2 検証 2 実衛星

本章では提案手法のケーススタディとして、実プロジェクトでの開発実績を踏まえたケーススタディを実施する。これにより、実衛星の規模で提案手法が活用できること、大量の知識が結びつく際に、3.5 章で提案した知識評価手法が活用できること、複数の知識を組み合わせた効果的な知識利用が可能なことを示す。本ケーススタディでは、東京大学中須賀船瀬研が開発した 6U 深宇宙探査機 EQUULEUS の開発中に発生した不具合事象に関して、知識分子を構築し、その不具合に関する知識 (原因、結果、対策) を、新しい 6U プロジェクト (以下、ISSL6U) に再利用することを行う。知識の再利用により、ISSL6U の設計時に、不具合リスクを事前に防止することを想定する。

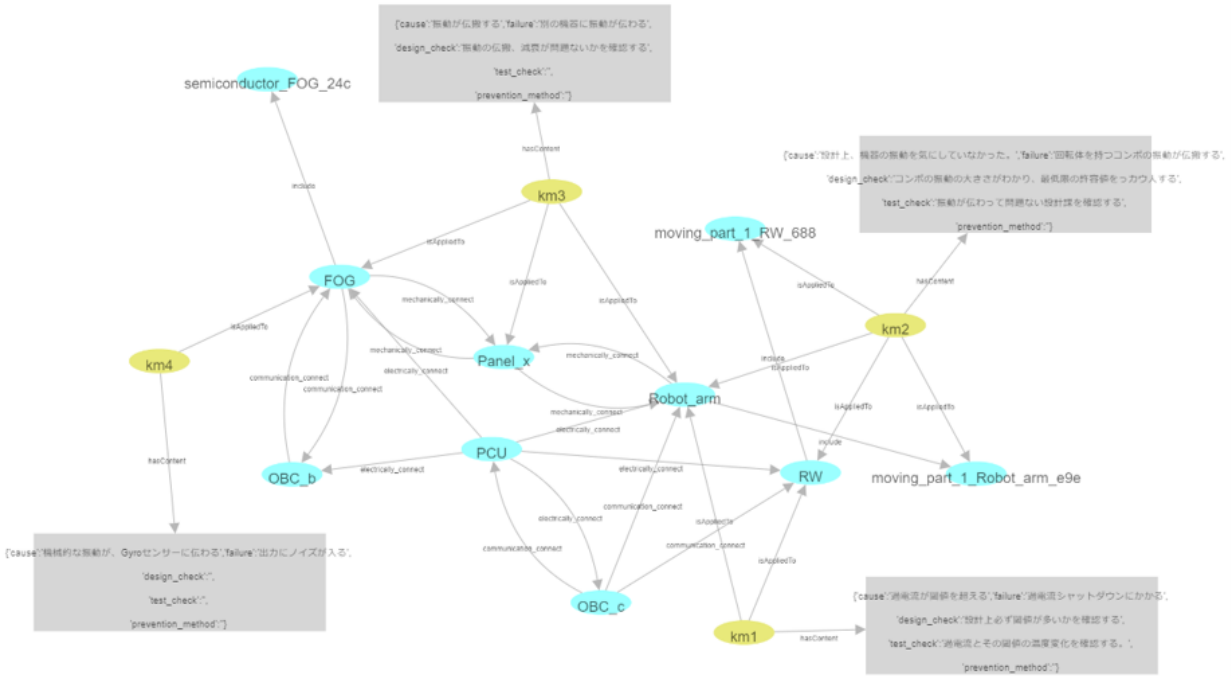


図 4.3　模擬衛星 2 に知識分子が結合する様子

4.2.1　衛星の概要

EQUULEUS は、2022 年 NASA の新型ロケット SLS によって打ち上げ予定の 6U CubeSat(図 4.4) である。 EQUULEUS には、水を用いた推進系が搭載されており、それを用いて地球から見て月の裏側に位置するラグランジュ点 (L2) を目指す。ラグランジュ点では、月、地球、太陽の重力が釣り合っており、少ない推進剤で周期的な軌道を維持することが可能である。EQUULEUS では内部に 3 つのミッション機器を搭載しており、それを用いた地球プラズマ圏の撮像、月面衝突閃光の観測、月近傍ダスト環境の観測を実施する (図 4.5)。

一方で ISSL6U は、地球低軌道の地球観測衛星を想定する。一つの望遠鏡をミッション機器として搭載し、姿勢制御を行いながら、地表面の観測を行う。EQUULEUS と異なり推進系は搭載しておらず、地球磁場を用いたアンローディングが可能であるため、磁気トルカや磁気センサーを搭載している。

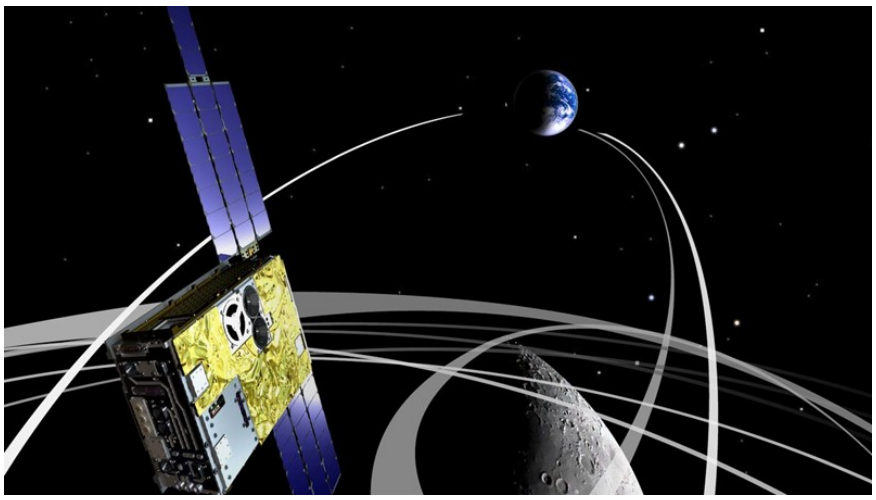


図 4.4　EQUULEUS ミッションイメージ図 (Image Credit: 東京大学/JAXA)

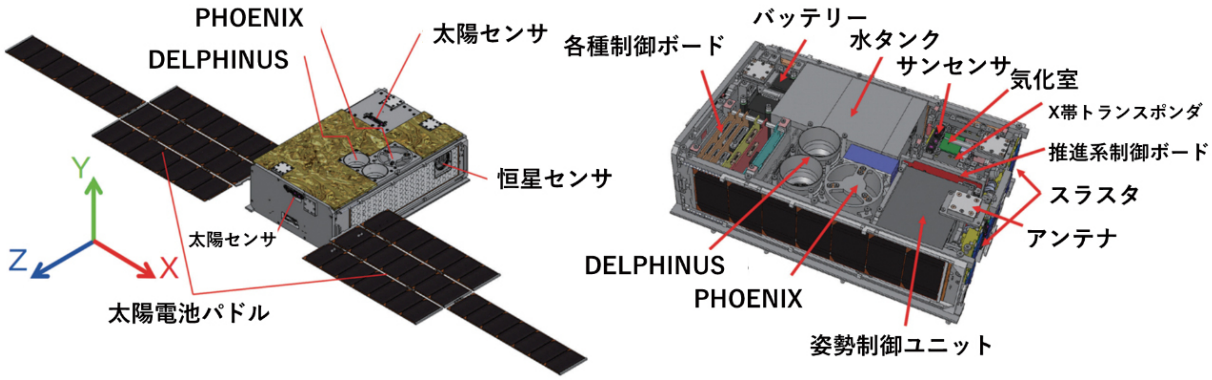


図 4.5　EQUULEUS 外観、内部構造図 (Image Credit: 東京大学/JAXA)

4.2.2　知識分子の生成

本研究では、EQUULEUS FM 開発時に使用していた不具合管理表を元に生成し、不具合に関する知識分子を生成する。EQUULEUS 不具合管理表では下記の項目が含まれ、不具合発生時にエンジニアによって記載された。

1. 発生のタイミング
2. 不具合の関連するコンポーネント
3. 事象の詳細
4. 原因特定の経緯、原因
5. 処置と結果

なお、開発完了の際には、これらの不具合が全て原因特定、あるいは何かしらの対策が実際されたことを確認した。 本研究では、事象の詳細、原因を元に知識分子の Condition Graph や Analysis, Content, Operator を記述を行った。また、処置と結果については Content の中で対策として記述した。さらに、不具合を未然に防ぐために行うべき、明らかな設計、試験のチェック項目があればそれを知識として追加で記載した。なお、知識の定義方法については第 3.6 章の、知識分子の定義方法を元に定義をおこなった。まらに、パラメータの異常、正常に関しては、-1 (正常値以下), 0 (正常値), 1 (正常値以上、もしくは単に異常) として表し Analysis の条件、Operator の効果を表した。これにより、計 40 個の EQUULEUS で起きた故障に関する知識分子を生成した。(現状のオントロジーの定義では、モデルかできない知識については、識別を行い、その原因の考察を行った。)

Appendix A に、知識分子とした不具合の一覧を示す。

また、今回 EQUULEUS の不具合に加えて、物理的に想定される不具合の伝搬に関する知識として以下のような知識を、知識分子に定義した。

- あるコンポの形状がゆがむと、そのコンポに結合した別のコンポの外側への圧力が異常になる
- 圧力が異常なコンポは、形状がゆがむ可能性がある
- 電流量が大きいと、そこに関する発熱量も異常になる
- 構造への入熱量が増えると、形状がゆがむ可能性がある。

これにより、EQUULEUS の知識により特定した不具合リスクに対して、発生する波及効果についても推論することを想定した。

4.2.3　ISSL6U のモデル

知識の結合先である ISSL6U は以下のようにモデルを構築した。

- 各機器のクラスはコンポーネントの抽象クラス + ”_6 u”として定義した。
- コンポ間の配線の結合や、配線が基盤内部を通っているか、機器の締結相手、熱的な結合をシステムモデルに入力した。
- 機器の機能を入力するとともに、パラメータの値を入力した。

配線の結合や、機器の締結相手、熱的な結合をシステムモデルに入力した。具体的な定義の一部分を表 4.4, 4.5, 4.6 に示す。 以上のようにモデルを構築した。モデル化したグラフの全体像を図 4.6 に示す。いくつかのコンポを中心に、巨大なグラフが生成されていることがわかる。特に依存関係が集中している部分として、Back Plane と呼ばれる電源中継基盤があげられる。その部分について拡大した様子を図 4.7 に示す。

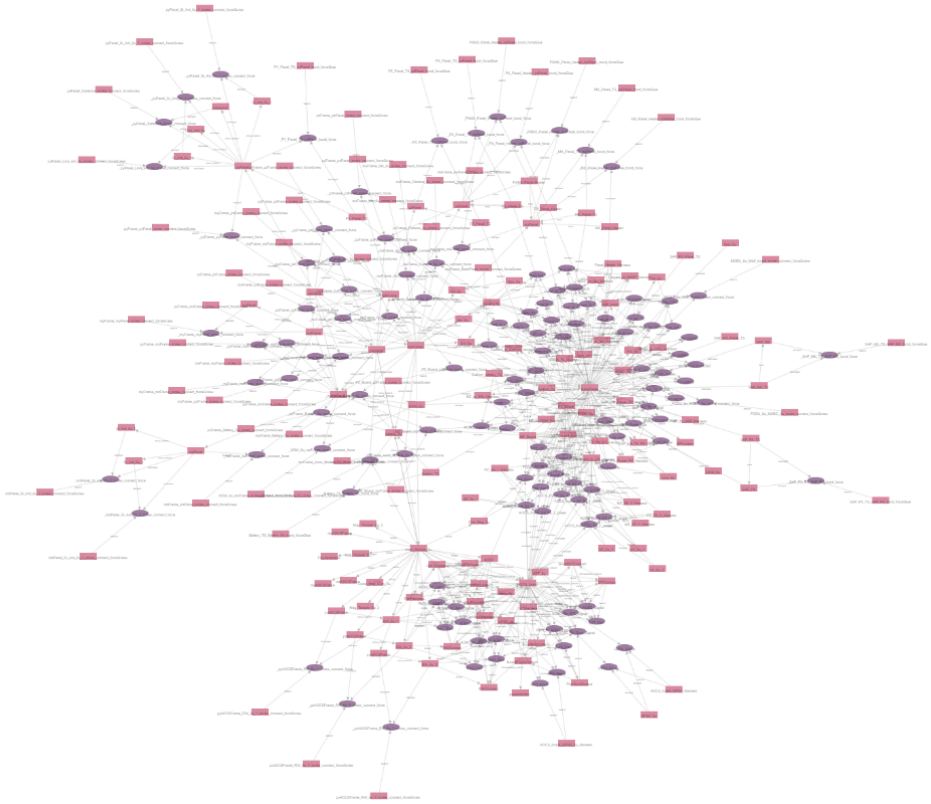


図 4.6　ISSL6U モデル全体像。四角形がコンポを表し、丸がコンポ間でやり取りされる電力、信号、熱などを表す。各コンポの機能、パラメータとその値に関しては表示を省略している。

4.2.4　提案手法による設計故障リスク提示の実施

実衛星での知識再利用の効果の検証として、まず設計フェイズで、設計に関する故障リスク、およびその原因、波及効果、対策を整理する Fault Modes and Effect Analysis (FMEA) を、知識分子により自動で実施した。

4.2.4.1　設定

以下の設定で推論を実施した。

1. 知識結合のアルゴリズムは、知識ごとに定義した SPARQL ベース or 類似度ベースか、の設定を使用し、知識ごとに適用
2. 知識の波及効果の最大数は 3 で設定

また、結びついた知識について、重要な知識の識別を第 3.5 章で提案した手法によって行った。

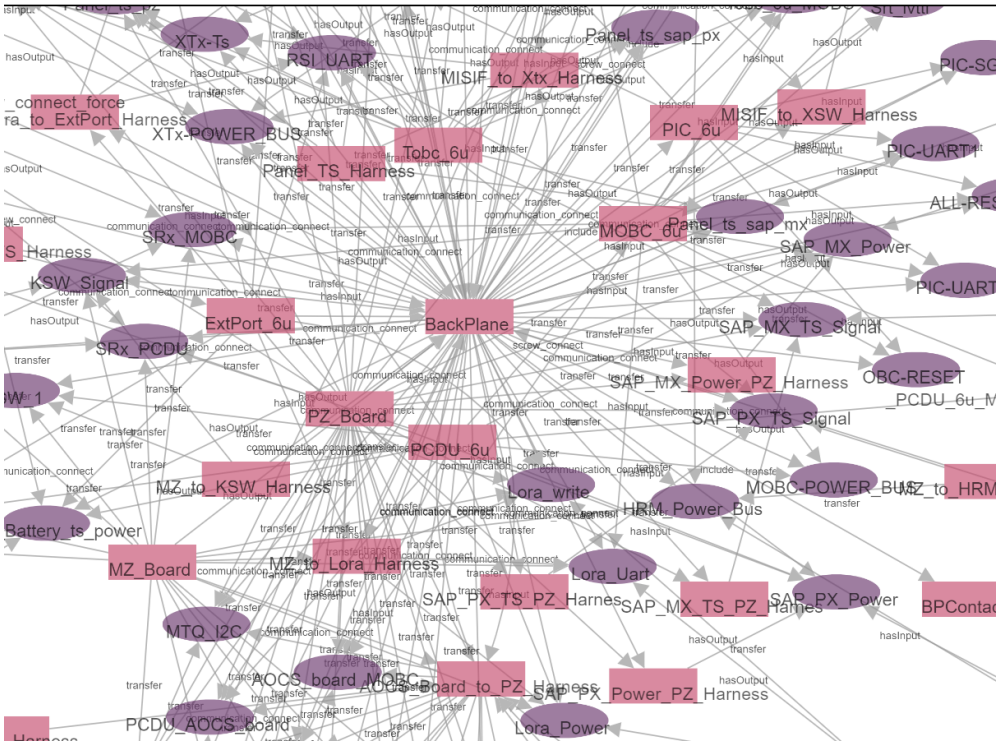


図 4.7　Back Plane 周囲の拡大図

その際の設定は下記のとおりである。

1. すでにユーザーが持っている知識として ID EQU_FAULT_17 (ハーネスに対して抜き差しでハーネスの断線、損傷が起きる) を指定した。
2. ユーザーが主に責任を持っているコンポーネントとして、姿勢制御アクチェーターである Reaction Wheel を指定した (RW_6u_1)
3. ユーザーが興味を持っている観点として「電源」を指定した。

これにより、リアクションホイール周辺の機器 (姿勢制御機器が中心) にかかわる知識で、電源に関する知識が重要と識別されると期待される。また、すでに持っているハーネスに関する知識は重要度が下がると期待される。

4.2.4.2　結果

知識結合のアルゴリズムに基づき、前述の ISSL6U に対して EQUULEUS 不具合の結合を行い、不具合リスクの提示を行った。その結果、ISSL6U 全体において、733 か所に知識の結合が発生し、不具合リスクが提示された。EQUULEUS で発生した知識分子は 40 個ほどであるため、同じ知識分子が、複数個所に結合していることにより、知識の提示数が増えている。どの故障に関する知識が、いくつの箇所に結合しているかを図 4.8 に示す。この図から、わか

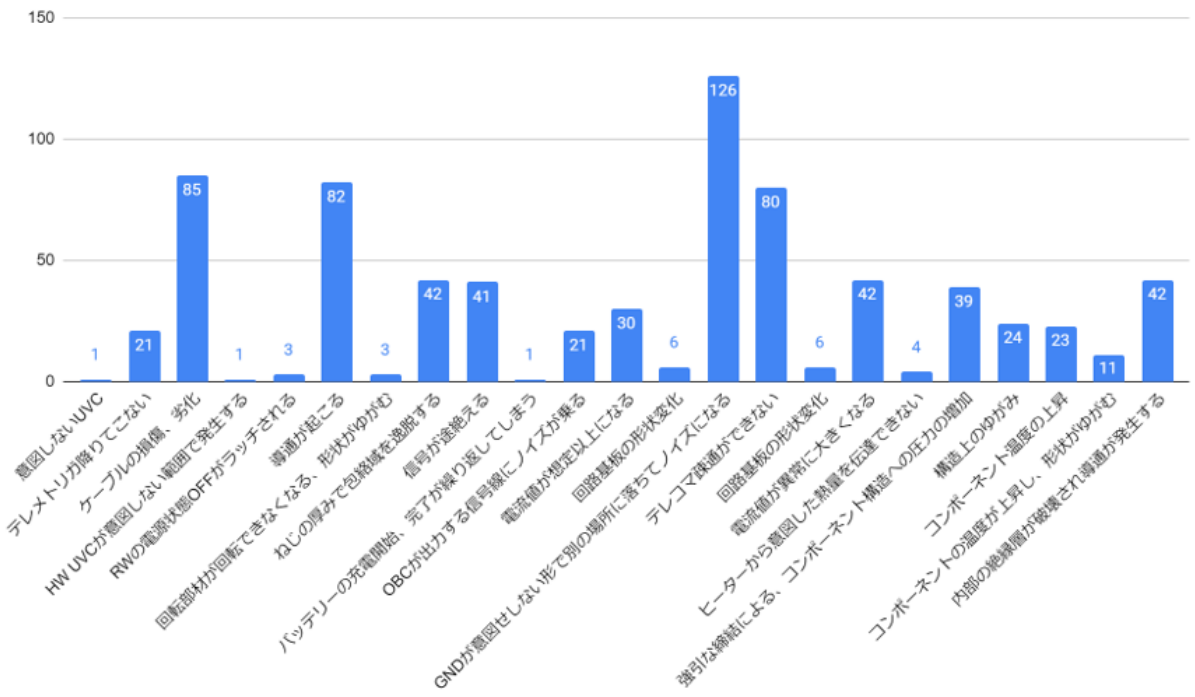


図 4.8　ISSL6U モデルに結合した不具合知識とその結合箇所の数

るように GND (電源におけるグラウンド) が意図せずに別のコンポと一致してノイズになる不具合の数が最も多く結合している。また、ケーブル劣化や導通のリスクが多くの箇所で指摘されている。これらは、実際衛星開発においてもよく起こる不具合であるため、直観に即した結果になっているといえる。

次に、結合した知識を、結合先のコンポごとに整理すると図 4.9 のようになる。多くのコンポーネントと依存関係にある電源中継基板 Back Plane, PZ Board, AOCS Board が、不具合の結合数が最も多くなった。実際、この依存関係の多さから不具合リスクも高いと考えられるため、これらの結果は妥当であるといえる。なお、同じコンポーネントに複数同じ知識分子が結合しているのは、一つの知識分子が複数のコンポをもとに結合する場合、そのコンポの組み合わせだけ同じ知識分子が、一つのコンポに結合したためである。

さらに、結合した知識のうち、「実際の ISSL6U 開発で、該当する不具合、もしくはそれに類する不具合が発生したもの」で、かつ「ISSL6U における人間の FMEA レビューには指摘、発見されなかった不具合」が存在した。それらを表にまとめる。これらの知識は、実際に設計フェイズで本手法を使用していれば、防げた可能性が高い不具合である。

また、知識分子の結合の評価により、上位に来た知識を表 4.8 に示す。上位 3 つのの知識は、確かに RW に近い姿勢制御モジュールや、ハーネスに関する知識であり、ハーネス以外で

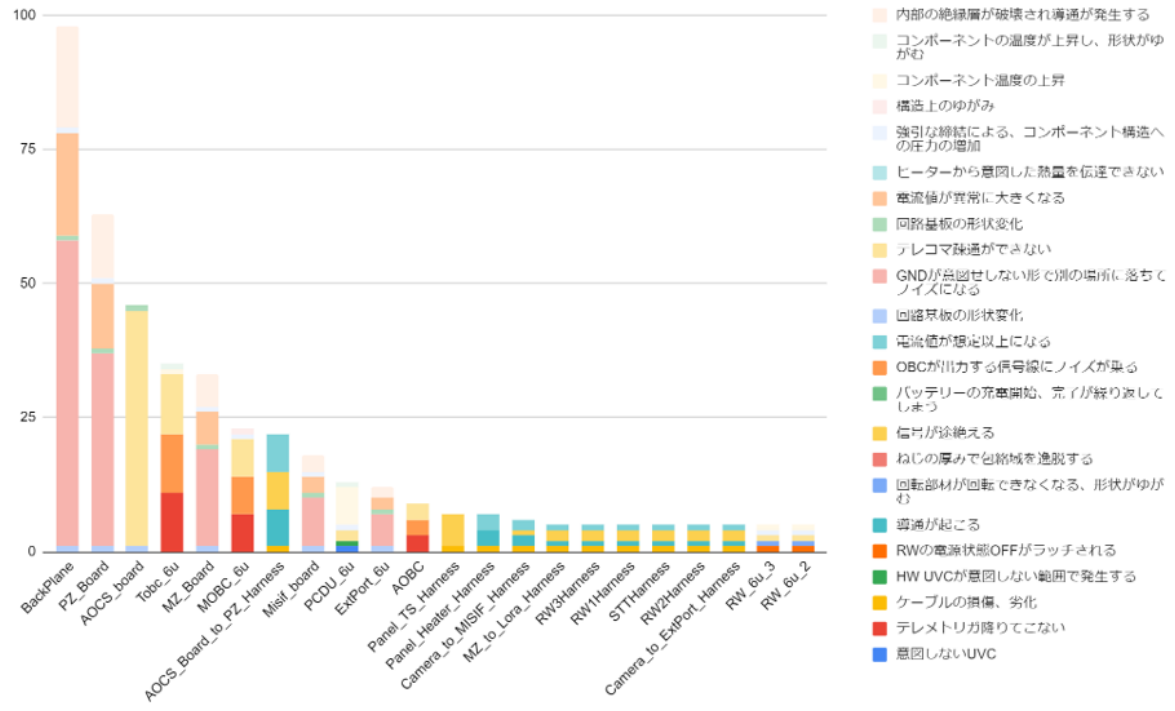


図 4.9　ISSL6U モデルに結合した不具合知識の結合場所と結合箇所の関係

の電源に関する注意点であり、期待した結果が得られているといえる。このように知識分子の評価として、ほしい観点やいらない知識を指定しておくことで、重要な知識を見逃さないようにすることができるといえる。

最後に、知識分子が複数結合し、波及効果的な故障が発見された様子を示す。図 4.10 に示すように、導通等による電流の増加と、それによるコンポーネントの発熱、それによるコンポーネントがゆがむ、といった連鎖が見られた。また、構造に関しては、結合してあるコンポのゆがみが波及し、別のコンポへとゆがみがつながる様子が見られた。このように、知識分子の Operator により故障の波及効果の様子が表現できるといえる。一方で、今回定義した知識は、定性的な故障知識のみであるため、実際には無視できるようなゆがみに関する不具合リスクの提示まで行っている。これらの、無視できる不具合リスクを取り除くには、Analysis を用いた定量的な解析を実施する必要があり、より知識分子の定義を洗練化する必要があると考えられる。

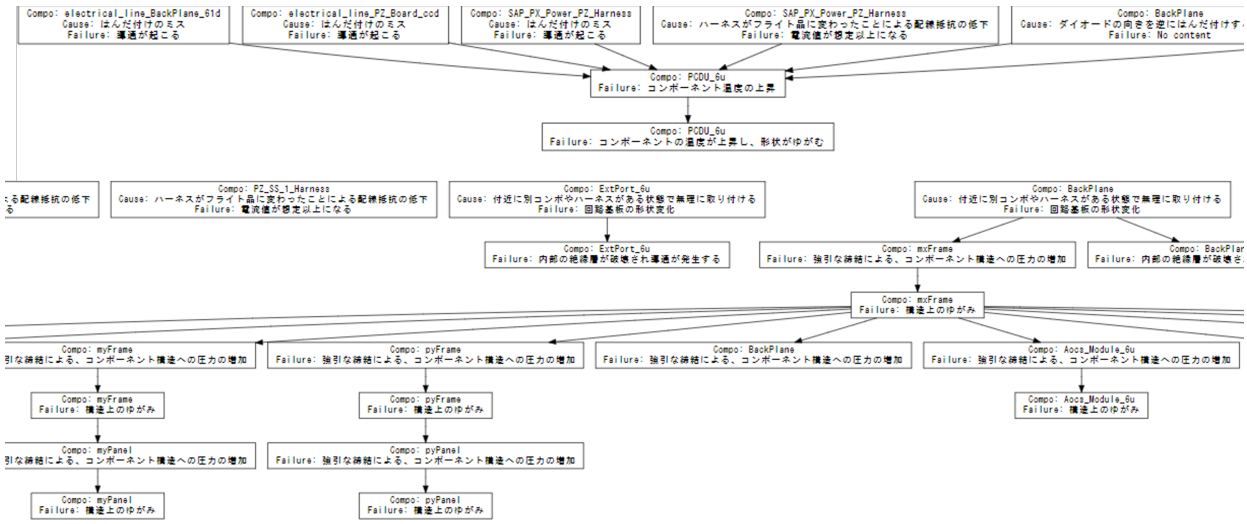


図 4.10　ISSL6U モデルに結合した不具合知識において、波及効果的に複数の知識分子が連続して結合した箇所の例

4.2.5　提案手法による故障原因の推定

別の実衛星での知識再利用の効果の検証として、衛星の機能にないか不具合が発生した場合に、その不具合原因の推定を知識分子を用いて実施した。

4.2.5.1　設定

以下の設定で推論を実施した。

1. 知識結合のアルゴリズムは、知識ごとに定義した SPARQL ベース or 類似度ベースか、の設定を使用し、知識ごとに適用
2. 知識の波及効果の最大数は 5 で設定

また、EQUULEUS から生成した知識分子における、Analysis と Operator を入れ替えた。これにより、波及効果となる状態を Analysis として判定し、Operator で原因となる状態が存在することを断定するように推論が可能になる。(前述のとおり、パラメータの異常、正常に関しては、-1 (正常値以下), 0 (正常値), 1 (正常値以上、もしくは単に異常) として表す)。

今回は、衛星の精 Gyro センサーの角速度検知機能に不具合が発生したとし、その原因の推論を実施した。

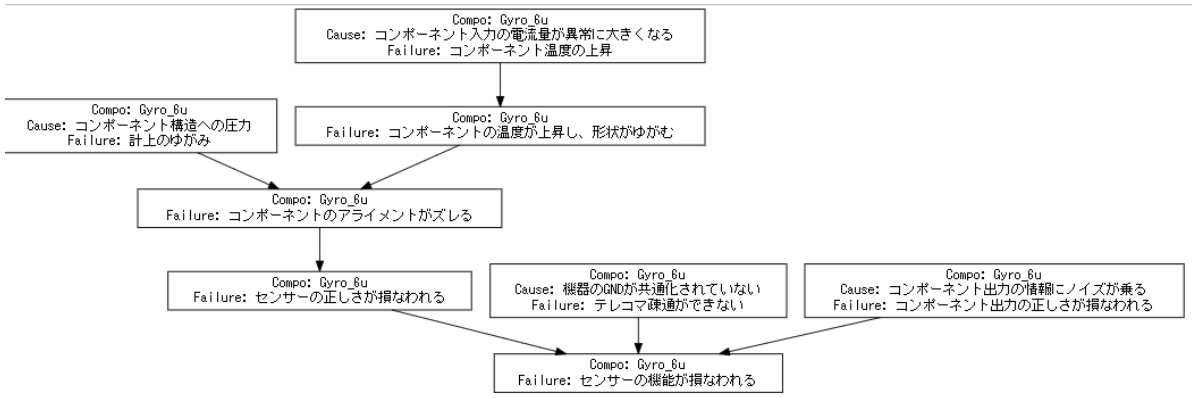


図 4.11　ISSL6U モデルにおける角速度検知の異常に対して、原因の推論を知識分子の結合により実施した結果

4.2.6　結果

推論の結果を図 4.11 に示す。図のとおり、Gyro センサーの角速度検知の不具合に対して、その根本となる事象が導かれていることがわかる。また、導かれている事象に関しては、実際に起こりうるものであり、妥当であると考えられる。一方で、さらに根本原因を導けるような形で、推論の最終端が残っている部分がある (コンポーネント出力にノイズが乗る、コンポーネントの電流量が以上に大きくなる、コンポーネント構造への圧力)。これについては、知識分子の定義不足が考えられ、EQUULEUS に加えたさらに多様な知識分子を定義することにより達成することができると考えられる。

4.2.7　本章全体の結論

本章では、実衛星を忠実にモデル化し、それに対して実際に発生した不具合知識の再利用を行った。モデルにおける Entity の数が非常に多いため、大量の知識が結びつくことになった。その際、実際に衛星開発途中で起きたような不具合を知識の再利用で指摘することができた。また、3.5 章で提案した知識評価手法を活用することができ、確かに重要な知識を識別することができた。また、複数の知識を組み合わせた不具合の波及効果の推論や、不具合事象から原因の推論を実施できることを示した。

4.3 検証 3 提案手法と従来手法の比較

本章では、提案手法である知識分子による知識再利用と、先行研究でみられるようなルールベースによる推論の比較実験を行い、提案手法の優位性の検証を行う。これにより、知識分子の特徴である「複数種類の知識の組み合わせ」、「Analysis による定量的な推論」、「類似度ベースによる結合」の効果を示す。

比較実験では以下の方法で行う。

1. 実際の Cubesat プロジェクトで発生した不具合や知見を、別のプロジェクトで再利用するシミュレーションを行う。その際、知識は知識分子、あるいはルールベースの手法を用いて再利用する。
2. シミュレーションの結果を比較し、提案手法となる知識分子を生かした手法とルールベース手法で、再利用のプロセス、提示される知識にどのような差が出るかを検証する。

なお、ここでいるルールベースとは、単純に構造的な結合のみが、完全に合致した知識をユーザーに提示することを想定する。今回の解析ではその状況を、知識分子で知識を定義した結果から、知識分子の特徴である Analysis, Operator を取り除き、Condition Graph のみですべての結合を SPARQL ベースで行うことで再現をした。

4.3.1 再利用する知識の内容

今回は、実際に 6UCubeSat の EQUULEUS 開発中に起きた不具合、および未然に防止した不具合に関する以下の経験を知識として再利用することを考える。

不具合 1

XACT-50 (EQUULEUS で使用された姿勢制御モジュール) が温度 -15 degC で電源を ON した時、Power Control Unit の過電流防止機能にかかり、意図せず電源が落とされた。原因として、XACT-50 が温度が低下すると、平温で 3.8 A のところ、過電流値が最大 4.2A 程度まで上昇すること、Power Control Unit (PCU) の過電流閾値が、4.0 A で設定されていたことが考えられた。ただし、XACT-50 がなぜ低温時に過電流が上昇するかについては、XACT-50 の内部の解析が必要である一方で、XACT-50 が購入品で内部構造はブラックボックスなため究明できなかった。

不具合 2

ハーネスの太さが、規定される許容電流量に対して十分ではなく、焼き切れるリスクがあった。

4.3.2　再利用先の別プロジェクト衛星

今回、提案手法の優位性を示すために、二つの異なるアーキテクチャをもつ衛星開発プロジェクトを想定し、知識の再利用を実施した。

衛星 1
: XACT-50 を姿勢制御装置として搭載する、月面観測衛星。Laser Altimeter を、ミッションコンポーネントとして搭載している。XACT-50 の想定温度範囲は-5degC から 40degC までを想定してる。また XACT への過電流の閾値は、4.0 A としている。

衛星 2
: 内製化した独自の姿勢制御モジュールを利用した地球観測衛星で、カメラをミッションコンポーネントとして搭載している。独自姿勢制御モジュールの想定温度範囲は-20 degC から 40 degC までを想定してる。また過電流の閾値は、4.0A としている。

衛星 1 のような衛星として NASA SLS で打ち上げ予定の Lunar Flashlight などがある。衛星 2 は、前章で紹介した ISSL6U が当てはまる。また、そのほかのコンポーネントも定義し、パラメータについても一部分定義を行った。

4.3.3　知識再利用シミュレーション

前述の設定に対して各手法を用いて知識を再利用するプロセスを考える。

4.3.3.1　ルールベース

まず不具合 1 に関しては、

- XACT-50 が-15 degC 以下で、XACT-50 の過電流が 4.2 A になる
- PCU の過電流閾値を上回った場合、PCU の過電流防止機能にかかる。

というように原因を分解でき、それぞれをルールとして格納することになる。ルールベース (if-then ベース) の手法にもいくつか種類があり、パラメータベースのものと、定性的な構造をもとに、条件を判断する方法の二つが主として考えられる。パラメータベースの場合、事前にどの種類のパラメータが存在し入力されるかが定義される必要がある。今回の場合、不具合 1 に関しては、XACT-50 を用いたアーキテクチャに対して、XACT-50 の温度パラメータを入力しとした条件を考えることが可能である。ただし、知識再利用の際、アーキテクチャがわからない状況、すなわち入力として与えられる衛星に XACT-50 の温度というパラメータが存在するかどうかわからない状況で、このパラメータベースのルールを定義しておくことは難しい。

また、定性的な条件、すなわち使用コンポが XACT-50 であるということを条件に、知識を再利用するケースも考えられる。この XACT-50 が衛星に定義されていれば、条件に当てはまったとして、上記の不具合 「低温時に XACT-50 の電流値が上昇する」を提供することになる。

PCU についても同様に、PCU が存在することを条件に、上記の知識を提供すると、if-then で定義できる。また、ハーネスに関しても、ハーネスが構造に含まれていれば、「許容量を超えている場合に、断線のリスクがある」というルールを定義することが可能である。ルールベースの方法では、このようなコンポの存在を条件に知識を再利用することにする。

4.3.3.2　知識分子ベース

ルールベースと同様に、不具合 1 は、「XACT-50 の過電流に関する不具合」と「PCU の過電流防止機能」に分けて知識を考えることができる。XACT-50 の過電流に関しては、以下のように知識分子に落とし込むことができる。

1. 「XACT-50 が電源供給されている」をオントロジーを用いて Condition Graph に入れ込む
2. XACT-50 に定義されている Temperature パラメータを読み込み、Analysis で、-15 degC 以下であれば電流値 4.2A として出力する Analysis を定義する。
3. Analysis の結果をもとに、Operator で、XACT-50 の出力電流値を 4.2 A とする Operator を定義する
4. この知識の物理、機能的な要因はわかっていないため (すなわちどの程度一般化してよいかの範囲が、この不具合事象からはわからないので)、類似度ベースで推論をするように、知識に定義する。

次に、PCU の過電流防止機能については、PCU に限らず過電流防止機能が付いた機器が電源供給を行う場合は同様の事象が発生すると考えられる。そのため、以下のように知識分子に落とし込む

1. 「Component が過電流防止機能を持ち、ほかコンポーネントに電源を供給している」をオントロジーを用いて Condition Graph に入れ込む
2. Component に定義されている過電流閾値のパラメータと、出力電力の電流値を読み込み、Analysis で、電流値が閾値を上回っていれば、知識分子を結合して過電流防止機能が発動するとする Analysis を定義する。
3. 過電流防止機能の結果、出力電力の電流値を 0 にするように Operator を定義する。
4. 物理、機能的な要因が分かっているためルールベースで結合判定を行う

また、ハーネスに関しては許容電流値が太さから計算できるため、以下のように定義する。

1.「ハーネスが電流を流す」をオントロジーを用いて Condition Graph に入れ込む
2. ハーネスに定義されている、ハーネス太さ、電源の電流値を読み取り、太さから計算される定格値を上回っているかどうかを判定する Analysis を定義する。上回っている場合、結合を判定する
3. 物理、機能的な要因が分かっているためルールベースで結合判定を行う

さらに、今回の不具合から、衛星のコンポにかかわる設計の重要な機能、パラメータの存在に関する知識を合わせて定義することができる。すなわち、

1. Power Control Unit には一般的に過電流防止機能がついている
2. ハーネスには太さという概念が定義されている必要がある。

これらに関して、モデルに定義されていなければ、定義を促す知識分子を定義することで、不十分なモデルにより、定義した不具合知識が利用されないことを防ぐことができる。以上の知識分子を表にすると、表 4.9,　4.10 のようになる。

表 4.9　不具合に関する知識分子

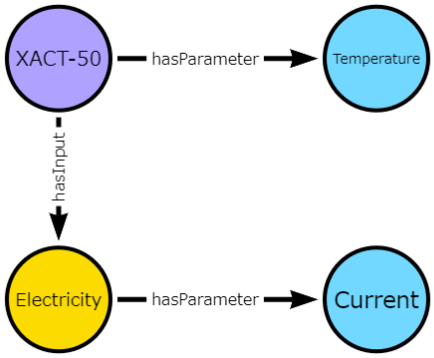
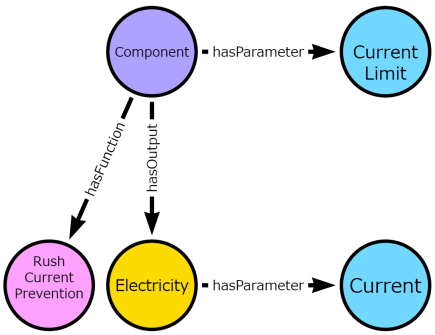
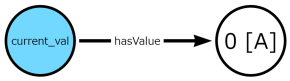
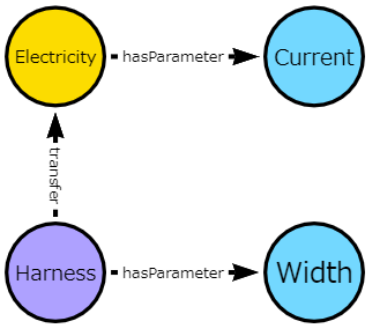
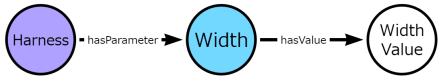
1		**Content** 故障:XACT-50 の突入電流の大きさが 増加する 設計確認:メーカに温度による突入 電流の変化を確認しておく **Analysis** if Temperature ＜ -15 Then current_val = 4.2 else current_val = 3.8 **Operator** 
2		**Content** 故障:意図せずにコンポの電源が切れる 設計確認:想定される電流値を 閾値が超えているかを確認する **Analysis** if Current ＞ Current Limit: Knowledge Connect **Operator** 
3		**Content** 故障：熱により断線する 設計確認：ハーネスの許容電流を、 実際の電流値が下回っているか確認する **Analysis** If 4*width^2 + 0.25 ∗ *width* − 0.27 ＜ Current: Knowledge Connect **Operator**　なし

表 4.10　設計補足に関する知識分子

1		**Content** 補足：PCU には過電流防止機能が一般的につく **Analysis**:なし **Operator** 
2		**Content** 補足：ハーネスは太さというパラメータが重要である **Analysis** width value = Input (Width Value [mm]) (ユーザーへの入力を依頼) **Operator** 

4.3.4　結果

前述のアーキテクチャの異なる二機の衛星に知識の再利用を実施し、提示された不具合リスクの結果を比較する。

4.3.4.1　衛星 1　XACT-50 を用いた衛星プロジェクト

ルールベースで知識を再利用した結果を表 4.11 に示す。また、知識分子ベースで知識を再利用した結果を表 4.12 に示す。

結果として、ルールベースのほうがより多くの不具合リスクが提示されている。これは、今回のルールベースが定性的なルールのみを考慮しているためであると考えられる。すなわち、

- XACT-50 に関しては運用温度が-15 degC より大きいが、XACT-50 の故障リスクとして提示されている
- PCU の過電流防止機能についても、同様に電流値自体の値を見ずに結合が起きている
- Harness に関しても、電流値とその太さの関係がみられていない

一方で、知識分子のほうでは、上記の定量的な値を考慮し、考慮する必要のない知識 (リス

ク) は提示されていない。また、ルールベースでは発見されていなかった、Battery と PCDU(電源制御基板) を接続するハーネスに関する不具合が提示されている。これは、知識分子によって、宇宙機モデルに定義されていなかったハーネス太さを、設計の補足を担う知識分子が結合し、太さのパラメータを定義したため、ハーネスの不具合に関する知識分子も結合できたためであると考えられる。

4.3.4.2　衛星 2 独自姿勢制御モジュールを開発した衛星プロジェクト

ルールベースで知識を再利用した結果を表 A.1 に示す。また、知識分子ベースで知識を再利用した結果を表 4.14 に示す。

ルールベースは、多くの知識が提示されものの、姿勢制御モジュールの過電流が大きくなる不具合に関してはリスクが提示されていない。これは、XACT-50 を厳密な条件として判定し、知識を利用したからであると考えられる。

一方で、知識分子ベースの法は、姿勢制御モジュールに関しては知識が結合している。これは、類似度による推論を実施し、姿勢制御モジュールが電源供給されている部分に関して、今回の設定では類似度が高いと判定されたからと考えられる。また、結果として姿勢制御モジュールの電流値が上昇し、PCU の過電流機能の不具合リスクが提示されていることがわかる。ハーネスに関しては、衛星 1 に加えて、姿勢制御モジュールで電流値が上昇した部分に関して、不具合リスクが提示されており、こちらも現実に即していると考えられる。

表 4.11　ルールベースでの衛星 1 への知識再利用結果

対象コンポ	原因	不具合	設計での確認項目
Xact_50_lf	XACT-50(姿勢制御モジュール) の温度が-15degC 以下になる	突入電流の大きさが増加する	温度変化をさせた時に ON OFF 実験を行い、起動できるかを確認する
PCDU_lf	過電流防止機能の閾値を電流値が超える	意図せず、コンポの電源が落ちる	閾値を電流値が超えないかを確認する
PZ_SS_2_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	電流の最大値を通して、暫くおいてみて問題ないかを確認する
MISIF_to_Xtx_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	電流の最大値を通して、暫くおいてみて問題ないかを確認する
Panel_Heater_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	電流の最大値を通して、暫くおいてみて問題ないかを確認する
PZ_SS_1_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	電流の最大値を通して、暫くおいてみて問題ないかを確認する
Panel_Heater_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	電流の最大値を通して、暫くおいてみて問題ないかを確認する
MZ_to_HRM_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	電流の最大値を通して、暫くおいてみて問題ないかを確認する

表 4.12　知識分子ベースでの衛星 1 への知識再利用結果

対象コンポ	原因	不具合	設計での確認項目
ExtPort_Battery TS_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する

表 4.13: ルールベースでの衛星 2 への知識再利用結果

対象コンポ	原因	不具合	設計での確認項目
PCDU_6u	過電流防止機能の閾値を電流値が超える	意図せず、コンポの電源が落ちる	想定される電流値を、閾値が超えているかを確認する
MZ_to_HRM_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
Panel_Heater_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
MZ_SS_4_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
Panel_Heater_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
MISIF_to_Xtx_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
PZ_SS_2_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
PZ_SS_1_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
MZ_to_SRx_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
Panel_Heater_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する

RW1Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
RW3Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
MZ_to_Lora_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
RW2Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する

表 4.14　知識分子ベースでの衛星 2 への知識再利用結果

対象コンポ	原因	不具合	設計での確認項目
Aocs_Module_6u	XACT-50(姿勢制御モジュール) の温度が-15degC 以下になる	突入電流の大きさが増加する	メーカーに温度による電流、突入電流の依存性を聞いておく
PCDU_6u	過電流防止機能の閾値を電流値が超える	意図せず、コンポの電源が落ちる	想定される電流値を、閾値が超えているかを確認する
ExtPort_Battery_TS_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する
AOCS_Board_to_PZ_Harness	ハーネスの太さが許容電流量に対して十分ではない	熱により断線する	ハーネスの許容電流を、実際の電流値が下回っているか確認する

4.3.5　結論

本章で示した検証により、提案手法が、単純なルールベースと比較し、以下の特徴を有していることが示された。

- Conditioin Graph と Analysis の組み合わせで、より厳密に知識をモデル化することが可能である。これにより無駄な知識の再利用を防ぐことができる。
- 設計の補完などの知識分子と故障に関する知識分子を組み合わせることで、モデル化されていない詳細な設計に踏み込んで故障のリスクを特定することが可能である。あるいは、パラメータの未定義など、モデル化のミスを許容して、知識を再利用することが可能である。
- 類似度ベースの推論により、異なるアーキテクチャでも柔軟に知識を再利用することができる。

4.4　検証 4 ユーザーインタビュー

本研究では、衛星開発経験を持つユーザーを用意し、提案手法を利用してもらい、効果、利用しやすさの評価を実施した。これにより、提案手法がシステム全体として、実際に利用され、プロジェクトで効果を発揮することを検証を行った。本章ではその検証試験について説明する。

4.4.1　試験方法

以下の方法で試験を実施した。

4.4.1.1　ツール

前述のツールを用いて試験を実施した。試験の際は、モデル、知識記述のエクセル、csv、および CUI の実行ファイルをユーザーに提供し、ユーザーに操作をしてもらった。

4.4.1.2　使用したモデル、知識分子

宇宙システムモデルをユーザーに一から記述してもらうのは時間がかかるため、事前に完成した宇宙システムモデルを利用した。宇宙システムモデルは、CubeSat の姿勢制御コンポーネントを 1 つに包含した姿勢制御モジュールをモデル化し、内部のコンポーネント間の電源、構造、熱的な接続を考慮した。図に全体像を示す知識分子に関しても同様に、複数の定義するのに時間がかかるため、事前に用意した。知識は、衛星コンポの機能検証に関する知識であり、

各コンポの機能の何を、どのように確認すべきかを記述している。

4.4.1.3 手順

インタビューは下記の手順で実施した。

1. 知識記述、宇宙システムモデルの記述方法のデモンストレーション
2. CUI の実行方法のデモンストレーション
3. 知識記述をユーザーにしてもらう、もしくはユーザーに宇宙システムモデルを追加してもらう。
4. ユーザーに姿勢系エンジニアとして、検証計画を立てる想定で、考えられるうる検証項目を述べてもらう
5. ユーザーにツールを利用してもらい、検証項目に関する知識を取得してもらう。
6. 取得した知識を見て、見逃した知識や思いつかない知識があったか、利用しやすさはどうだったかを聞き取る

4.4.2 結果

表 4.15 のようなインタビュー結果となった

4.4.3 結論と考察

前述のケーススタディに対する考察をここに述べる。まず、有効性についてであるが、以下の 2 つの面で効果があるという言及があった。

- 知っているが、気づかなかった知識を獲得できる
- まったく知らない知識を獲得できる

本研究では、もともとの目的は 2 を想定していたが、1 に対しても、見逃しを防ぐという面で効果があるといえる。そのため、要素数の多い複雑なシステムや、レビューに十分にリソースが取れないときにも本研究が効果を発揮する可能性がある。

使用しやすさについては、結果として IF を構築したのは良かったものの、積極的な利用をする程の体験はまだ提供できていないという結果になっている。改善方法として、知識を記述するインセンティブをより増やす、すなわち知識分子が利用できるアプリケーションをより増やすことがあげられる。一方で、それだけでは統一的な知識が難しい、という点はこれでは解決しない。類似度を用いた推論により、多少のブレはロバストに推論ができる一方で、大きな過ちは、知識分子に対する人間のレビューで担保する必要があり、利用するユーザーにとって、

信頼の高い利用は労力がかかる。Appendix に記載の自然言語処理を用いた知識分子の生成など、より人間の言葉で概念を記述しつつ、処理を実行できる枠組みを構築する必要がある。

最後に、改善点であるが、知識記述のインセンティブの導入、他人からの信頼度評価が利用しやすさを向上するのは、比較的容易な改善で実行できると考えられる。インセンティブはアプリケーションの種類を増やすことで達成でき、信頼度評価は、実際に推論し、利用する際にユーザーから評価を入力できる仕組みを作ることで達成できる。より細かい粒度の知識に関しては、宇宙システムモデルをどこまで詳細に記述するかが重要になる。現在のエクセル IF では固定した入力しかできないが、より開発段階に合わせて細かい知識を入力できるように IF を組むことで、実現することは可能であると考えられる。ただし、細かい衛星の部品に関する概念も定義していく必要があり、より知識管理が膨大になっていく可能性がある。知識が膨大になると、重要な知識を見逃すだけでなく、知識のマッチングに計算時間も必要になる。現状、知識分子の評価により、重要知識を識別しているが、それに加えて、推論を実施する知識を選別し、計算時間を抑える必要がある。設計変更時に置けおる改善は非常に重要であると考えられる。特に知識分子は検証計画やチェックリストを生成することに使えるため、HW をモデル情報として管理しつつ、重要な update があった場合に、自動でチェックを走らせ確認を求める、といった運用は技術的に可能であると考えられる。

表 4.2　不具合 2 に関する知識分子

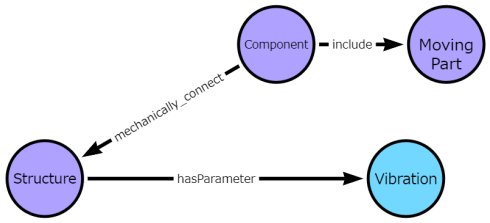
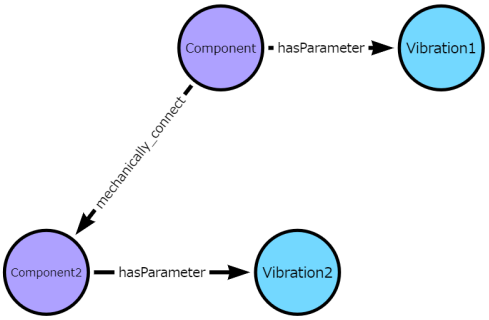
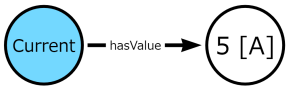
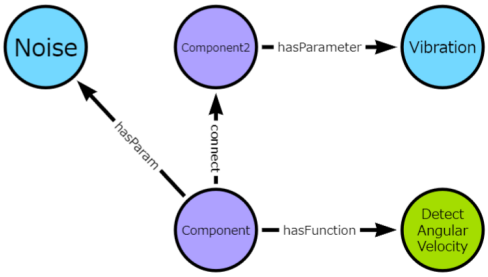
3		**Content** 故障:動く部品を持つコンポの振動が伝搬する 設計確認:コンポの振動の大きさと最低限の許容値を確認する **Analysis** なし **Operator** 
4		**Content** 故障:別の機器に振動が伝わる 設計確認:振動の伝搬、減衰が問題ないかを確認する **Analysis** Status(Vibration) == 1 **Operator** 
5		**Content** 故障：振動の伝搬、減衰が問題ないかを確認する 設計確認：センサーに機械的な振動ノイズが入らないか確認する **Analysis** Status(Vibration1) == 1 **Operator** 

表 4.3　結合した知識分子から生成した設計確認リスト

対象	故障リスク	設計での確認項目
RW	過電流シャットダウンにかかる	設計上必ず閾値が大きいかを確認する
Robot_arm	過電流シャットダウンにかかる	設計上必ず閾値が大きいかを確認する
RW	動く部品を持つコンポの振動が伝搬する	コンポの振動の大きさと最低限の許容値を確認する
Robot_arm	動く部品を持つコンポの振動が伝搬する	コンポの振動の大きさと最低限の許容値を確認する
Robot_arm	別の機器に振動が伝わる	振動の伝搬、減衰が問題ないかを確認する
Panel_x	別の機器に振動が伝わる	振動の伝搬、減衰が問題ないかを確認する
FOG	出力にノイズが入る	センサーに機械的な振動ノイズが入らないか確認する

表 4.4　ISSL6U コンポーネント定義の例

コンポーネント	タイプ	コンポが包含されている親コンポ
Xtx_6u	Xtransponder	
Xsw_6u	Switch	
Xantenna	Patch_Antenna	
Tobc_6u	Onboard_Computer	
Stx_6u	Stransmitter	Strp_6u
STTHarness	Harness	Aocs_Module_6u
STTContact	Contact	STT_6u
STT_6u	Star_Tracker	Aocs_Module_6u
Aocs_Module_6u	Aocs_Module	
AOCS_Board_to_PZ_Harness	Harness	
AOCS_board	Circuit_Board	Aocs_Module_6u
AOBC	Onboard_Computer	Aocs_Module_6u

表 4.5 ISSL6U 電源結合関係定義の例

接続元コンポ	接続先コンポ	電圧 [V]	電流 [A]	経由先
AOCS_board	RW_6u_1	12	0.8	BackPlane
PIC_6u	HRM_6u	5	1	RW2Harness
Srx_6u	PCDU_6u	12	0.2	MZ_to_SRx_Harness
AOCS_board	STT_6u	5	0.5	STTHarness
Tobc_6u	MX_Panel_Heater	5	1	BackPlane

表 4.6 ISSL6U 構造結合関係定義の例

コンポ 1	コンポ 2	接続方法	接続部材 (ねじ種類、接着剤)
mxFrame	BackPlane	screw_connect	Screw
mxFrame	Aocs_Module_6u	screw_connect	Screw
mxFrame	Battery_6u	screw_connect	Screw
SAP_PX_TS	SAP_PX	bond	Glue
SAP_MX_TS	SAP_MX	bond	Glue
PX_Panel_Heater	pxPanel	bond	Glue

表 4.7　結合した知識のうち、実際に ISSL6U 開発で発生し、人間のレビューで見逃されたもの

対象コンポ	原因	設計での確認項目	過去事例	不具合
AOCS_board	付近に別コンポやハーネスがある状態で無理に取り付ける	くみ上げ時、回路のそばに挟まるようなものがないか、当たっている構造物はないかを確認する	EQU FM 不具合表 id 88	回路基板の形状変化
BackPlane	回路基板端の GND ビアがブラケットと導通する	くみ上げ時にビアなどの接触で GND が意図せず落ちていないかを確認する	EQU FM 不具合表 id 93	GND が意図せしない形で別の場所に落ちてノイズになる
BackPlane	付近に別コンポやハーネスがある状態で無理に取り付ける	くみ上げ時、回路のそばに挟まるようなものがないか、当たっている構造物はないかを確認する	EQU FM 不具合表 id 88	回路基板の形状変化
PZ_Board	付近に別コンポやハーネスがある状態で無理に取り付ける	くみ上げ時、回路のそばに挟まるようなものがないか、当たっている構造物はないかを確認する	EQU FM 不具合表 id 88	回路基板の形状変化
PCDU_6u	PCU 内部の抵抗地誤差	HW UVC の閾値は抵抗地の誤差で変動するか、変動する場合それが許容できるか	EQU FM 不具合表 id 23	HW UVC が意図しない範囲で発生する

表 4.8 結合した知識のうち重要度評価が上位のもの

順位	対象コンポ	原因	不具合	Unexpect.	Action.	Interest	Sum
1	RW_6u_1	RW をコースティングさせ逆起電力が発生する	RW の電源状態 OFF がラッチされる	0.24	1.00	0.60	1.84
2	RW_6u_1	コンポーネント入力の電流量が異常に大きくなる	コンポーネント温度の上昇	0.15	1.00	0.66	1.81
3	RW_6u_1	機器の GND が共通化されていない	テレコマ疎通ができない	0.13	1.00	0.58	1.71
4	RW_6u_1	隣接するコンポーネントの形状が、設計と異なる、もしくは設計で干渉が気にされていない	強引な締結による、コンポーネント構造への圧力の増加	0.13	1.00	0.55	1.69
5	PyAOCS Frame _RW_6u_1 _screw _connect _forceScrew (AOCS 締結用ねじ)	ねじどめの際のワッシャ有無のコミュニケーションを行わなかった	ねじの厚みで包絡域を逸脱する	0.22	1.00	0.45	1.67

表 4.15　ユーザーインタビューの結果

ユーザー	有効性	使用しやすさ	将来の改善性
A	認知しない知識に気づくというのは確かにできそう	Excel の IF であれば、ある程度慣れればかけそう。ただしプロジェクト中に、書くとなると忘れること、手間に感じることは増えそう	知識を記述することへのインセンティブが重要。不具合の原因の推論に使えるとありがたい。他の人が知識分子をどう評価したかを見れると、実際に推論した結果の信頼性が分かるので良いと思う
B	知っていたが、忘れていた知識を見つけることができたそもそも知らない知識も知ることができた。機器配置の決定など、複数分野の知識が必要な検討を効率的に行うときなどに利用できそう	検証項目やチェックリストの生成のためなら書こうという気になる	専門の系から見ると知識の粒度が甘いとあまり役に立たない。より粒度が細かい知識を登録できると良い
C	知らない知識を知ることができたので、開発素人には良さそう	統一した記法で知識を書くのは難しそうであり、それによってストレスがたまる可能性がある。	設計変更時に、以前気を付けていたことを忘れて気づかないまま、不具合につながるケースがある。そのようなケースで常にこのようなツールで警告できると嬉しい。

第 5 章

議論

5.1　本研究の目的と実装の対応

本研究の目的と、それを達成するために検討した実現方法と、本論文におけるそれらの説明箇所の関係を表 5.1 を再掲する。提案手法により、宇宙システム開発のための効率的な知識再利用に必要な機能は達成されている。また、システム全体として、目的とする効率的な知識再利用が実現できているかについては、第 4.4 章で実施している。

5.2　提案手法がもたらす効果

本研究は、下記のような価値を宇宙プロジェクト開発を行うエンジニアに提供する。

- 認知していない知らない知識を教えること (Unkown Unknown)
  ユーザーが宇宙システムを設計、開発する際に、必要な知識を提供する。その際ユーザーはその必要な知識について認知する必要がない
- 認知していない知っている知識に気づかせる (Known Unknown)
  ユーザーが知っている知識であっても、それを設計時に適用可能であること、あるいは重要であることを認知せず、見逃してしまう場合があり、そのような見逃しを防ぐことが可能である。例えば、システム自体が複雑で、様々な観点で設計を見る必要がある場合や、設計変更時で意思した設計確認が難しい場合などがそれに当てはまる。
- 過去の知識を組み合わせて、新しい知識を提供する
  本研究で提案する知識分子は Operator により、宇宙システムのモデルを更新する。これにより、過去に全く同じ不具合が発生していなくても、複数の知識を組み合わせることで見える不具合リスクの提供を行うことができる。

表 5.1 提案手法に対する要求と実装方法、説明箇所の対応 (再掲)

要求	提案手法での実現方法	備考	説明 (章)	検証 (章)
必要な知識をユーザーが認識しなくても獲得することができる	知識分子と結合アルゴリズム	ユーザーが知識を認知しなくても、システムに結合する知識が推論される	3.1	4.1
コンテキストを踏まえて様々な知識 (物理関係、パラメータ、不明確) が適切に再利用できる	知識分子の Condition Graph による結合判定	使用される部品、機能、属性とそれらの構造など、定性的な情報をもとに結びつける	3.2, 3.4	4.1
	知識分子の Analysis の結合による結合	パラメータをもとにした解析により知識を結びつける	3.2	4.3
	知識分子結合アルゴリズム (クエリ言語 or 類似度ベース)	類似度によって曖昧な原因の知識を再利用することができる	3.4	4.3
不足した情報の補完、複数の知識を組み合わせが可能である	Analysis によるパラメータの取り込み、解析	ユーザーに対して不足した情報を問い合わせる	3.2	4.3
	知識分子の Operator による複数知識による推論	複数種類の知識を組み合わせ、複雑な知識を再利用可能にする	3.2	4.2, 4.3
効率的に知識を再利用できる	知識分子評価手法	大量に結合する知識から、有用な知識を人間が識別できるようにする	3.5	4.2
	Excel ベースのインターフェース, 不具合知識定義、運用プロセス	知識やモデルを効率的に定義できる	3.7	4.4
大量のデータを集めることが難しい宇宙領域で適用できる	Condition Graph と知識分子結合アルゴリズム	機械学習のような大量の学習データは必要ない	3.1	4.1

5.3 提案手法の特徴

5.3.1 プロジェクト間の知識引継ぎ, 再利用

先行研究の Knowledge Based Systems のほとんどは、ある特定の問題を解決するために、それに役立つ知識を専門のエンジニアが一貫して、システムに入力しておくケースがほとんどである。一方で、本研究では、プロジェクトで出てきた知識をどう再利用するか、が目的であり、知識入力を一貫して行うことは難しい。さらに、多種多様な知識を入力しておく必要があり、知識自体の条件の不確定性、曖昧さも表現する必要がある。これらの課題に対して、解決する方法を、知識分子という形で提供している。

5.3.2 再利用性

本研究では、拡張デバイスオントロジーを用いた物理および機能ベースの知識のモデル化を行い、より再利用性の高い知識を実現している。拡張デバイスオントロジーを利用することで、別分野 (例えば船、自動車など) の知識も取り込むことが可能となる。一方で、すべての知識が拡張デバイスオントロジーで記述できるわけではないので、その部分については宇宙システムオントロジーを用いて記述できるようにしている。二つのオントロジーを組み合わせることで、知識の表現力と再利用性の両立を行っている。

5.3.3 柔軟性

前述のように、拡張デバイスオントロジーを用いると、知識の再利用性は向上する、一方で、物理的に説明することが難しい知識に関しては表現が困難になる（例えばある特定のコンフィギュレーションだと発生する不具合、ある状況だと起こりやすいヒューマンエラーに関する知識)。これに対して、別途宇宙システムオントロジーを構築することで、説明性を上げ、再利用性と柔軟性を上げている。さらに、知識分子の構造を用いた推論を実現することで、厳密には概念のクラスが異なったり、関係が違っていても知識を再利用することを可能にしている。また、保存したい知識は、定性的な概念の関係だけではない可能性がある。例えばある衛星のコンポにおいて、温度が一定以上だと発生する不具合減少があったとする。この場合、温度のパラメータを見て、知識を適用したいといった要望がある。それに対して、本研究では、解析と結合を一つの知識分子にまとめることで、人が解析のコンフィギュレーションを指定しなくても、自動的に解析のコンフィギュレーション (シミュレーターのインプット、アウトプット）を決定し、解析を実行することが可能である。

5.3.4　Analysis,Operator による創発

Operator により、知識分子は宇宙システムモデルに変化を与えることができる。例えば、モデルに存在するコンポーネントの内部詳細構造を、知識に基づき追加したり、システムの状態を更新することができる。これにより、成熟していない粗い粒度の宇宙システムモデルであっても、モデルを自動で補完し、細かい不具合リスクを予測することができる。また、複数の知識を組み合わせることで、過去に起きていない知識を予測することも可能であると考えられる。さらに、Analysis を組み合わせることで、多様な Operator 機能を実現することが可能である。例えば、Analysis はユーザーへのパラメータ入力や問い合わせを行うことができるので、一般的なコンポーネント構造の知識を Operator で与えるようにしつつ、例外の可能性も考慮し、結合の際は逐一ユーザーへ確認をとることも可能である。また、Analysis をシミュレータの要素として活用し、自動でシミュレーションを構築することも、可能であると考えられる。(Appendix D)

5.4　結果を踏まえた本研究の課題

5.4.0.1　膨大な知識の結合

本研究では、知識に Actionability, Unexpectedness などの評価を与えることで、膨大な知識の結合の中に重要な知識が埋もれることを防いでいる。一方で、本研究で提案する結合アルゴリズムは、知識一つ一つに対して結合判定を行うため、知識の数が増大すると、それに比例して実行時間も長くなる。また検索アルゴリズム SPARQL の性質や、類似度計算の関係上、知識の結合対象である宇宙システムモデルが大きくなる場合も計算時間が増大する。そのため、大きな宇宙システムモデルに対して、数多くの知識を結合しようとした場合、無視できない長さの計算時間が必要となり、提案手法のユーザービリティに影響を与える。これらの影響を緩和する方法として、知識分子の評価を知識結合前に実施し、ある評価以下の場合は結合を判定する知識分子のリストから除外する方法が考えられる。本研究で提案した知識結合の評価のうち、Unexpectedness, Interest Level に関しては、知識結合前に評価することが可能である。すなわち

- ユーザーの持っている知識に類似する知識 (Unexpectedness が低い知識) は除外する
- ユーザーの興味がない (Interest Level が低い) 知識

を事前に除外することが考えられる。また、知識に関するオントロジー (抽象、具体) を定義し、必要な知識の分野を指定することがあげられる。これは、認知していない重要な知識を見逃すリスクは増える一方で、関係ない可能性が高い知識は排除することが可能となる。

また、別の方法として、類似する知識を集約し、一般化した知識分子を生成することがあげられる。膨大な知識分子を人力で一般化するのは非常に難しいため、Appendix B で示すようなデータマイニングの手法等を用いて、機械的に一般化する方法が考えられる。ただし、物理的な根拠に基づかない一般化は、誤ったものであるリスクがあるため、常に反例に注意し、もしあればすぐに一般化をやり直すなどの管理が重要となる。（これによりかえって管理コストが増大するリスクはある）

5.4.0.2 推論のループ、多重結合

現状、知識分子は operator によって状態を更新し、それを元にさらに別の知識分子が結合する。複数の知識分子の operator が、相反する効果を持つ場合、これらが無限ループを形成する可能性がある。そのため、知識の結合する回数などに上限を設ける必要があり、長い知識の結合などが阻害されるリスクはあるまた、不具合の波及、原因の追及の場合、各知識分子がどのシステムモデルの要素に影響を与えたかを記録することで、関連する知識をツリー構造にすることは可能である。一方で、現状のアルゴリズムでは結合できる知識が全て結合してしまう。そのため、一つだけの波及をたどる、もしくは原因の候補を一つに絞って探索する、ということは出来ず、アルゴリズムの改良が必要である。

5.4.0.3 知識による誤った意思決定

知識分子を活用し、不具合リスクの整理やそれに対する対策を行う際に、100% 正しい (最適な) 知識をユーザーエンジニアに届けれられるわけではないという課題がある。これは専門家のレビューもそうであるが、例えば不具合に対する最適な対応策は、Condition Graph で記述されていないコンテキストの差、すなわち時代背景や、ステークホルダーとの関係などによって、変わってしまう。そのため、誤った意思決定をユーザーエンジニアに促してしまい、重大なミスを犯すリスクがないとは言えない。

ある程度の経験を持つユーザーエンジニアの場合、そのような情報も考慮し意思決定を行えるため、知識分子は見逃しを補完するシステムとして活用することが可能である。一方で素人は、得られる新しい知識が多いものの、どの知識が真に正しいのかを判別する能力がないため、前述のようなリスクは高いと考えられる。

考えられる対応策として、知識分子そのものに、使用後のレビューや評価を付けられる機能を付与することがあげられる。すなわち、e-commerce サイトの商品レビューのように、知識分子を活用した人間が、その知識分子の正しさや活用性についてレビューを付ける。知識に対する知識を蓄積することができ、注意すべき知識や誤った知識の場合は、注意喚起がつくことになる。これにより、素人のエンジニアであっても、レビューを見つつ安心できそうな知識分子から、見逃していた知識を取得することが可能になる。

また、根本的に誤った知識分子や、最適性が個人や状況で依存しそう (かつ Condition Graph

で表現できない Context に依存しそう) な知識分子の記述を防ぐように、知識分子を管理する人間を用意することも、対策としてあげられる。

5.4.0.4 知識の記述と自然言語処理、管理者の必要性

知識分子の記述は、オントロジーをベースに知識の結合条件をグラフ構造で表しつつ、内容を自然言語で記述する。このグラフ構造の記述は、どのようなオントロジーを使って表現するべきかという知識がエンジニアに必要なため、一定の修練が必要である。本研究では、この問題に対して、Excel ベースのインターフェースを用いてある程度簡易に入力できるシステムにした。これにより、プロジェクトの一部のエンジニアが訓練することで知識分子が活用できると考えられる。一方で、このどのように表現するべきか、という部分は、オントロジーで完全に規定することは難しい。知識の条件や範囲は曖昧であり、細かく多様なオントロジーを準備したとしても、完全に表現するにはできない。多様な表現は、知識分子の結合が起こりづらくなり、せっかく定義した知識が再利用されいない可能性がある。本研究ではこの問題に対して、類似度ベースでの結合判定を行えるようにすることで、曖昧さをあえて残したうえで、ユーザーの好みに合わせて厳密さを規定した推論を可能にする方法を提案した。しかしながら、類似度計算も、Ad-hoc に、結合を判定するスレッショルドを決める必要があり、それをどう決めるかについては、トライエラーが必要となってしまうという欠点が存在する。

別の解決方法として、知識記述に関して随時レビューを行う管理者を立てることがあげられる。知識記述のオントロジー、文法を整理したうえで、表現方法が経験と会わない場合は、修正を促す。そのような役割を担った管理者を立てることで、記述方法をできるだけ画一化することが可能になる。GitHub などのソフトウエア開発プラットフォームを活用し、レビューを実施することで、知識分子のバージョン管理や、知識を記述する際のノウハウを Web 上に蓄積することが可能となる。これは、宇宙システムを開発することで培う知識、ノウハウから 1 段階メタの、知識を蓄積するためのノウハウであり、それを身に着ける労力は実際に宇宙システムを開発するよりは少ないと考えられる。一方で、そのノウハウを蓄積することで、知識を継続的に管理、update できれば、非常に有効な知識利用、それ等を用いた効率的な宇宙システム開発が可能になると考えられる。

また、まったく別の方法として、画一的な知識の記述をあきらめ、ある程度雑に知識分子の条件を記述しつつ、すべて類似度で知識を再利用する方法があげられる。これは、ほどほどに正しい Condition Graph を自動生成できる自然言語処理と相性が良い (Appendix C で提案)。この場合、複数の知識を組み合わせた創発的な知識の利用は難しくなるものの、ユーザーが見逃した知識を提供するだけであれば十分可能であり、それに必要な労力を抑えられるので有効であると考えられる。(ユーザーは、宇宙システムのモデルを作るだけで、あとはこれまでと同じく、自然言語で知識を記述するだけでよい)

5.4.0.5 オントロジーの増大

知識分子をより活用していく場合、多様な知識を表現するためにオントロジーを追加していくことは必要不可欠である。現状、宇宙システムオントロジーには、試験系に関するオントロジーが定義されていないが、試験系と宇宙システムのインターフェースで起こる不具合は非常に多いため、試験系を表現するオントロジーは今後実装していく必要があると考えられる。このようにオントロジーを追加していくと、知識分子と同様、同じような表現が増えてしまい、オントロジーにおける概念のうち、どれをどのように使用すべきかわからなくなる危険性がある。

単純な対応策として、知識分子と同様、管理、レビューが可能なオントロジーの管理人を立てることがあげられる。そのほかの方法としては、宇宙システムで使用される単語からオントロジーを自動で構築する方法がある。近年、自然言語処理を活用し、大量の文章から、オントロジーを構築、補完できるようにする手法が提案されており、それを利用することが考えられる [54]。ただし、宇宙システム開発はやはりデータ数が少ないので、適切なオントロジーが構築するには何かしら工夫が必要であると考えられる。

また、すでに公開されているオントロジーを取り入れることも一つの手段であると考えられる。新たに独自に開発するよりも、様々な考慮の元構築されているオントロジーを利用することで、信頼性の高い運用が可能である。宇宙システムの場合、[55] や [56] などで、オントロジーが提案されている。

5.4.0.6 モデルの記述

宇宙システムモデルについても、どのように表現するべきかという曖昧さはあるとともに、全体システムモデルを構築するのは非常に労力が必要であるという課題が存在する。本研究では、宇宙システムのコンポーネントやモジュールなど一部分をモデル化するだけでも、それがエンジニアの設計開発対象であれば効果を発揮する一方で、それでも詳細な知識を推論するには、モデル化の労力を必要とする。衛星の設計書をモデルとして、自然言語処理でマッチングをとる先行研究があるが、設計書の記述のみで完全なモデルを構築することは容易ではなく、教師データも必要となるうえで正確性も低い。オントロジーで簡単にモデルを構築するための方法として、エクセルのインターフェースを本研究では構築したが、さらなる改良が、実応用には求められると考えれられる。考えられる方法として、構造 CAD モデルや回路設計の PCB ファイルを利用する方法があげられる。これらの外部ファイルは衛星設計開発に必要不可欠であり、知識利用のために追加で構築する必要がなくなる。また宇宙システム設計そのものを SysML などの言語に基づき、モデルで表現する場合はそれらを取り込むことも有効な手段であると考えられる。

第 6 章

結論

本研究では、宇宙システムプロジェクトにおける不具合、スケジュール、コスト遅延の解決に向け、特にエンジニアの不具合に関する知識不足に関する問題の解決方法を提案した。一般的なデータベース蓄積型の知識では、エンジニアは知識を認知しなければ、必要な知識を取得出来ず、またコンテキストに対して適用性を自ら判断する必要がある。また、人を使ったレビューの場合、属人性やコミュニケーションコストが膨大になってしまう。先行研究では、知識を保存し利用するルールベースの故障推論があるが、知識の記述を厳密に実施する必要があり、条件が一意に特定できないような経験的な知識を再利用することは難しい。また、モデルを構成する要素やそれらの関係を条件とする知識と、モデルに含まれるパラメータの定量的な値に対する知識を区別して、推論を実施する必要があった。そこで本研究では、知識分子と呼ばれる知識の保存方法を記述し、そこに知識がモデルに結合する条件を記述することで、知識を提供するフレームワークを提案した。知識分子は、オントロジーで記述したグラフ構造を結合条件として持ち、それをもとにモデルと結合を行う。さらに定量的な解析を実施して結合を評価することができる。グラフ構造を用いて、類似度をベースに、知識当てはまる設計箇所を探索することで、原因が不明確な不具合に関する知識であっても、適切に再利用し、新しいシステム設計の不具合リスクを指摘することが可能である。このような柔軟性を持つフレームワークを利用することで知識の記述できる幅や評価を広げ、様々なコンテキストを表現しつつ、ユーザーの認知していない知識を提供できることが可能になる。提案した手法は実際の衛星開発プロジェクトに適用し、その応用性を確かめるとともに、ユーザーへのインタビューを行いその実用性を確認した。一方で、これらの実際の利用におけるユーザービリティ、特に記述を一貫して保つこと、もしくは大量の知識を効率的に対処するには問題が残っている。これらの課題解決には、オントロジーの追加や、知識分子自体のレビュープロセスなどに関する、さらなる研究が必要である。

謝辞

本研究に取り組み、博士論文をまとめるまでには、多くの方々のご支援とご指導を賜りました。博士論文を上梓するにあたり、お世話になった方々に、この場をお借りして感謝の意を申し上げます。

はじめに、指導教官である中須賀真一先生には、筆者が学部 4 年生のころから、熱心にかつ丁寧にご指導をしてくださり、大変感謝をしております。博士三年では、ほぼ毎月先生と相談させていただき、研究の方向性や価値の出し方、宇宙機開発におけるプロジェクトの問題点など、多様な議論を深くさせていただき、大変貴重な時間をいただくことができたと思っております。研究の方向性が見えた時の先生の励ましの言葉はとても勇気づけられ、筆者は終始前向きに研究に取り組むことができたと思っております。また、研究以外においても、先生との議論を通して、宇宙開発の課題を見つけることができ、これらの課題を解決して宇宙機開発を発展させようという志を持つことができました。先生の紹介で、国内外の研究者の方と一緒にプロジェクトや議論をする機会も頂くことができ、それらも筆者にとって非常に貴重な機会だったと思います。そして、失敗を恐れず、失敗しても泥臭く前に進むことの重要さを教えていただくことができました。本当にありがとうございました。

また、副指導教官である船瀬龍先生には、実際の EQUULEUS, ISSL6U など、実際の CubeSat 開発を通して、どうすれば宇宙機が作れるのかをご指導いただきました。船瀬先生の本質を見抜くようなアドバイスには、実際に研究やプロジェクトの課題解決を行うのに助かりました。実は、本研究のベースとなるモチベーションである、「レビュープロセスを効率的にしたい」というのは、EQUULEUS の Critical Design Review が終わった後に、船瀬先生とレビュープロセスについて議論させてもらった時に、自分の中に根付いたものです。このような課題意識を見つける機会をいただき、本当にありがとうございました。

また、研究室の先輩であり、研究員の小畑俊裕さんには、筆者を Systems Engineering の研究領域に導いてくださり大変感謝しております。EQUULEUS の開発で不具合が頻発し、宇宙機開発の難しさに悶々としていた際、小畑さんが研究室でしてくださった、Systems Engineering に関する発表には衝撃を受けたのを覚えています。今の研究室に足りていないのは Systems Engineering に関する知識や研究だと思え、この分野を研究してみたいと思うことができまし

た。また、その後も定期的に研究の相談にのってくださり、実務経験を踏まえてコメントから些細なロジックのずれまで、たくさんのアドバイスをいただけました。本当に感謝しております。

また、副査の先生方にも、博論審査以外でも多くミーティングをしていただき大変感謝しております。

堀浩一先生には知識の活用において、基本的な Knowledge Based Systems の傾向や、自然言語処理の動向、参考になる論文など、人工知能の面で多くのアドバイスを頂けました。本当にありがとうございました。またまだ構想段階だった博士論文の内容について、面白そうだと励ます言葉をいただき大変勇気づけられました。

矢入健久先生には、提案手法を実際に応用する面で、難しい点とどうすればいいかに関するアイデアをいただけました。知識分子をレポジトリごとに分けて開発し、必要に応じて共有したり、異分野でも活用できるように工夫を考える必要性など、大変役立つアドバイスをいただきありがとうございました。

また、稗方先生には、より深く Systems Engineering を理解するきっかけをいただけたこと、大変感謝致します。稗方先生の授業を受け、マサチューセッツ工科大学に留学させてもらい、そこで Systems Engineering や Project Management に関する深い知識を学ぶ機会をいただけたのは、筆者にとって非常に幸運だったと思います。本当にありがとうございました。

また、MIT で直接ご指導をいただいた Bryan Moser 先生にも感謝申し上げます。Bryan 先生のおかげで、Systems Approach を深く理解し、身に着けることができたと考えております。Systems Engineering において固定されたプロセスが重要ではなく、目的に合わせて適切にシステムをとらえ理解することが重要であることに気づけたと思います。本当にありがとうございました。

また、NASA Jet Propulsion Laboratory でインターンのメンターを指導してくださった小野雅弘さんにも感謝を申し上げます。小野さんのおかげで、世界最先端の宇宙開発を肌で感じることができるとともに、自分のキャリアパスにおいてやりたいことを明確化できたと感じております。本当にありがとうございました。

また、本研究で行った検証項目の一つである、インタビューを受けってくださった、航空宇宙工学専攻研究員の野村俊一郎さん、博士課程の船曳敦漠さん、修士課程の唐川大輝さんにも感謝申し上げます。三名の協力がなければ、提案手法の効果を十分確認するのは難しかったと思います。特に、野村さんには Systems Engineering の研究グループをマネジメントしていただき、研究室でのこの分野の発展に協力いただけたこと、また、EQUULEUS では姿勢系開発をリードしていただいたこと、大変感謝しております。

また、その他これまで中須賀船瀬研で、一緒にプロジェクトを進めてきた優秀な同期、先輩方、後輩にも感謝申し上げます。輪講や研究グループのディスカッション、プロジェクトを通して、多くの議論をさせていただき、大変に良い成長の機会をいただくことができました。

また、筆者が共同創業者として参画しているアークエッジ・スペ―スの社員、同期の方々にも感謝申し上げたいと思います。特に博士論文提出付近で実施していた概念検討をリードしてくれた柿原浩太さん、渋川季裕さんには、博士論文に集中する時間を作ってもらい助かりました。ありがとうございました。

また、本研究の一部は、日本学術振興会科学研究費 (特別研究員 No. 20J12002)、および社会構想マネジメントを先導するグローバルリーダー養成プログラム、およびリーダー博士育成基金に助成を受けて実施されました。これらの助成金により研究を円滑に実施できたこと、大変感謝しております。

そして、筆者をこれまで温かく応援してくれた母 髙橋和恵、父 髙橋浩一、子供の頃から仲良くしてくれた姉 髙橋明日香、弟 髙橋周平、そして祖父母や親族の方々には大変感謝しております。幼いころから、家族には支えてもらい、そのおかげで最終的にこの博士論文を提出することができたと感じております。本当にありがとうございました。

最後に、大学以降ずっと自分の生活、精神面を支えてくれた妻 髙橋沙耶に、心より感謝を申し上げたいと思います。妻の研究に対する努力の姿勢を見習い、最後まで頑張ることができたと感じております。本当にありがとうございました。

2022 年 2 月 9 日

参考文献

[1] O. of Inspector General, "NASA' S MANAGEMENT OF SPACE LAUNCH SYSTEM PROGRAM COSTS AND CONTRACTS," 2020.

[2] 日経クロステック（xTECH）, "三菱スペースジェット事業凍結、型式証明に泣いた 12 年目の結末." [Online]. Available: https://xtech.nikkei.com/atcl/nxt/column/18/00001/04760/

[3] K. O' Donnell and G. Richardson, "Small satellite trending & reliability 2009-2018," 2020.

[4] P. D. Collopy and P. M. Hollingsworth, "Value-driven design," *Journal of aircraft*, vol. 48, no. 3, pp. 749–759, 2011.

[5] O. De Weck, C. M. Eckert, P. J. Clarkson, *et al.*, "A classification of uncertainty for early product and system design," in *DS 42: Proceedings of ICED 2007, the 16th International Conference on Engineering Design, Paris, France, 28.-31.07. 2007*, 2007, pp. 159–160.

[6] W. Majerowicz and S. A. Shinn, "Schedule matters: Understanding the relationship between schedule delays and costs on overruns," in *2016 IEEE Aerospace Conference*. IEEE, 2016, pp. 1–8.

[7] G. F. Dubos, J. H. Saleh, and R. Braun, "Technology readiness level, schedule risk, and slippage in spacecraft design," *Journal of Spacecraft and Rockets*, vol. 45, no. 4, pp. 836–842, 2008.

[8] L. J. Paxton, " "faster, better, and cheaper" at nasa: Lessons learned in managing and accepting risk," *Acta Astronautica*, vol. 61, no. 10, pp. 954–963, 2007.

[9] C. Venturini, B. Braun, D. Hinkley, and G. Berg, "Improving mission success of cubesats," 2018.

[10] C. Lange, J. T. Grundmann, M. Kretzenbacher, and P. M. Fischer, "Systematic reuse and platforming: Application examples for enhancing reuse with model-based systems engineering methods in space systems development," *Concurrent Engineering*, vol. 26, no. 1, pp. 77–92, 2018.

[11] D. Meza, "How nasa finds critical data through a knowledge graph," Oct 2018. [Online]. Available: https://neo4j.com/blog/nasa-critical-data-knowledge-graph/

[12] "NASA SSVI," Nov 2021. [Online]. Available: (https://www.nasa.gov/smallsat-institute/about-us

[13] S. J. Kapurch, *NASA systems engineering handbook*. Diane Publishing, 2010.

[14] J. A. Estefan *et al.*, "Survey of model-based systems engineering (mbse) methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[15] "Jmr/jerg," Nov 2021. [Online]. Available: https://sma.jaxa.jp/TechDoc/

[16] S. Nakasuka, N. Sako, H. Sahara, Y. Nakamura, T. Eishima, and M. Komatsu, "Evolution from education to practical use in university of tokyo's nano-satellite activities," *Acta Astronautica*, vol. 66, no. 7-8, pp. 1099–1105, 2010.

[17] R. Bocchino, T. Canham, G. Watney, L. Reder, and J. Levison, "F prime: an open-source framework for small-scale flight software systems," 2018.

[18] K. Jenks, "Nasa product peer review process," 2009.

[19] I. Nonaka and T. Nishiguchi, *Knowledge emergence: Social, technical, and evolutionary dimensions of knowledge creation*. Oxford University Press, 2001.

[20] R. V. Ramasesh and T. R. Browning, "A conceptual framework for tackling knowable unknown unknowns in project management," *Journal of operations management*, vol. 32, no. 4, pp. 190–204, 2014.

[21] 岡啓 and 当麻哲哉, "Opm と scorecarding 法の組合せによるシステム開発手法の評価," in 情報システム学会全国大会論文集 第 *5* 回全国大会・研究発表大会論文集. 一般社団法人情報システム学会, 2009, pp. A1–2.

[22] 福田哲志, 久住憲嗣, 福田晃, *et al.*, "Sysml を用いたシステム開発における制約の充足可能性検証," 研究報告システム *LSI* 設計技術 *(SLDM)*, vol. 2012, no. 12, pp. 1–6, 2012.

[23] D. Wagner, S. Y. Kim-Castet, A. Jimenez, M. Elaasar, N. Rouquette, and S. Jenkins, "Caesar model-based approach to harness design," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–13.

[24] A. Austin, R. Miller, J. Murphy, M. Kolar, J. Blossom, T.-A. Phan, S. Doudrick, K. Hogstrom, and A. Marinan, "Rapid smallsat mission formulation: Integrated and concurrent modeling in jpl's team xc," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–11.

[25] E. J. Reddy, C. Sridhar, and V. P. Rangadu, "Knowledge based engineering: notion, approaches and future trends," *American Journal of Intelligent Systems*, vol. 5, no. 1, pp. 1–17, 2015.

[26] W. Skarka, "Application of moka methodology in generative model creation using catia," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 5, pp. 677–690, 2007.

[27] Y.-H. Wu and H.-J. Shaw, "Document based knowledge base engineering method for ship

basic design," *Ocean Engineering*, vol. 38, no. 13, pp. 1508–1521, 2011.

[28] R. Arendt and E. Van Uden, "A decision-making module for aiding ship system automation design: A knowledge-based approach," *Expert Systems with Applications*, vol. 38, no. 1, pp. 410–416, 2011.

[29] C. L. Emberey, N. Milton, J. Berends, M. Van Tooren, S. Van der Elst, and B. Vermeulen, "Application of knowledge engineering methodologies to support engineering design application development in aerospace," in *7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov and Integr in Aero Sciences, 17th LTA Systems Tech Conf; followed by 2nd TEOS Forum*, 2007, p. 7708.

[30] G. La Rocca, L. Krakers, and M. van Tooren, "Development of an icad generative model for blended wing-body aircraft design," in *9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, 2002, p. 5447.

[31] S. Hossain, D. Sarma, R. J. Chakma, W. Alam, M. M. Hoque, and I. H. Sarker, "A rule-based expert system to assess coronary artery disease under uncertainty," in *International Conference on Computing Science, Communication and Security*. Springer, 2020, pp. 143–159.

[32] D. Grassian, M. Bahatem, T. Scott, and D. Olsen, "Application of a fuzzy expert system to analyze and anticipate esp failure modes," in *Abu Dhabi International Petroleum Exhibition & Conference*. OnePetro, 2017.

[33] F. Arévalo, C. Tito, M. R. Diprasetya, and A. Schwung, "Fault detection assessment using an extended fmea and a rule-based expert system," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 740–745.

[34] 井戸大成 and 來村徳信, "産業機械の仕様調整のための知識モデルとオントロジーの構築," in 人工知能学会全国大会論文集 第 *32* 回全国大会 *(2018)*. 一般社団法人 人工知能学会, 2018, pp. 2H104–2H104.

[35] 稗方和夫, 大和裕幸, 辻本翔, *et al.*, "オントロジーを用いた製造現場の不具合情報検索手法に関する研究," *SIG-KST*, vol. 5, no. 02, pp. 1–6, 2008.

[36] 來村徳信, 西原稔人, 植田正彦, 池田満, 小堀聡, 角所収, and 溝口理一郎, "故障オントロジーの考察に基づく故障診断方式," 人工知能学会誌, vol. 14, no. 5.

[37] J. C. R. Bejarano, T. Coudert, E. Vareilles, L. Geneste, M. Aldanondo, and J. Abeille, "Case-based reasoning and system design: An integrated approach based on ontology and preference modeling," *AI EDAM*, vol. 28, no. 1, pp. 49–69, 2014.

[38] Y. Zhang, X. Liu, J. Jia, and X. Luo, "Knowledge representation framework combining case-based reasoning with knowledge graphs for product design," *Comput.-Aided Des. Appl.*, vol. 17, no. 4, pp. 763–782, 2019.

[39] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI communications*, vol. 7, no. 1, pp. 39–59, 1994.

[40] D. Das, L. Sahoo, and S. Datta, "A survey on recommendation system," *International Journal of Computer Applications*, vol. 160, no. 7, 2017.

[41] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," *arXiv preprint arXiv:1910.12892*, 2019.

[42] X. L. Dong, X. He, A. Kan, X. Li, Y. Liang, J. Ma, Y. E. Xu, C. Zhang, T. Zhao, G. Blanco Saldana, *et al.*, "Autoknow: Self-driving knowledge collection for products of thousands of types," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2724–2734.

[43] 古崎晃司, 來村徳信, 佐野年伸, 本松慎一郎, 石川誠一, and 溝口理一郎, "オントロジー構築・利用環境 「法造」 の開発と利用," 人工知能学会論文誌, vol. 17, no. 4, pp. 407–419, 2002.

[44] 來村徳信 and 溝口理一郎, "オントロジー工学に基づく機能的知識体系化の枠組み," 人工知能学会論文誌, vol. 17, no. 1, pp. 61–72, 2002.

[45] 川井翼 and 堀浩一, "人工衛星の設計支援のためのデバイスオントロジーの拡張," in 人工知能学会全国大会論文集 第 *29* 回全国大会 *(2015)*. 一般社団法人 人工知能学会, 2015, pp. 2M11–2M11.

[46] O. Lassila, "With the resource description framework," 1999.

[47] E. Prud' hommeaux and A. Seaborne, "Sparql query language for rdf. w3c working draft," *World Wide Web Consortium, February. Latest version: http://www. w3. org/TR/rdf-sparql-query*, 2006.

[48] D. Zhang, T. Song, J. He, X. Shi, and Y. Dong, "A similarity-oriented rdf graph matching algorithm for ranking linked data," in *2012 IEEE 12th International Conference on Computer and Information Technology*. IEEE, 2012, pp. 427–434.

[49] F. Hussain, H. Liu, and H. Lu, "Relative measure for mining interesting rules," in *PKDD' 00, Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*. Citeseer, 2000, pp. 117–132.

[50] A. S. Al-Hegami, "Subjective measures and their role in data mining process," in *Proceedings of the 6th International Conference on Cognitive Systems, New Delhi, India*. Citeseer, 2004.

[51] T. R. Amer, J. N. Ortiz, M. A. Calloway, R. M. Greathouse, C. A. Polen, R. L. Baker, H. E. Borchardt, C. C. Chromik, E. Moran, M. W. Paraska, *et al.*, "Nasa standing review board handbook. rev a," 2014.

[52] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[53] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*. PMLR, 2015, pp. 957–966.

[54] A. Konys and Z. Drażek, "Ontology learning approaches to provide domain-specific knowledge base," *Procedia Computer Science*, vol. 176, pp. 3324–3334, 2020.

[55] D. A. Wagner, M. B. Bennett, R. Karban, N. Rouquette, S. Jenkins, and M. Ingham, "An ontology for state analysis: Formalizing the mapping to sysml," in *2012 IEEE Aerospace Conference*. IEEE, 2012, pp. 1–16.

[56] D. Bouquin, P. Papadeas, D. Chivvis, A. Williams, V. Tsiligiannis, F. Damkalis, and K. Frey, "Describing smallsat missions with metasat," 2020.

[57] S. Arora, "A survey on graph neural networks for knowledge graph completion," *arXiv preprint arXiv:2007.12374*, 2020.

[58] S. Wu, F. Sun, W. Zhang, and B. Cui, "Graph neural networks in recommender systems: a survey," *arXiv preprint arXiv:2011.02260*, 2020.

[59] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[60] 猪口明博, 鷲尾隆, and 元田浩, "Apriori-based graph mining アルゴリズムの効率化," 人工知能学会全国大会論文集, no. 0, pp. 133–133, 2001.

[61] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE, 2002, pp. 721–724.

[62] 三好裕樹, 尾崎知伸, 江口浩二, and 大川剛直, "定量的アイテム集合付き単一グラフからの頻出パターンマイニング," 人工知能学会論文誌, vol. 26, no. 1, pp. 284–296, 2011.

[63] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh, "A survey on open information extraction," *arXiv preprint arXiv:1806.05599*, 2018.

[64] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proceedings of the 2011 conference on empirical methods in natural language processing*, 2011, pp. 1535–1545.

[65] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld, "Open information extraction from the web," *Communications of the ACM*, vol. 51, no. 12, pp. 68–74, 2008.

[66] F. Wu and D. S. Weld, "Open information extraction using wikipedia," in *Proceedings of the 48th annual meeting of the association for computational linguistics*, 2010, pp. 118–127.

[67] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

付録 A

EQUULEUS 不具合から生成した知識分子一覧

本章に、3.5 で定義した、EQUULEUS の不具合に関する知識分子の一覧を示す。なお、以下の形で表を生成している。

- 画像化の労力がかかるため、Condition Graph は、triples (S, P, O = ノードラベル, エッジラベル、ノードラベル) の 3 つの形で表現している。
- ?hoge (hoge は任意の文字) は、インスタンスを表し、ont:hoge は、ontology で定義された概念を表す。
- Content は、辞書形式で保存し、知識の分野を複数保存している。

EQUULEUS の不具合知識分子は下記のとおりである。

表 A.1: ルールベースでの衛星 2 への知識再利用結果

Condition	**Analysis**	**Operator**	**Content**
?meisei_obc ont:hasOutput ?information. ?correctness rdf:type ont:Correctness. ?information rdf:type ont:Electrical_Signal. ?information ont:hasParameter ?correctness. ?meisei_obc rdf:type ont:Obc_Meisei.		?correctness ont:hasStatus 1.	{”cause”:”原因特定には至っていない”, ” failure”:”OBC の UART から信号が出力されない”, ”check”:”試験中に信号波形が出なくなることがないかを確認する”, ”example”:”EQU FM 不具合表 ID 2”}

?power_control_unit ont:hasOutput ?electricity. ?electricity ont:hasParameter ?voltage. ?power_control_unit rdf:type ont:Power_Control_Unit. ?electricity rdf:type ont:Electrical_Power. ?voltage rdf:type ont:Voltage.		?voltage ont:hasStatus -1.	{”cause”:”PCU BUS 電圧が低いとき、UVC の閾値をどこまで変動させないかがキチンと会話されていない”, ”failure”:”意図しない UVC”, ” check”:”BUS 電圧低下時に突入電流などで UVC にかからないこと”, ”example”:”EQU FM 不具合表 11”}
?communication_device ont:hasParameter ?gain. ?communication_device rdf:type ?t. ?t rdfs:subClassOf* ont:Communication_Device. ?gain rdf:type ont:Gain.		?gain ont:hasStatus 1.	{”cause”:”通信系に内部オペアンプの素子取付位置を間違える”, ”failure”:”出力ゲインが低下する”, ”check”:”取付位置にミスはないか、ゲインが設計から下がっていないか”, ”example”:”EQU 不具合 FM 12”}
?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Onboard_Computer. ?component ont:hasOutput ?information. ?information ont:hasParameter ?correctness. ?information rdf:type ont:Electrical_Signal. ?correctness rdf:type ont:Correctness.		?correctness ont:hasStatus 1.	{”cause”:”Driver の SW バグ”, ”failure”:”テレメトリガ降りてこない”, ”test_check”:”Driver が正しく、テレメトリが正しいことを試験で確認する”, ”example”:”EQU FM 不具合表 14”}
?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Conduit. ?component ont:hasParameter ?shape. ?shape rdf:type ont:Shape.		?shape ont:hasStatus 1.	{”cause”:”ハーネスやケーブルを抜き差し”, ”failure”:”ケーブルの損傷、劣化”, ”test_check”:”打ち上げ前に、ハーネスやケーブルの劣化がないかを確認する”, ”example”:”EQUULEUS FM 不具合 16”}

?harness ont:include ?mdmconnecter. ?harness ont:transfer ?electricity. ?electricity ont:hasParameter ?correctness. ?harness rdf:type ont:Harness. ?electricity rdf:type ont:Electrical_Power. ?mdmconnecter rdf:type ont:MDMConnector. ?correctness rdf:type ont:Correctness.		?correctness ont:hasStatus 1.	{”cause”:”MDM コネクタの極性を変えた際のピンアサ変化に気づかず設計する”, ”failure”:”ピンアサが意図と異なる”, ”design_check”:”MDM コネクタの極性とピンアサインを確認する”, ”example”:”EQU FM 不具合表 id 21”}
?kill_switch ont:hasInput ?electricity. ?kill_switch ont:hasOutput ?information. ?information ont:hasParameter ?correctness. ?kill_switch rdf:type ont:Kill_Switch. ?switch rdf:type ont:Switch. ?information rdf:type ont:Electrical_Signal. ?electricity rdf:type ont:Electrical_Power. ?correctness rdf:type ont:Correctness.		?correctness ont:hasStatus 1.	{”cause”:”電圧差を見てスイッチを判定する場合の、電圧差の見積もりが甘い”, ”failure”:”誤った起動が起こる”, ”design_check”:”キルスイッチの判定に使う電圧差条件は十分か確認する”, ” example”:”EQU FM 不具合表 id 22”}
?power_control_unit ont:hasOutput ?electricity. ?electricity ont:hasParameter ?voltage. ?power_control_unit rdf:type ont:Power_Control_Unit. ?electricity rdf:type ont:Electrical_Power. ?voltage rdf:type ont:Voltage.		?voltage ont:hasStatus 1.	{”cause”:”PCU 内部の抵抗地誤差”, ”failure”:”HW UVC が意図しない範囲で発生する”, ”check”:”HW UVC の閾値は抵抗地の誤差で変動するか、変動する場合それが許容できるか”, ”example”:”EQU FM 不具合表 id 23”}
?power_control_unit ont:hasOutput ?electricity. ?reaction_wheel ont:hasInput ?electricity. ?power_control_unit ont:hasOutput ?electricity2. ?electricity2 ont:hasParameter ?voltage. ?electricity2 rdf:type ont:Electrical_Power. ?power_control_unit rdf:type ont:Power_Control_Unit. ?reaction_wheel rdf:type ont:Reaction_Wheel. ?electricity rdf:type ont:Electrical_Power. ?voltage rdf:type ont:Voltage.		?voltage ont:hasStatus 1.	{”cause”:”RW の回転突入電流で、バス電圧が降下する”, ”failure”:”バス電圧が低下する”, ”check”:”大きな突入電流が発生する際に、ほかの機器が再起動にかからないか”, ”example”:”EQU FM 不具合表 id 24”}

?resister ont:hasInput ?electricity. ?electricity ont:hasParameter ?current. ?resister ont:hasParameter ?resistance_value. ?resister rdf:type ont:Resister. ?electricity rdf:type ont:Electrical_Power. ?current rdf:type ont:Current. ?resistance_value rdf:type ont:Resistant_Value.		?current ont:hasStatus 1.	{”cause”:”抵抗値の意図しない値”, ”failure”:”大きな電流が流れる”, ”check”:”抵抗値を変えても問題ないか”, ”example”:”EQU FM 不具合表 id 26”}
?reaction_wheel ont:hasInput ?electricity. ?electricity ont:hasParameter ?voltage. ?reaction_wheel rdf:type ont:Reaction_Wheel. ?electricity rdf:type ont:Electrical_Power. ?voltage rdf:type ont:Voltage.		?voltage ont:hasValue -1.	{”cause”:”RW をこーステぃングさせ逆起電力が発生する”, ”failure”:”RW の電源状態 OFF がラッチされる”, ”check”:”逆起電力発生時に RW を再起動させたい要求があるかどうかを確認し、必要なら仕様を詰める”, ”example”:”EQU FM 不具合表 id 28”}
?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Conduit. ?component ont:transfer ?electricity. ?electricity ont:hasParameter ?voltage. ?electricity ont:hasParameter ?current. ?electricity rdf:type ont:Electrical_Power. ?voltage rdf:type ont:Voltage. ?current rdf:type ont:Current.		?current ont:hasStatus 1. ?voltage ont:hasStatus -1.	{”cause”:”はんだ付けのミス”, ”failure”:”導通が起こる”, ”check”:”ピン間抵抗、はんだ付けミスにより電圧に誤差がないかを確認する”, ”example”:”EQU FM 不具合 29”}
?star_tracker ont:hasInput ?electricity. ?electricity ont:hasParameter ?noise. ?star_tracker ont:hasOutput ?information. ?information ont:hasParameter ?correctness. ?star_tracker rdf:type ont:Star_Tracker. ?electricity rdf:type ont:Electrical_Power. ?information rdf:type ont:Electrical_Signal. ?noise rdf:type ont:Noise. ?correctness rdf:type ont:Correctness.	?noise ont:hasStatus 1.	?correctness ont:hasStatus 1.	{”cause”:”STT 電源ラインにノイズが乗る”, ”failure”:”STT 画像撮影がノイズの影響を受ける”, ”test_check”:”STT への電源ラインに許容できるレベルのノイズが入っていることを確認する、STT 画像をくみ上げ状態で確認する”, ”example”:”EQU FM 不具合表 id 36”}

?compo ont:hasParameter ?shape. ?compo ont:include ?moving_part. ?compo ont:screw_connect ?compo2. ?shape rdf:type ont:Shape. ?moving_part rdf:type ont:Moving_Part.		?shape ont:hasStatus 1.	{”cause”:”ねじ止めを中途半端に行う”, ”failure”:”回転部材が回転できなくなる、形状がゆがむ”, ”test_check”:”ねじ止めが十分になされているか確認する”, ” example”:”EQU FM 不具合表 id 44”}
?mli ont:near ?component. ?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Component. ?component ont:include ?electrical_line. ?electrical_line ont:transfer ?information. ?information ont:hasParameter ?noise. ?mli rdf:type ont:Mli. ?electrical_line rdf:type ont:Line. ?information rdf:type ont:Electrical_Signal. ?noise rdf:type ont:Noise.		?noise ont:hasStatus 1.	{”cause”:”MLI の導電面が機器の配線に短絡する”, ”failure”:”信号線にノイズが発生する”, ”design_check”:”MLI の導電面の接触の有無と接触する場合問題ないかを確認する”, ” example”:”EQU FM 不具合表 id 47”}
?screw ont:hasParameter ?shape. ?screw rdf:type ont:Screw. ?shape rdf:type ont:Shape.		?shape ont:hasStatus 1.	{”cause”:”ねじどめの際のワッシャ有無のコミュニケーションを行わなかった”, ”failure”:”ねじの厚みで包絡域を逸脱する”, ”design_check”:”ねじ止めの際にワッシャの有無を指定する”, ”example”:”EQU FM 不具合表 id 61”}
?harness ont:hasParameter ?shape. ?harness ont:transfer ?information. ?information ont:hasParameter ?correctness. ?harness rdf:type ont:Harness. ?information rdf:type ont:Electrical_Signal. ?shape rdf:type ont:Shape. ?correctness rdf:type ont:Correctness.	?shape ont:hasStatus 1.	?correctness ont:hasStatus 1.	{”cause”:”ハーネスの損傷、断線”, ”failure”:”信号が途絶える”, ”test_check”:”ハーネスの断線が起きていないかを確認する”, ”example”:”EQU FM 不具合表 id 64”}

?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Battery.			{”cause”:”バッテリーの充電完了電圧と開始電圧が近く、充電完了の瞬間に電圧降下で充電開始に入ってしまう”, ”failure”:”バッテリーの充電開始、完了が繰り返してしまう”, ”design_check”:”充電完了電圧と開始電圧が十分離れているか”, ”example”:”EQU FM 不具合表 id 65”}
?power_control_unit ont:hasInput ?electricity. ?electricity ont:hasParameter ?noise. ?power_control_unit ont:hasInput ?electricity2. ?electricity2 ont:hasParameter ?current. ?electricity2 rdf:type ont:Electrical_Power. ?power_control_unit rdf:type ont:Power_Control_Unit. ?electricity rdf:type ont:Electrical_Power. ?noise rdf:type ont:Noise. ?current rdf:type ont:Current.	?noise ont:hasStatus 1.	?current ont:hasStatus -1.	{”cause”:”PCU リセットラインにノイズが入る”, ”failure”:”意図しない電源リセット”, ”test_check”:”PCU リセットラインにノイズが入らないかを確認”, ”example”:”EQU FM 不具合表 id 69”}
?onboard_computer ont:hasOutput ?information. ?information ont:hasParameter ?noise. ?onboard_computer rdf:type ont:Onboard_Computer. ?information rdf:type ont:Electrical_Signal. ?noise rdf:type ont:Noise.		?noise ont:hasStatus 1.	{”cause”:”OBC が WDT で落ちる”, ”failure”:”OBC が出力する信号線にノイズが乗る”, ”test_check”:”信号線にノイズが乗るか、乗る場合それでリセットピンなどのノイズが乗ることはないか”, ”example”:”EQU FM 不具合表 id 69”}
?harness ont:transfer ?electricity. ?electricity ont:hasParameter ?current. ?harness rdf:type ont:Harness. ?electricity rdf:type ont:Electrical_Power. ?current rdf:type ont:Current.		?current ont:hasStatus 1.	{”cause”:”ハーネスがフライト品に変わったことによる配線抵抗の低下”, ”failure”:”電流値が想定以上になる”, ”test_check”:”EM 品、フライト品に抵抗値に変化はないか”, ”example”:”EQU FM 不具合表 id 80”}

?thermometer ont:hasOutput ?electricity. ?electricity ont:hasParameter ?current. ?thermometer rdf:type ont:Thermometer. ?electricity rdf:type ont:Electrical_Power. ?current rdf:type ont:Current.		?current ont:hasStatus -1.	{”cause”:”温度計貼り付け時にはんだ部を折り曲げる”, ”failure”:”温度計がはがれる”, ”test_check”:”くみ上げ後温度計が, くみ上げ時の力などではがれていないかを確認する”, ” example”:”EQU FM 不具合表 id 83”}
?reaction_wheel ont:hasOutput ?mechanical_torque. ?mechanical_torque ont:hasParameter ?torque. ?reaction_wheel ont:hasInput ?electricity. ?electricity ont:hasParameter ?current. ?reaction_wheel rdf:type ont:Reaction_Wheel. ?electricity rdf:type ont:Electrical_Power. ?torque rdf:type ont:Torque. ?current rdf:type ont:Current.	?toruqe ont:hasStatus 1.	?current ont:hasStatus 1.	{”cause”:”RW の想定外の出力トルクの増大”, ”failure”:”RW に想定以上の電流が流れる”, ”test_check”:”RW トルク上昇時の突入電流が、想定した範囲ないかを確認する。”, ”example”:”EQU FM 不具合表 id 87”}
?circuit_board ont:hasParameter ?shape. ?circuit_board ont:include ?electrical_line. ?electrical_line ont:transfer ?electricity. ?electricity ont:hasParameter ?current. ?circuit_board rdf:type ont:Circuit_Board. ?electrical_line rdf:type ont:Line. ?electricity rdf:type ont:Electrical_Power. ?shape rdf:type ont:Shape. ?current rdf:type ont:Current.	?shape ont:hasStatus 1.	?current ont:hasStatus 1.	{”cause”:”回路基板が外力等により反り返る”, ”failure”:”内部の絶縁層が破壊され導通が発生する”, ”test_check”:”くみ上げ時は基板の反り返りが発生していないかを目視で確認する”, ”example”:”EQU FM 不具合表 id 88”}
?circuit_board ont:hasParameter ?shape. ?circuit_board rdf:type ont:Circuit_Board. ?shape rdf:type ont:Shape.		?shape ont:hasStatus 1.	{”cause”:”付近に別コンポやハーネスがある状態で無理に取り付ける”, ”failure”:”回路基板の形状変化”, ”test_check”:”くみ上げ時、回路のそばに挟まるようなものがないか、当たっている構造物はないかを確認する”, ”example”:”EQU FM 不具合表 id 88”}

?sun_sensor ont:hasOutput ?information. ?information ont:hasParameter ?noise. ?sun_sensor rdf:type ont:Sun_Senser. ?information rdf:type ont:Electrical_Signal. ?noise rdf:type ont:Noise.		?noise ont:hasStatus 1.	{”cause”:”部材の表面処理を指定できていない ,failure”, ”test_check”:”SS を取り付ける際の構造体と SS の表面処理について設計段階で確認し、ノイズが乗っていないことを試験で確認する”, ”example”:”EQU FM 不具合表 id 92”}
?circuit_board ont:include ?electrical_line. ?electrical_line ont:transfer ?electricity. ?electricity ont:hasParameter ?current. ?circuit_board rdf:type ont:Circuit_Board. ?electrical_line rdf:type ont:Line. ?electricity rdf:type ont:Electrical_Power. ?current rdf:type ont:Current.		?noise ont:hasStatus 1.	{”cause”:”回路基板端の GND ビアがブラケットと導通する”, ”failure”:”GND が意図せしない形で別の場所に落ちてノイズになる”, ”design_check”:”くみ上げ時にビアなどの接触で GND が意図せず落ちていないかを確認する”, ”example”:”EQU FM 不具合表 id 93”}
?circuit_board ont:hasParameter ?shape. ?circuit_board rdf:type ont:Circuit_Board. ?circuit_board ont:hasOutput ?elec_signal. ?elec_signal rdf:type ont:Electrical_Signal. ?elec_signal ont:hasParameter ?correctness. ?correctness rdf:type ont:Correctness.		?correctness ont:hasStatus 1.	{”cause”:”機器の GND が共通化されていない”, ”failure”:”テレコマ疎通ができない”, ”design_check”:”機器間の GND は共通化されているかを確認する”, ”example”:”EQU FM 不具合表 id 88”}
?circuit_board ont:hasParameter ?shape. ?circuit_board rdf:type ont:Circuit_Board. ?shape rdf:type ont:Shape.		?shape ont:hasStatus 1.	{”cause”:”付近に別コンポやハーネスがある状態で無理に取り付ける”, ”failure”:”回路基板の形状変化”, ”check”:”くみ上げ時、回路のそばに挟まるようなものがないか、当たっている構造物はないかを確認する”, ”example”:”EQU FM 不具合表 id 88”}

?sun_sensor ont:hasOutput ?information. ?information ont:hasParameter ?noise. ?sun_sensor rdf:type ont:Sun_Senser. ?information rdf:type ont:Electrical_Signal. ?noise rdf:type ont:Noise.		?noise ont:hasStatus 1.	{”cause”:”部材の表面処理を指定できていない ,failure”, ”design_check”:”SS を取り付ける際の構造体と SS の表面処理について設計段階で確認し、ノイズが乗っていないことを試験で確認する”, ”example”:”EQU FM 不具合表 id 92”}
?circuit_board ont:include ?electrical_line. ?electrical_line ont:transfer ?electricity. ?electricity ont:hasParameter ?current. ?circuit_board rdf:type ont:Circuit_Board. ?electrical_line rdf:type ont:Line. ?electricity rdf:type ont:Electrical_Power. ?current rdf:type ont:Current.		?noise ont:hasStatus 1.	{”cause”:”回路基板端の GND ビアがブラケットと導通する”, ”failure”:”GND が意図せしない形で別の場所に落ちてノイズになる”, ”design_check”:”くみ上げ時にビアなどの接触で GND が意図せず落ちていないかを確認する”, ”example”:”EQU FM 不具合表 id 93”}
?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Component. ?component22 rdf:type ?component222. ?component222 rdfs:subClassOf* ont:Component. ?component ont:hasOutput ?information. ?component22 ont:hasInput ?information. ?information ont:hasParameter ?correctness. ?information rdf:type ont:Electrical_Signal. ?correctness rdf:type ont:Correctness.		?correctness ont:hasStatus 1.	{”cause”:”機器の GND が共通化されていない”, ”failure”:”テレコマ疎通ができない”, ”design_check”:”機器間の GND は共通化されているかを確認する”, ”example”:”EQU FM 不具合表 id 113”}
?communication_device ont:hasParameter ?frequency. ?communication_device rdf:type ont:Communication_Device. ?frequency rdf:type ont:Communication_Frequency.			{”cause”:”TCXO の温度依存性による周波数シフト”, ”failure”:”通信機の周波数ロックが外れる”, ”test_check”:”温度依存性による周波数の変化に通信機が追従できるかを確認する”, ”example”:”EQU FM 不具合表 id 115”}

?circuit_board ont:include ?electrical_line. ?electrical_line ont:transfer ?electricity. ?electricity ont:hasParameter ?current. ?circuit_board rdf:type ont:Circuit_Board. ?electrical_line rdf:type ont:Line. ?electricity rdf:type ont:Electrical_Power. ?current rdf:type ont:Current.		?current ont:hasStatus 1.	{”cause”:”ダイオードの向きを逆にはんだ付けする ,failure”, ”design_check”:” はんだ付けした素子の向きは正しいかを目視で確認、電流を流して電流値に異常がないかを確認する”, ”example”:”EQU FM 不具合表 id 126”}
?circuit_board ont:include ?electrical_line. ?electrical_line ont:transfer ?electricity. ?electricity ont:hasParameter ?current. ?circuit_board rdf:type ont:Circuit_Board. ?electrical_line rdf:type ont:Line. ?electricity rdf:type ont:Electrical_Power. ?current rdf:type ont:Current.		?noise ont:hasStatus 1.	{”cause”:”回路基板端の GND ビアがブラケットと導通する”, ”failure”:”GND が意図せしない形で別の場所に落ちてノイズになる”, ”design_check”:”くみ上げ時にビアなどの接触で GND が意図せず落ちていないかを確認する”, ”example”:”EQU FM 不具合表 id 93”}
?heater ont:hasOutput ?heat. ?heat ont:hasParameter ?heat_transfer_amount. ?heater rdf:type ont:Heater. ?heat rdf:type ont:Heat. ?heat_transfer_amount rdf:type ont:Heat_Transfer_Amount.		?heat_transfer_amount ont:hasStatus 1.	{”cause”:”ヒーターがくみ上げ時のテンションではがれる”, ”failure”:”ヒーターから意図した熱量を伝達できない”, ”test_check”:” くみ上げ時、ヒーターに余計なテンションがかかっていないか、コンポと干渉していないか確認する”, ”example”:”EQU FM 不具合表 id 139”}
?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Senser. ?component ont:hasOutput ?information. ?harness ont:transfer ?information. ?information ont:hasParameter ?correctness. ?harness rdf:type ont:Harness. ?information rdf:type ont:Electrical_Signal. ?correctness rdf:type ont:Correctness.		?correctness ont:hasStatus 1.	{”cause”:”コネクタの接触不良 ,failure”, ”test_check”:”くみ上げ時にテレメトリの異常が起きていないかを確認する”, ”example”:”EQU FM 不具合表 id 155”}

?mli ont:hasParameter ?shape. ?mli rdf:type ont:Mli. ?shape rdf:type ont:Shape.		?shape ont:hasStatus 1.	{”cause”:”ボンディングワイヤ厚みが要求値を超える”, ”failure”:”包絡域を逸脱する”, ”design_check”:”厚みを調整するときはマージンを込めて指定する”, ”example”:”EQU FM 不具合表 id 157”}
?valve ont:hasParameter ?shape. ?valve rdf:type ont:Valve. ?shape rdf:type ont:Shape.		?shape ont:hasStatus 1.	{”cause”:”流路に微小な粒子が混入する”, ”failure”:”Valve が Open 故障する”, ”design_check”:”どの程度の粒子なら許容できるか、それを取り除く手順を持てるかを確認する”, ”example”:”EQU FM 不具合表 id 188”}
?reaction_wheel ont:hasInput ?electricity. ?component rdf:type ?component2. ?component2 rdfs:subClassOf* ont:Component. ?component ont:hasOutput ?electricity. ?component ont:hasOutput ?electricity2. ?electricity2 ont:hasParameter ?noise. ?electricity2 rdf:type ont:Electrical_Power. ?reaction_wheel rdf:type ont:Reaction_Wheel. ?electricity rdf:type ont:Electrical_Power. ?noise rdf:type ont:Noise.		?noise ont:hasStatus 1.	{”cause”:”Reaction Wheel の回転が電気的ノイズとして伝わる”, ”failure”:”センサー値にノイズが乗る”, ”design_check”:”試験時に Reaction Wheel を動作させつつ、センサー値が正常であることを確認する”, ”example”:”EQU FM 不具合表 id 189”}

付録 B

帰納法を用いた知識分子の一般化

本研究での提案手法を用いて、知識を知識分子に落とし込んでいく際、複数の知識分子間で類似の知識、条件が保存される可能性がある。この場合、それらの知識分子間の共通する特徴から、知識を帰納的に一般化し、より再利用性が高く、信頼性の高い知識を定義することが可能になると考えられる。本章では、計算機を用いた知識分子の帰納法について、検討を行った結果を説明する。

B.1　帰納法が活用できる事例

複数の知識分子に共通する特徴がみられる場合、それらの知識を帰納的に一般化することで、効率的な知識利用ができる可能性がある。例えば、図 B.1 のように、複数の知識分子で、過電流に伴い低下した電圧が Under Voltage Contnrol の閾値に掛かり電源が落ちるという不具合が定義されていたとする。その知識が再利用される条件として、ある知識分子では XACT RW が、別の知識分子では Control Momentum Gyro (CMG) が条件として、指定されていたとする。このように、「トルクを発生させる装置が電源供給されている」という状況で過電流 UVC に関する不具合を表す知識分子が、それ以外にもいくつも存在する場合、これらの知識分子は「トルクを発生させる装置が電源供給されている」、という条件として一般化することができる。

B.2　帰納法のメリット

計算機を用いて帰納法的に知識分子を一般化することは、以下のようなメリットが存在する。

1. 知識分子を集約し、知識再利用にかかる時間を短縮できる
2. 集約し一般化した知識と、反例となる知識をセットで整理することで、どのような条件でなぜ発生するかが明らかな深い知識へと昇華することができる。

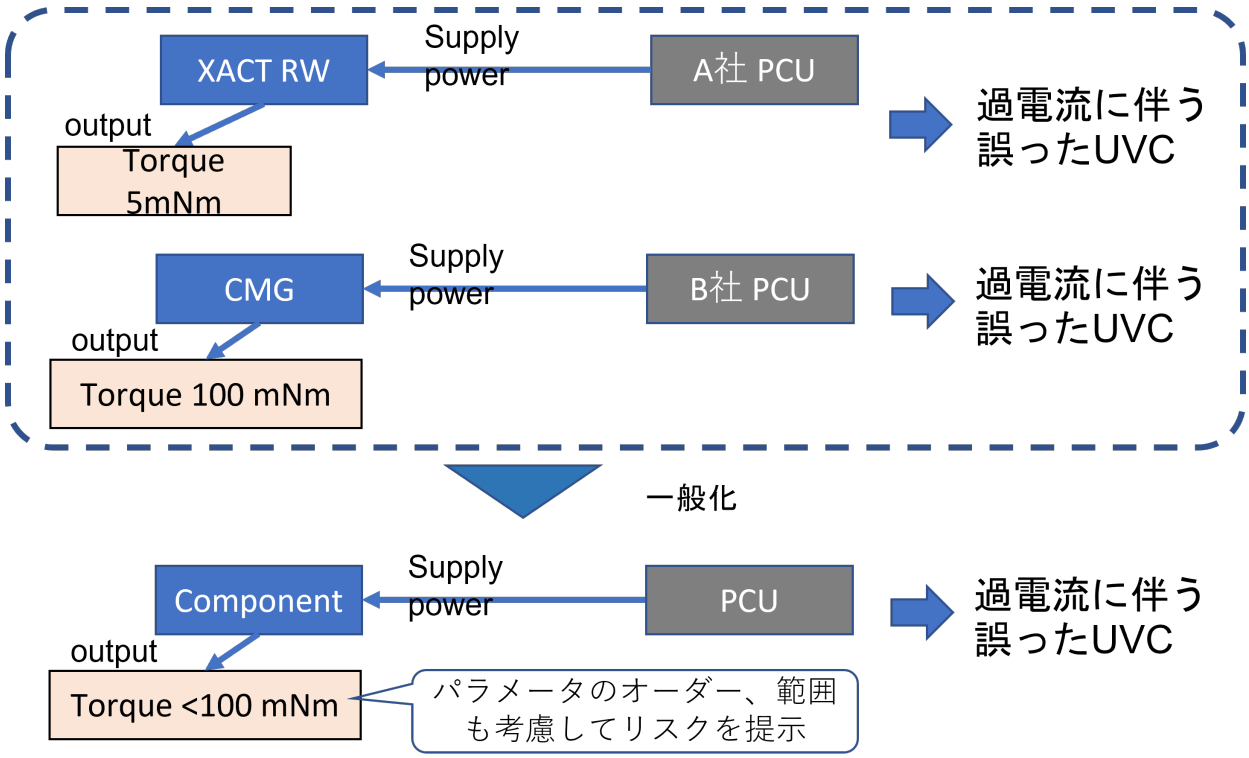


図 B.1　知識分子一般化のイメージ

3. 機械的に一般化が可能になるため、エンジニアが一般化について悩む必要がなくなる。

このうち 2 の効果については、原因があまりわかっていない不具合に関する知識分子が蓄積した際は、重要な知見が得られると考えられる。

B.3　実現方法

複数の知識をもとに、帰納的方法で知識を一般化する方法として、以下のような方法が考えられる。

1. グラフ構造に特化した機械学習 (Graph Neural Network)
2. グラフ構造に特化した頻出パターンマイニング手法

知識分子は、特に Condition Graph, Operator 部分がグラフ構造を持つため、一般化するためにはその構造を入力に取れる手法を用いる必要がある。そのため、上記の手法はどちらもグラフ構造に特化した手法となっている。

グラフ構造に特化した機械学習は、Graph Neural Network と呼ばれ、Knowledge Graph の補完 [57] や、物品販売を行う Web サイトの Recommendation System に応用されている [58]。

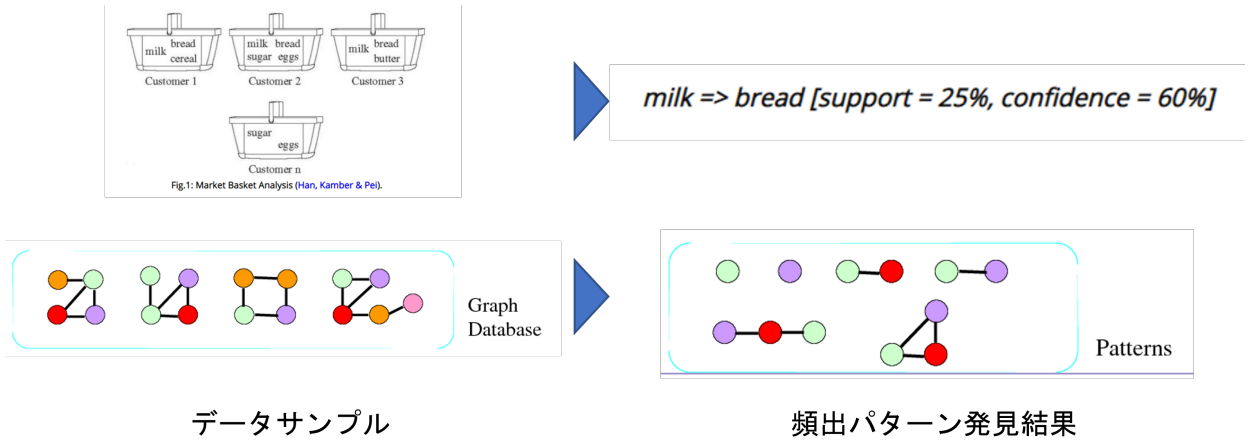


図 B.2　頻出パターンマイニングのアルゴリズムイメージ [59]

グラフ上のノード間の関係、あるいはグラフ全体と、それに対応するラベルを学習し、グラフのかけた関係を補完したり、分類したりする。これらの手法は、様々なグラフを柔軟性高く学習でき、サンプル数が増えるにしたがって、ある知識が結びつくかどうかを精度よく判定できるようになると考えられる。一方で以下のようなデメリットが存在する。

- 一般化した知識は、Neural Network のパラメータとして保存されるため、人間に解釈することができない
- 精度を出すためには非常に多くの学習が必要である。

一方で、頻出パターンマイニングは、データマイニングで用いられる手法で、事例間で共通する特徴をカウントし、共通するものが多いものを頻出パターンとして提示するものである。例えば、物品販売を行う Web サイトにおいてユーザーの購買履歴から、よくある購入品の組み合わせを発見し、抱き合わせのパッケージを検討することなどに使用される (図 B.2) [59]。こちらは、人間に理解できない特徴をサンプルから抽出することはできない一方で、少ないデータ数でも共通部分を抜き取ることが可能であり、また抽出結果も人間が理解でき、正しいかを判断することが可能である。

宇宙システムにおいて、プロジェクトの頻度、共通で起こる不具合の頻度は年単位であり、機械学習で使用できるようなサンプル数を入手するのに非常に時間がかかる。そこで本研究では、頻出パターンマイニングの手法を用いて、知識分子を一般化することにした。

グラフ構造に対する頻出パターンマイニングは以下のようなアルゴリズムが存在する。

1. Apriori-based Graph Mining (AGM) [60]
2. gSpan [61]

3. FAGv-gSpan [62]

AGM は、グラフの頂点をもとに頻出パターンを探っていくアルゴリズムで、幅優先で頻出パターンの探索を行う。一方で、gSpan はグラフの辺をもとに頻出パターンを探る方法で、深さ優先で頻出パターンの探索を行う。また FAGv-gSpan は、gSpan に加えて、定量的な数値のノードに対しても、頻出パターンを発見し、不等式などパターンの条件を発見することができる。

本研究では、複数の知識分子から、できるだけ大きく共通するパターンを 1 つでも探索できることが重要であることがから、深さ優先の gSpan を採用し検討を行った。数値を考慮できる FAGv-gSpan については、今後の検討事項とする。

B.4 実験

過電流シャットダウンに関する不具合に関する知識を題材に、gSpan を用いた一般化を行い、帰納法が実現できるかの検証を行う。今回、図 B.5 の 4 つを Condition Graph に持つ異なる知識分子が生成されたとする。いじれのグラフにおいても、特定のコンポ (RW, Motor, Valve, CMG) が Power Control Unit から電源供給されており、トルク、または摩擦力を output に持っている。なお、実験を簡易化するため、いずれの知識分子も、Content として「電流が急激に増加し、過電流シャットダウンが起きる」とし、Operator および Analysis はないものとする。(本来は、Content が類似しているかどうかも、人間が判断、もしくは自然言語処理により判定し、類似の知識を集める必要がある)。Condition Graph に対して gSpan を使用し、共通する特徴を導き出す。

B.5 結果と考察

gSpan を実施した結果、図のようなグラフパターンが、頻出パターンとして抽出された。(a) は、「Power Control Unit から電力を供給されたコンポがトルクを発生する」というグラフである。また、(b) は、「Power Control Unit から電力を供給されたコンポが摩擦力を発生する」というグラフである。いずれの場合も、RW と CMG, Motor と Valve に関する知識分子を一般化できているといえる。一方で (c) は、さらに共通部分が抽出されたもので、「コンポが Power Control Unit から電源供給されている」というグラフになる。

(a), (b) に関しては、確かに今回の不具合に関する特徴を捉えられていると考えられる一方で、(c) はさらに共通化してしまった結果、不具合を起こす特徴 (今回の場合、トルクや力の発生など）が損なわれてしまっており、一般化として妥当であるとは考えにくい。

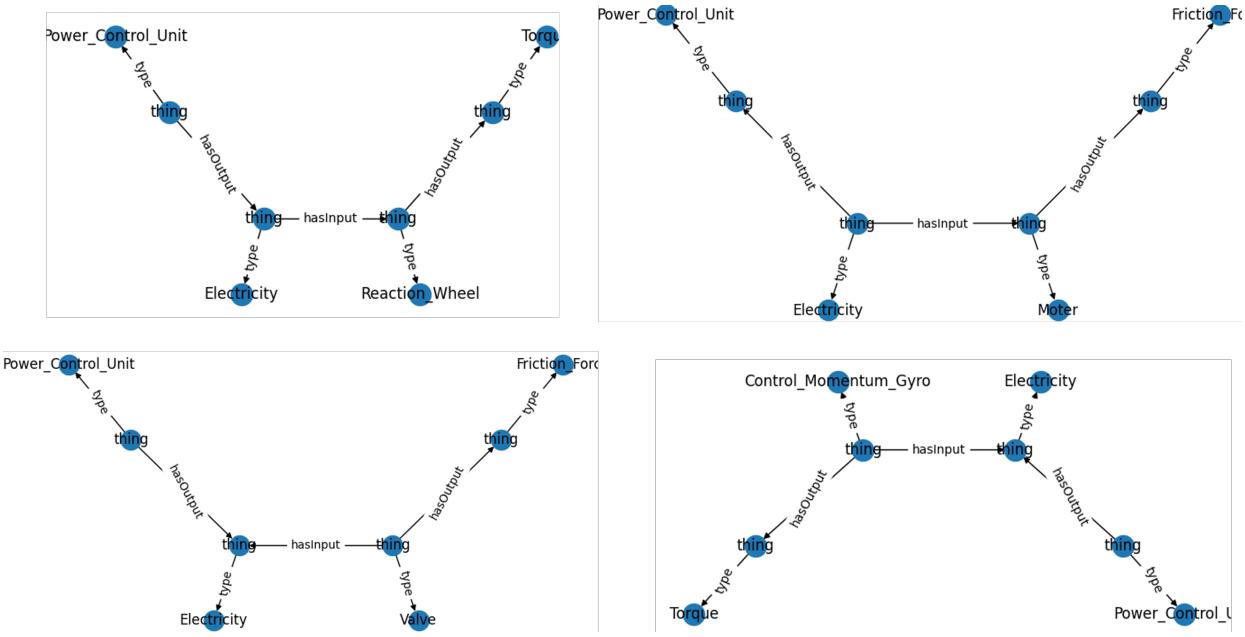


図 B.3　実験で一般化する知識分子

B.6　今後の課題

前述のように、頻出パターンマイニングの手法により、知識分子の共通する特徴を抽出し、一般化することは可能であると考えられる一方で、共通する要素を絞りこみすぎて、誤った一般化をする危険性がある。今後の課題としては、そのように絞り込みすぎてしまうことを防ぐため、例えば全く関係な知識分子にも共通でみられるような特徴は、一般化の結果として判定しないなどの工夫が必要であると考えらる。また、このように完全な一般化は計算機を用いても難しいと考えられるため、知識分子を生成する際に、できるだけ物理的なオントロジーを使用するとともに、人間の議論に基づく一般化も重要であると考えられる。

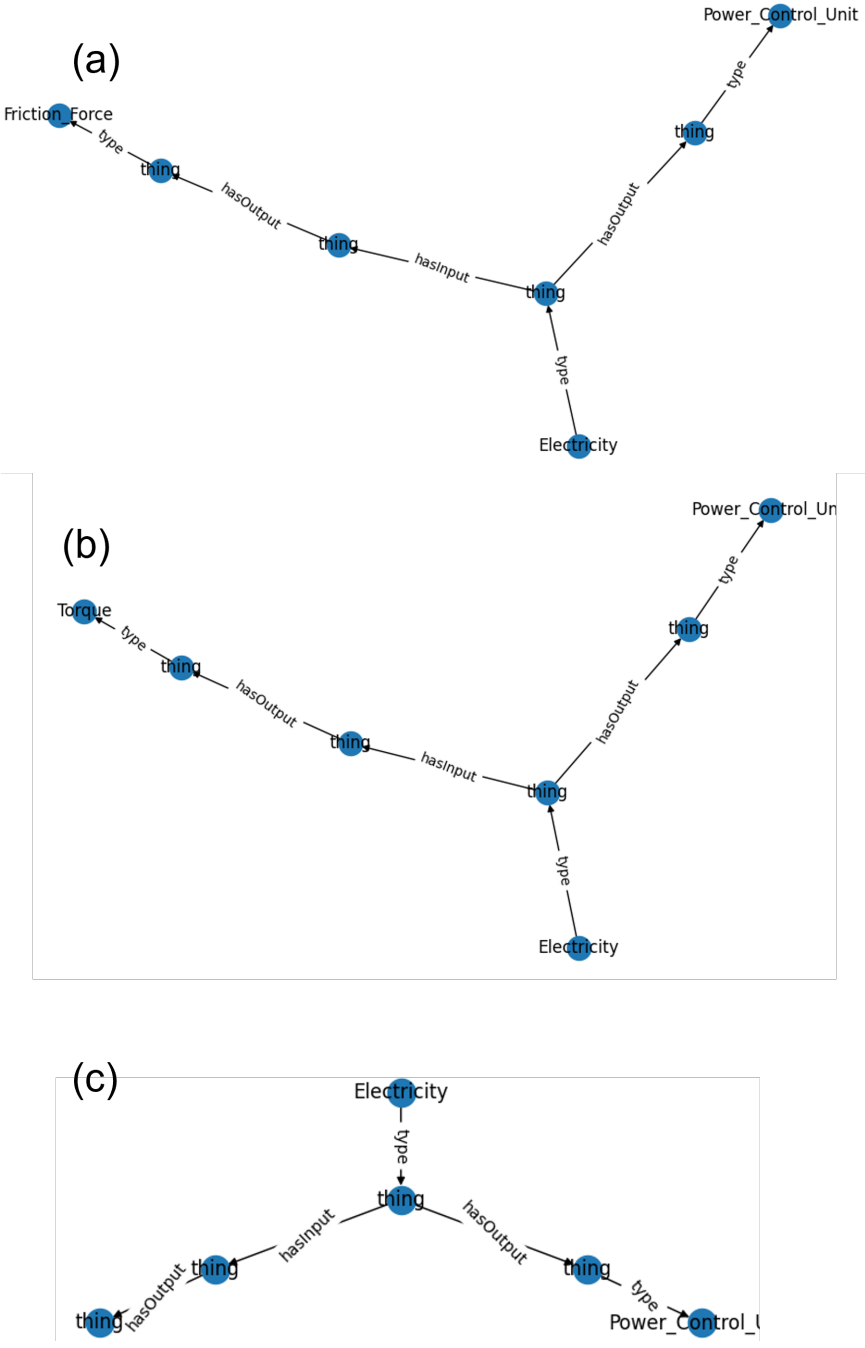


図 B.4　gSpan により抽出したグラフ構造

付録 C

自然言語処理による知識分子の自動生成

本研究で提案した知識分子に基づく知識再利用手法は、計算機による知識利用により効率化を目指した一方で、知識分子の記述自体には一定の慣れと時間がかかる。特に、知識分子の Condition Graph の生成には、通常のドキュメントの記述に加えて、オントロジーへの理解とそれに基づいた記述が必要であり、慣れと時間が必要となってしまう。この課題に対して、特に衛星の不具合に関する知識分子について、本研究では自然言語処理を活用して自動で知識分子の Condition Graph を自動生成する手法の検討を行った。本章ではその検討結果について説明する。

C.1　Condition Graph の自動生成の課題

自然言語から、グラフ構造を生成する手法に関する研究として、非構造データから構造化データを取得する Open Information Extractioin と呼ばれる手法がある [63]。Open Information Extraction は、さらに形態素分析をベースにルールベースで構造化する方法 [64]、または教師あり学習で取得する方法がある [65, 66]。ルールベースの構造化は、文章自体の表記ゆれに弱く、正確な構造化データを獲得するのは難しい。また、知識分子の生成特有の問題として、文章に表れない要素も、グラフ内に定義する必要がある場合がある。例えば衛星の不具合に関する知識分子を、不具合に関する解析レポートから自動生成することを考える。不具合のレポートにおいては、不具合が発生するコンポーネント名などは含まれている可能性が高いが、物理的な関係、例えばコンポーネント間で電力が入力、出力されていることまでは表現されない可能性がある。そのため、文章中の単語をもとに推論を行うルールベース手法では、正しい知識分子が生成されない可能性がある。一方で、教師あり学習では、ベースとなる大量の教師データが必要となり、実装にコストがかかる。特に、知識分子の場合、学習する対象の知識の種類

も増え続けるため、継続的な教師データの作成、学習が必要となってしまう。

C.2　提案手法

従来の手法では、精度や利用までのコストに問題がある。そこで本研究では、以下のアイデアに基づき、知識分子を自動生成する方法を検討した。

- 知識分子を用いて知識を再利用するために、提案手法のフレームワークでは、宇宙システムの設計は同じオントロジーでモデル化されている。
- 不具合の知識分子は、どのような衛星内の要素が存在し、関係するかが大切をグラフ要素として定義する。一方で、重要な要素は知識分子を再利用するための宇宙システムのモデルに含まれている。
- 不具合に関する知識分子の Condition Graph は、その不具合が起きた宇宙システムの設計モデルから、関連する要素を自動的に抜き出すことで生成できるのではないか

すなわち、知識利用のために作っていた宇宙システムモデルを、新たな知識分子生成のためにも利用する、というアイデアである。そしてこのアイデアを実現するためには、以下の機能が必要である。

- 自然言語で書かれる不具合に関するドキュメントから、関連する宇宙システムモデルの部分グラフを抜き出す

これを実現することで、図 C.1 のようなフローで、知識分子を自動生成できると考えられる。

この、自然言語から関連するモデルを抽出するというタスクは以下のアルゴリズムを組み合わせることで実現した。

1. Word2Vec [52]：教師データをラベリングする労力を必要とせず、単語を特徴ベクトルへと埋め込む手法である。本研究では実際のプロジェクトの議事録、論文をベースに学習を行い、宇宙に関連する単語の学習を実現した
2. Word Mover's Distance [53]: 前述の Word2vec を使用し取得した単語のベクトルを元に、文中の単語の出現頻度なども踏まえて、文と文の距離を計算できる手法である。
3. 遺伝的アルゴリズム (GA) [67]: ヒューリスティックな最適化アルゴリズム。宇宙システムモデルのうち最適な部分グラフを抜き出す際に、抜き出し方が膨大であるため、GA を利用した。

イメージを図 C.2 に示す。これらを組み合わせて、以下のような流れで、知識分子の自動生成を実現する。

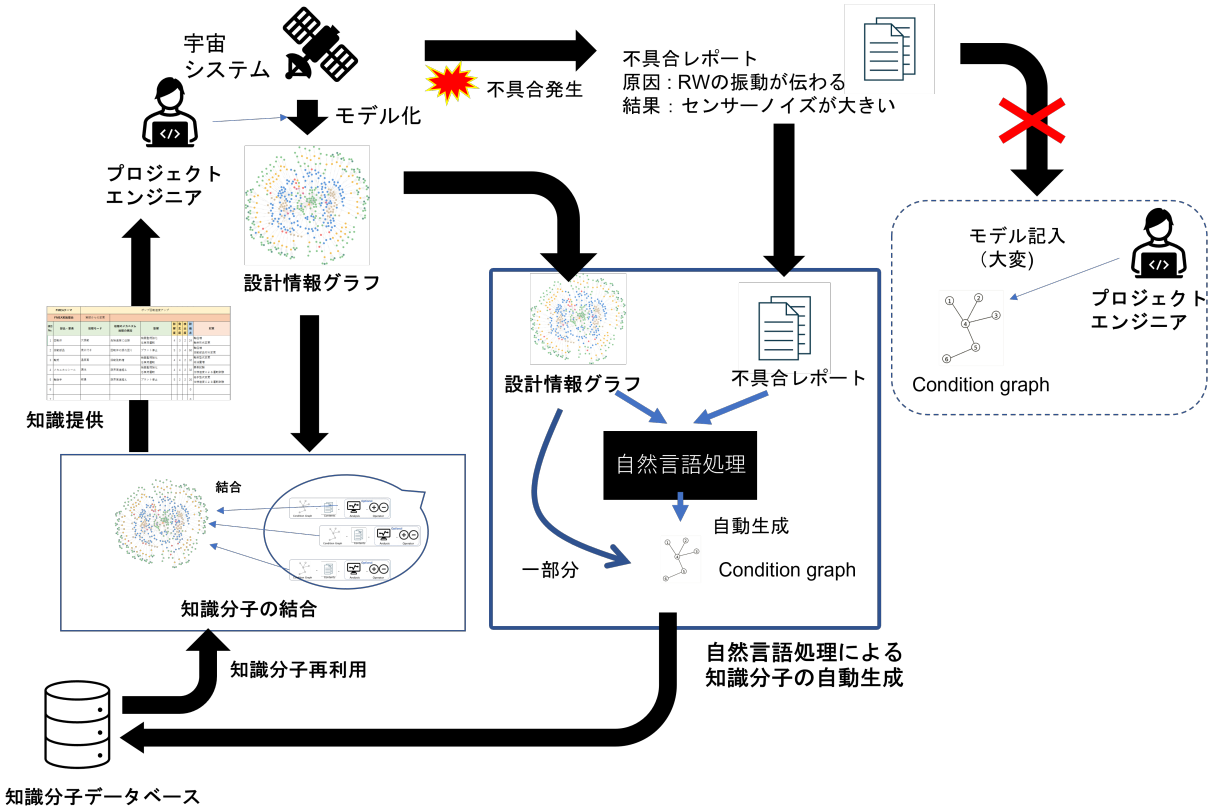


図 C.1　知識分子自動生成のフロー

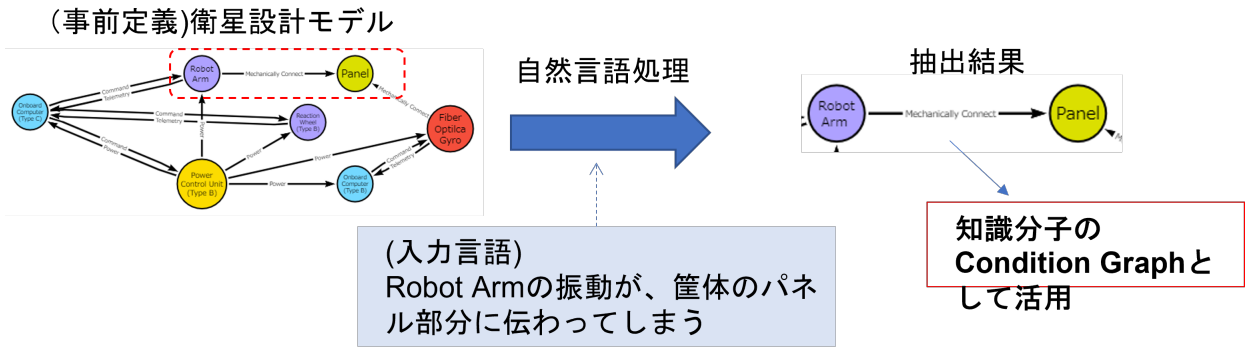


図 C.2　設計情報グラフからのグラフ抽出のイメージ

1. (事前準備 1) 宇宙システムに関するドキュメントを大量に集めて、Word2Vec を用いて単語の特徴ベクトルを学習する。この時、教師データのラベリングは必要ない。
2. (事前準備 2) オントロジーに登場する各概念の自然言語での説明、単語を定義しておく。
3. 不具合が発生した宇宙システムモデル、および不具合に関するレポート (原因や関連コンポについて書かれたドキュメント) を入力する
4. 宇宙システムモデルの部分グラフを一つ抜き出す。
5. 抜き出した部分グラフを、オントロジーに関する説明で、自然言語の文に変換する
6. 不具合に関するレポートと、部分グラフの変換文の距離を Word Mover's Distance で計算する
7. 手順 4 から 6 を繰り返し、遺伝的アルゴリズムで最も距離が近い部分グラフを抜き出す。

最後の最もレポートと距離が近い部分グラフが、不具合の知識分子の Condition Graph となる。この提案手法は以下のような特徴を持つ

- 教師データのラベル付けが必要なく、ほとんど学習に手間がかからない
- 同じオントロジーで作られたモデルを切り抜いたものなので、正しい知識分子が生成されやすい。

すなわち、プロジェクトの中でエンジニアは、宇宙システムモデルを構築しつつ、従来通り不具合のレポートを自然言語で書くだけで、知識の蓄積、再利用が可能となる。

C.3　実験

以下の設定で、不具合に関するレポートを生成し、知識分子を自動生成した。

不具合レポート　図 C.3 に示す二つの文章を入力の不具合レポートとした。
宇宙システムモデル　章で利用した ISSL をそのまま利用した。
遺伝的アルゴリズムの設定　: 1 世代の個体数を 30, 世代数を 100 として設定した。
初期解の工夫　: 精度を高めるため、次のような方法で初期解を求めた。宇宙システムモデルのうち、1 辺だけを抜き取り、それぞれ不具合レポートとの距離を計算した。最も距離が近い上位 10 個を、初期解のうちの 1 部として定義した。

初期解の工夫をしたのは、そもそも 1 辺でも不具合レポートとの距離が遠いような部分グラフは、最終的に切り抜かれている部分グラフの中にも含まれる可能性は低いからであり、よりよい解への収束を早められる効果がある。

id	condition	cause	effect
1	回路基板からジャイロへの電源供給	PCUからの供給電流が少ない	Gyroが動作しない
2	回路基板とリアクションホイール	RWの電流値が想定よりも大きい	過電流にかかる

図 C.3　実験での入力としての不具合レポート

Name	Condition	Content
1	?contact rdf:type ont:Contact . ?circuit_board rdf:type ont:Circuit_Board . ?gyro_senser rdf:type ont:Gyro_Senser . ?electrical_power rdf:type ont:Electrical_Power . ?circuit_board ont:hasOutput ?electrical_power . ?circuit_board ont:include ?contact . ?gyro_senser ont:hasInput ?electrical_power .	{'cause': 'PCUからの供給電流が少ない', 'effect': 'Gyroが動作しない'}
2	?contact rdf:type ont:Contact . ?circuit_board rdf:type ont:Circuit_Board . ?reaction_wheel rdf:type ont:Reaction_Wheel . ?structure rdf:type ont:Structure . ?circuit_board ont:communication_connect ?contact . ?structure ont:screw_connect ?reaction_wheel . ?circuit_board ont:include ?contact . ?contact ont:communication_connect ?circuit_board .	{'cause': 'RWの電流値が想定よりも大きい', 'effect': '過電流にかかる'}

図 C.4　実験結果として生成された知識分子

C.4　結果

図 C.4 に示すような、知識分子の Condition Graph が自動生成された。結果として、いくつか関係のない要素が含まれるものの、必要な要素が存在する Condition Graph が生成できた。

C.5　今後の課題

上記のように簡単な例では、ある程度の知識分子を自動生成できることが分かった。一方で、今回の研究では、提案手法がそのほかあらゆる宇宙システムモデルに対して、同様の結果が出せるか、ほかの手法と比較して優位化については、評価を行うことができていない。また、アルゴリズムに関してもほかの手法を検討する余地があると考えられる。これらの点については今後の検討課題としたい。

付録 D

不具合以外の知識分子の応用

本研究で提案する知識分子は、不具合に関する知識の再利用、設計補完以外にも様々な応用に利用できると考えられる。本章では、「自動設計」と「自動シミュレーション構築」の二つの応用を紹介し、それらの効果、簡単な適用事例を紹介する。

D.1 自動設計

知識分子は、Operator により、結合先に複数の要素 (グラフのノード、エッジ) を追加することが可能である。この性質を利用し、特定の「設計に関する要求」を条件に結びつき、「実現方法」、あるいは「分解した機能」を残す知識分子を定義することで、宇宙システムをはじめとするシステムの要求の自動分解、実現方法の提示が可能になると考えられる。すなわち、目標を実現するシステムのアーキテクチャ候補を自動で生成することが可能であると考えられる。実際に以下のような知識分子を定義し、実験を行ってみた。

1. 機能要求: 3 軸姿勢制御 → 機能要求: 姿勢決定、姿勢制御 に分解する知識分子
2. 機能要求: 姿勢決定、姿勢制御 → 姿勢制御モジュール を提案する知識分子

これらの知識分子に対して、三軸姿勢制御を機能要求とする衛星の設計グラフを入力した。知識分子を結合した結果を図 D.1 に示す。想定通り、三軸姿勢制御に対して、姿勢制御モジュールを用いるという設計解が提示できた。以下に知識分子での自動設計に期待される効果、応用を示す。

1. 過去に思いついたアイデアをもとに、計算機によって網羅的にアーキテクチャ候補を探ることができる
2. Analysis を使うことで、アーキテクチャ候補から、設計の解析、設計性能評価を自動で行うことができる。

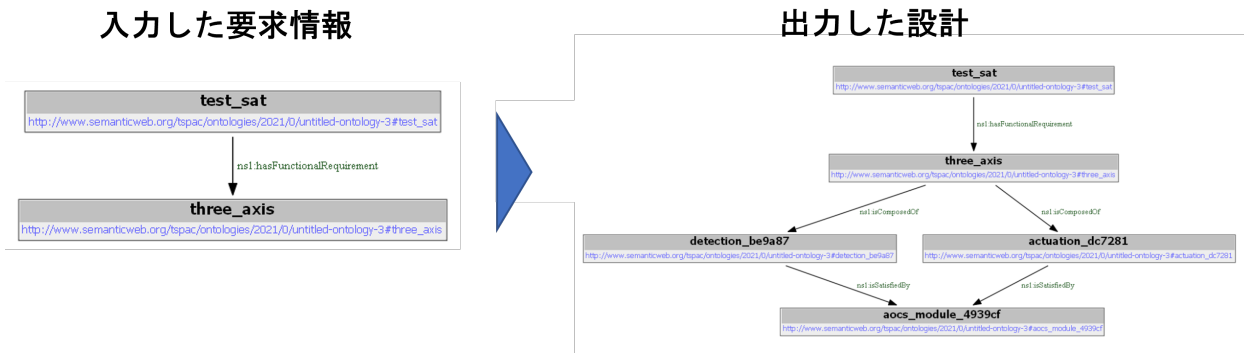


図 D.1　知識分子による自動設計結果

3. 故障リスクや設計補完の知識分子を組み合わせることで、開発フェイズでのリスクや検証項目の多さを踏まえたうえでアーキテクチャ候補を選定することができる。

一方で、現状知識分子の結合アルゴリズムには最適化機能がないため、さまざまな設計解を探索したい場合は、最適化のアルゴリズムや結合する知識分子を操作するアルゴリズムを追加で検討する必要がある。

D.2　シミュレーターの自動構築

本研究で提案したように、知識分子は Analysis という定量的なパラメータのインプット、アウトプットを持ち、結合先からそれらの値を読み取って解析を実施する機能を持つ。この機能を応用することで、知識分子の結合と解析を組み合わせ、自動でシミュレーショターを構築することが可能であると考えられる。ただし、現在の知識分子結合アルゴリズムにおける単純な Analysis 活用では、結合した瞬間 1 回のみでしか実行できないため、アルゴリズムを改良する必要がある。そこで下記のような工夫を行う。

- シミュレーターに使用する解析モデルを、知識分子の Analysis に定義する。また、その Analysis が実行される適切な条件 (コンポの種類、コンポ間の関係）を Condition Graph に定義する。
- 結合のアルゴリズムの際は、Analysis は実行せず Condition Graph のみで結合する。結合後、知識分子の ID と、Input となるパラメータの ID, output となるパラメータの id を保存する。
- 知識分子に定義された Analysis を逐次実行し、対応するパラメータの値を更新する。これを繰り返すことで、シミュレーションとする。

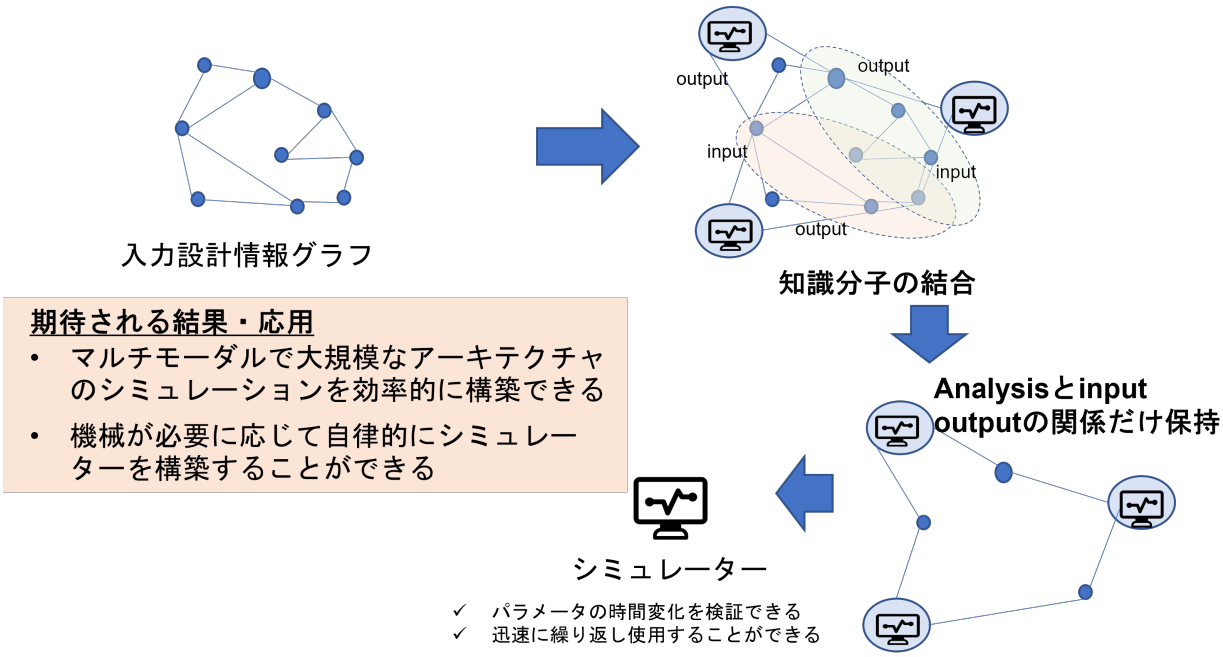


図 D.2　知識分子によるシミュレター自動構築の概要

この結合のイメージを図 D.2 に示す。

D.2.1　実験とその結果

実際に、図 D.3 に示すような単純な回路をシステムグラフに落とし込み、表 D.1 で示す知識分子をシミュレーターの要素として結合した。結合を実施した結果、図 D.4 のような結合が見られた。

結合したグラフをもとに、適切にパラメータの初期値を決定し、シミュレーションを実施した結果、Battery に定義されている電圧、回路の電流値は図 D.5 のように変化した。この結果から以下の様子が確認できる。

表 D.1: シミュレータ～を構築するために定義した知識分子

Condition	Analysis(スクリプト名)	Input	Output	Content
?time rdf:type ont:Time . ?dt rdf:type ont:Delta_Time .	time_val	?time ?dt	?time	{'content':' 時間経過の部分'}

?dt rdf:type ont:Delta_Time . ?pcu rdf:type ont:Power_Control_Unit . ?elec rdf:type ont:Electricity . ?pcu ont:hasOutput ?elec . ?elec ont:hasParameter ?voltage . ?current rdf:type ont:Current . ?elec ont:hasParameter ?current . ?voltage rdf:type ont:Voltage . ?elec_base rdf:type ont:Electricity . ?pcu ont:hasInput ?elec_base . ?voltage_base rdf:type ont:Voltage . ?elec_base ont:hasParameter ?voltage_base . ?capacity rdf:type ont:Power_Capacity . ?pcu ont:hasParameter ?capacity . PCU の電圧を変える’	power_output	?voltage ?current ?capacity ?voltage_base ?dt	?voltage ?capacity	’content’:’ 時間経過に伴って
?resist rdf:type ont:Resister . ?elec rdf:type ont:Electricity . ?resist ont:hasOutput ?elec . ?elec ont:hasParameter ?voltage . ?voltage rdf:type ont:Voltage . ?elec_base rdf:type ont:Electricity . ?resist ont:hasInput ?elec_base . ?voltage_base rdf:type ont:Voltage . ?current_base rdf:type ont:Current . ?elec_base ont:hasParameter ?voltage_base . ?elec_base ont:hasParameter ?current_base . ?current rdf:type ont:Current . ?elec ont:hasParameter ?current . ?resistance rdf:type ont:Resistance . ?resist ont:hasParameter ?resistance .	resist	?voltage ?voltage_base ?resistance	?current ?current_base	{’content’:’ 電圧降下’}
?ground rdf:type ont:Ground . ?elec rdf:type ont:Electricity . ?ground ont:hasOutput ?elec . ?elec ont:hasParameter ?voltage . ?voltage rdf:type ont:Voltage . の箇所の voltage 0’	ground_input	?voltage	?voltage	’content’:’Ground 接続
?ground rdf:type ont:Ground . ?elec rdf:type ont:Electricity . ?ground ont:hasOutput ?elec . ?elec ont:hasParameter ?voltage . ?voltage rdf:type ont:Voltage .	ground_output	?voltage	?voltage	’content’:’Ground 接続

の箇所の voltage 0’	

- バッテリーが放電し、電圧値が時間と共に低下している
- それとともに電流値も減少していく様子が見える

以上のように、知識分子を用いたシミュレーション構築が可能である。

D.2.2　今後の課題と展望

前述のとおり、知識分子を組み合わせてシミュレターを自動構築できる一方で、この方法には以下のような課題がある。

- 知識分子の結合の順番により、シミュレーションの解析の順番が決まってしまい、現実に即さない解析結果になる可能性がある。
- 知識分子内のノード数でパラメータ数を規定しているため、任意の個数のパラメータを統合したりする計算するような解析 (総重量や総エネルギー量など) が難しい。

一方で、知識分子を活用してシミュレターを構築することで、下記を実現できる可能性がある。

1. 設計補完の知識分子と組み合わせて、非常に詳細かつ大規模な粒度のシミュレーションを容易に構築できる。当然、マルチモーダルなシミュレーションも構築できる
2. 解析途中でアーキテクチャが変わってしまうようなケース (例えば不具合による断線など) のシミュレーションを、定量的なシミュレーションと、不具合の知識分子の結合を用いた推論を組み合わせることで実現する。

このように、知識分子を用いたシミュレーション構築は、大きな可能性があると考えられ、今後の重要な研究課題になりうると考えられる。

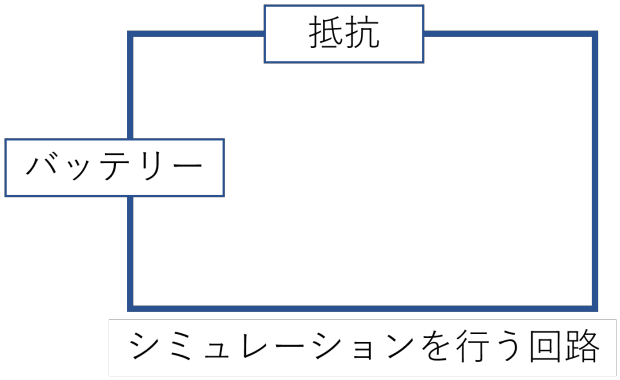


図 D.3　知識分子によるシミュレターを行う対象

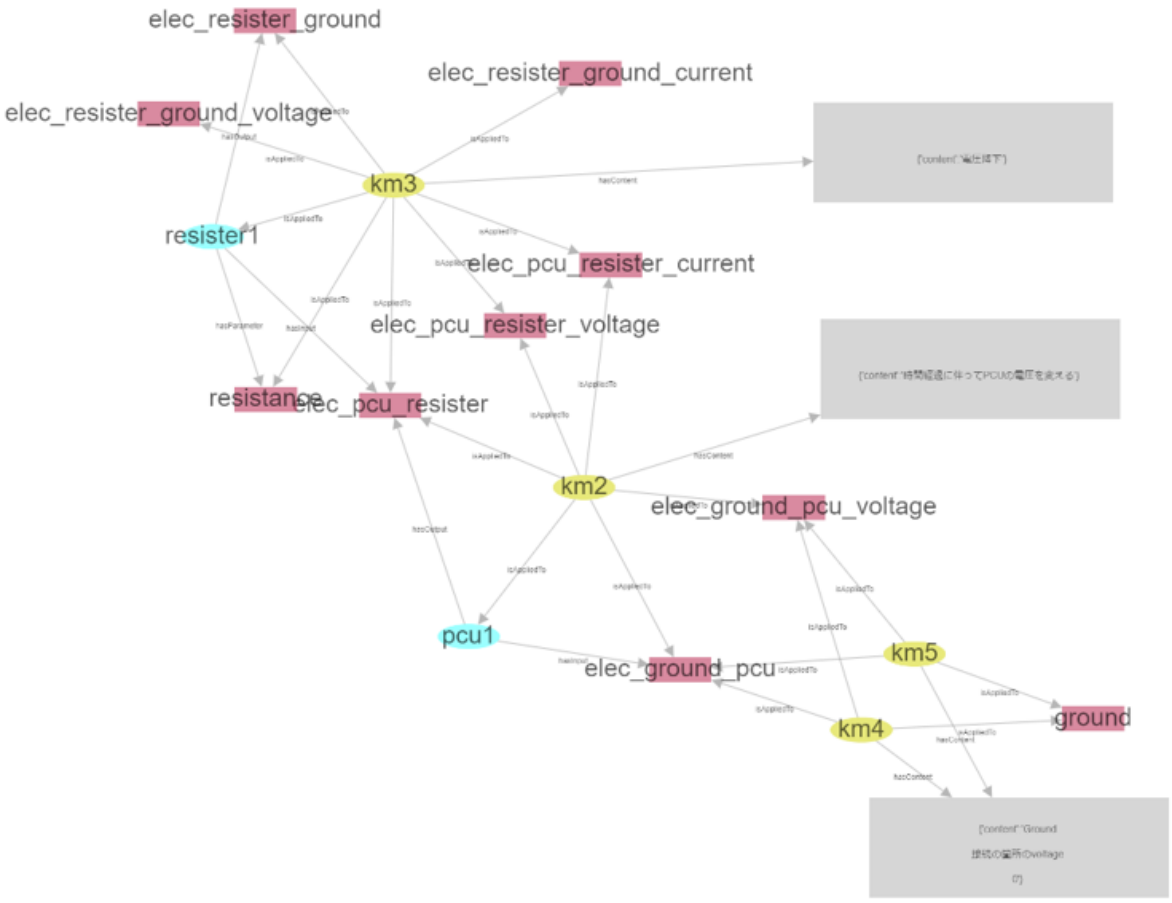


図 D.4　シミュレーション構築用知識分子の結合結果

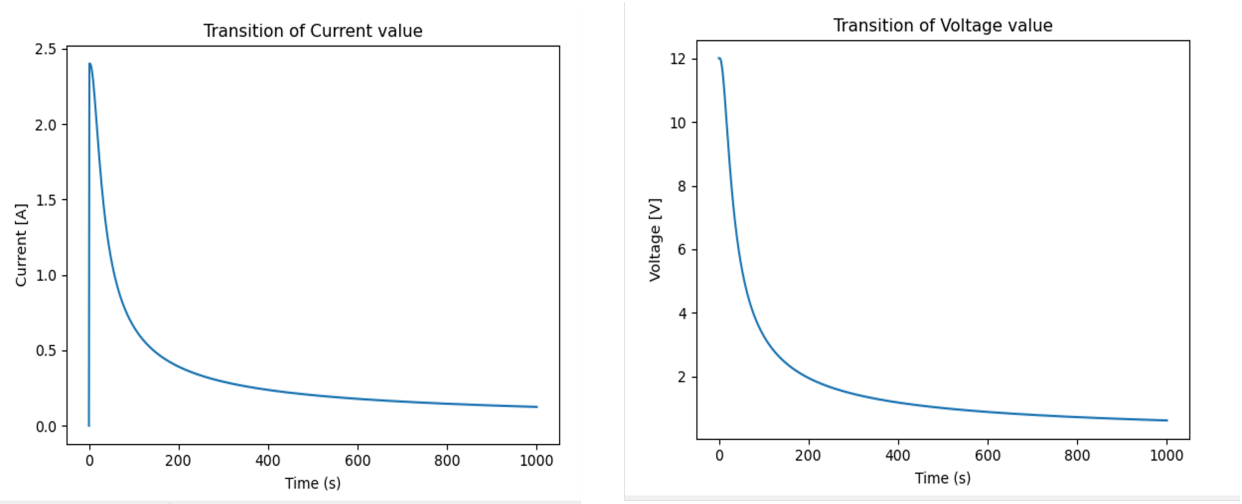


図 D.5　知識分子によるシミュレーション結果 (左：回路の電流量時間変化、右：バッテリー電圧の時間変化)

付録 E

知識分子同士の類似度計算

グラフ同士の類似度計算 [48] を使用することで、知識分子の Condition Graph 同士の類似度を計算することができる。また、Word Mover's Distance[53] を用いることで、知識分子の Content 同士の類似度も計算を行うことができる。これらを利用することで、下記の機能を実装することが可能である。

- 類似の条件で結合する知識分子同士を見つけ、知識分子を統合する (これにより知識分子の数を抑制する)
- 結合条件、内容も全く同じものに関しては、片方を削除し余計な推論を行う計算コストを縮小する。

すなわち、類似の知識を発見し、膨大な知識分子を管理し必要であれば統合、削除することに用いることができる。また、3.5 章で述べた知識分子の評価手法である Unexpectedness で利用することが可能である。図 E.1, E.2, E.3 に、EQUULUEUS の不具合知識 10 個について類似度を計算し、Heat map で可視化したものを示す。 この例では、EQU_FAULT_8 と EQU_FAULT_9 の類似度が高い (特に condition graph). 一つの知識に知識を集約することが可能かを検討するべきである。

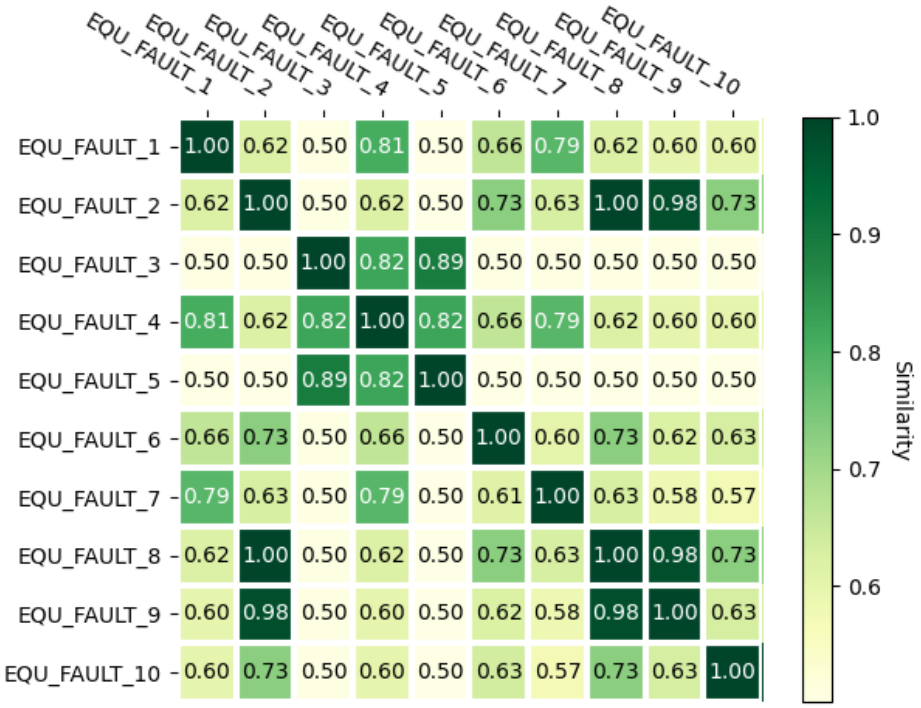


図 E.1　知識分子の Condition Graph の類似度

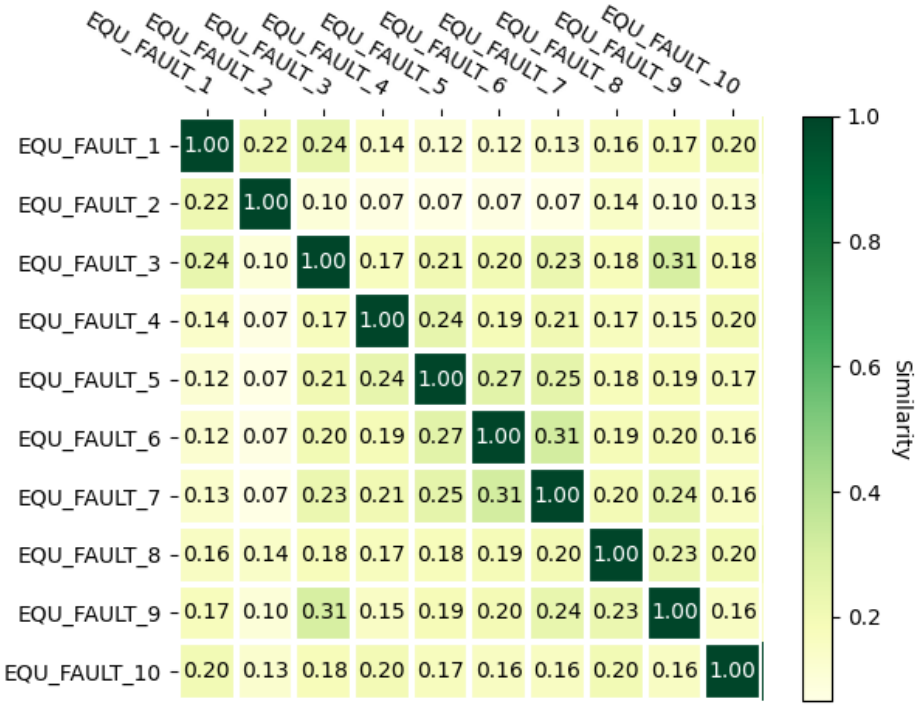


図 E.2　知識分子の Content の類似度

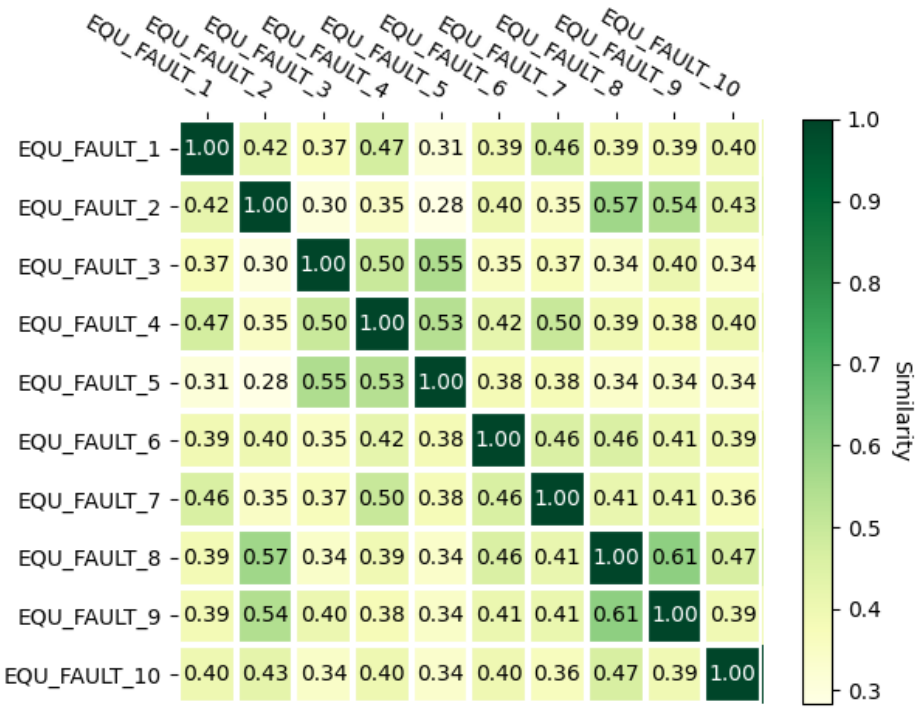


図 E.3　Condition Graph および Content の類似度の合計