実践 D-Case

ディペンダビリティケースを活用しよう！

名古屋大学
情報連携統括本部
情報戦略室
松野裕、山本修一郎

はじめに

本書では、システムのディペンダビリティを達成する手法とツールとして、JST CREST DEOS (Dependable Embedded Operating System)プロジェクトで研究開発を進めている D-Case について、わかりやすく説明し、実際のシステム開発運用でどのように利用するのか解説する。

D-Case は、ディペンダビリティケースと呼ばれるシステム保証のためのドキュメントを記述するための手法とツールを提供している。対象の読者はシステム開発、運用の実務を担当されている企業の方ならびにこれらの業務に興味のある大学生および大学院生である。

本書の構成は以下のとおりである。

1 章では、D-Case の概要を説明する。

2 章では、D-Case の基礎知識として、ディペンダビリティ、ディペンダビリティケース、リスク分析について解説する。やや専門的な内容を含むので、必要に応じて飛ばしてほしい。

3 章では、GSN(Goal Structuring Notation)という D-Case の記法の説明および演習を行う。

4 章では、GSN を用いた D-Case の作成法を説明および演習を行う。

5 章では、D-Case における議論分解パターンを説明する。

6 章でまとめを行う。

巻末に演習問題の答えをおいた。

本書で使われている D-Case の図は、すべて DEOS プロジェクトで開発中の D-Case Editor を用いて作成されている。D-Case Editor はフリーのツールであり、基本的な描画機能に加え、DEOS プロジェクトで研究開発されたさまざまな拡張機能がある。ダウンロード、説明などは、ぜひ

http://www.dependable-os.net/tech/D-CaseEditor/D-Case_Editor_J.html

をご参照いただきたい。

執筆分担

松野　裕　1 章 1.1 節、2 章 2.1 節、2.2 節，3 章 3.1 節，4 章 4.1 節、4.2 節

山本　修一郎　1 章 1.2 節、2 章 2.3 節、3 章 3.2 節、4 章 4.2 節、
5 章 5.1 節、5.2 節、6 章

目 次

1　D-Case の概要

本書は、DEOS プロジェクトで研究開発中の D-Case 手法[1、2、3、4]とツール[5、6]の紹介を行い、実際の利用方法をできるだけわかりやすく、実践的な解説を試みた。

D-Case は、システムのディペンダビリティをシステムに関わる人たち(利害関係者、ステークホルダ)が共有し互いに分かり合い、そのディペンダビリティを社会の人々にわかってもらい、説明責任を果たすための手法とツールを目指している。

D-Case は、欧米で近年高い安全性が要求されるシステムの開発運用において、提出が義務付けられるまでに普及しているセーフティケース[7]をもとに研究開発が始まった。セーフティケースとは、テスト結果や検証結果をエビデンスとしてそれらを根拠にシステムの安全性を議論し、システム認証者や利用者などに保証する、あるいは確信させる(assure)ためのドキュメントである。近年、安全性だけでなくセキュリティやディペンダビリティなども対象として使われはじめている。その場合、セキュリティケースや、ディペンダビリティケースと呼ばれる。これらを総称してアシュアランスケース(Assurance Case)と呼ばれる。セーフティケースにおいて用いられるエビデンスを元にした議論、保証という考え方はこれからのシステムのディペンダビリティを達成するために重要であると考え、D-Case の研究開発を始めた。

現在のセーフティケースでは、システム供給者、第 3 者コンサルティグ会社、システム利用者(国防省など)の間のコミュニーケーションなどに使われるが、主には認証のために提出されるドキュメントとして使われてきた。

[15]にあるように、規模が拡大し、ネットワーク化したこれからの情報システムは開発・運用を通し、多くの利害関係者やシステムと連携して、ディペンダビリティを達成する必要がある。DEOS で開発している他のツールやランタイムシステムなどとの連携のための基礎研究([8,9,10]など)と開発(詳しくは DEOS ホームページを御覧ください[1])に加えて、特に、企業の方との議論や記述実験を通して、以下の 3

[1] http://www.dependable-os.net/osddeos/index.html

点が実用化のために重要であると考えた。

- 一般の企業の方にとってわかりやすい入門書や講習の開発
  セーフティケースはこれまで高い安全性が求められるシステムに対して、高度な専門知識を持つコンサルタントなどによって書かれてきた。そのため、セーフティケースのガイドブックなどは、安全性分析など高い専門知識を前提とされたものがあるだけであった。これからのシステムのディペンダビリティは、一般の企業の多くの方が参加されなければ達成できない。そのためには、わかりやすい入門書や講習が必要であると考えた。
- 一般の企業の方にとって使いやすい、ニーズに即したツールの開発
  セーフティケースが普及し始めてまだ日が浅いこともあってか、ツールはイギリス Adelard 社の ASCE ツール[2]などいくつかあるだけである。また ASCE ツールなどは、主に認証ドキュメントを作成するためのツールであり、他の開発ツールとの連携が容易ではなく、企業の方の実際のニーズに即したツールにはまだなっていません。DEOS で開発中の、障害対応スクリプトである D-Script やランタイム環境である D-RE との連携に加えて、企業の方が使いやすい、ニーズに即したツールが必要であると考えた。
- 記述、応用例の充実
  セーフティケースの問題の一つは、企業の重要な情報を含むことから、なかなか実際の例が表に出てこない点がある。しかしそれでは一般の、特に日本企業の方に具体的なイメージを持っていただくことは困難である。わかりやすく、具体的な記述例や応用例が必要であると考えた。

我々はこれまでの基礎研究・開発に加えて、上記 3 点に着眼し、広く企業の方に使っていただけるための活動を本格化している。その一環として、2012 年 9 月 14 日に、名古屋で第 1 回 D-Case 実証評価研究会を開催した。30 名以上の企業の方にご参加いただき、活発な議論ができた(図 1 がその様子である)。図 1-1 がその様子である)。ご興味のある方は、ぜひご参加いただきたい。2 回目も 2012 年 12 月 20 日に行うことができ、30 名を超える多数の参加者があり、好評であった。D-Case 実証評価研究会の内容、予定などは D-Case のホームページ(http://www.dcase.jp)をぜひ御覧いただきたい。

[2] http://www.adelard.com/asce/choosing-asce/index.html

図 1-1　第 1 回 D-Case 実証評価研究会の様子

1.1 D-Case について

D-Case はディペンダビリティ合意形成の議論のための手法およびツールである。開発、運用を通じて活用される。ライフサイクルの各フェーズで生成されるドキュメントを元にして、GSN(Goal Structuring Notation) [11]と呼ばれるセーフティケースの表記法の一種を拡張した表記法によって基本構造を記述する。図 1.1-1 に D-Case の例を示す。これは簡単な例であるが、実際のシステムで書くのは結構難しい。本書では、D-Case を実践的に、簡単に記述し効果的に活用するための説明を行う。

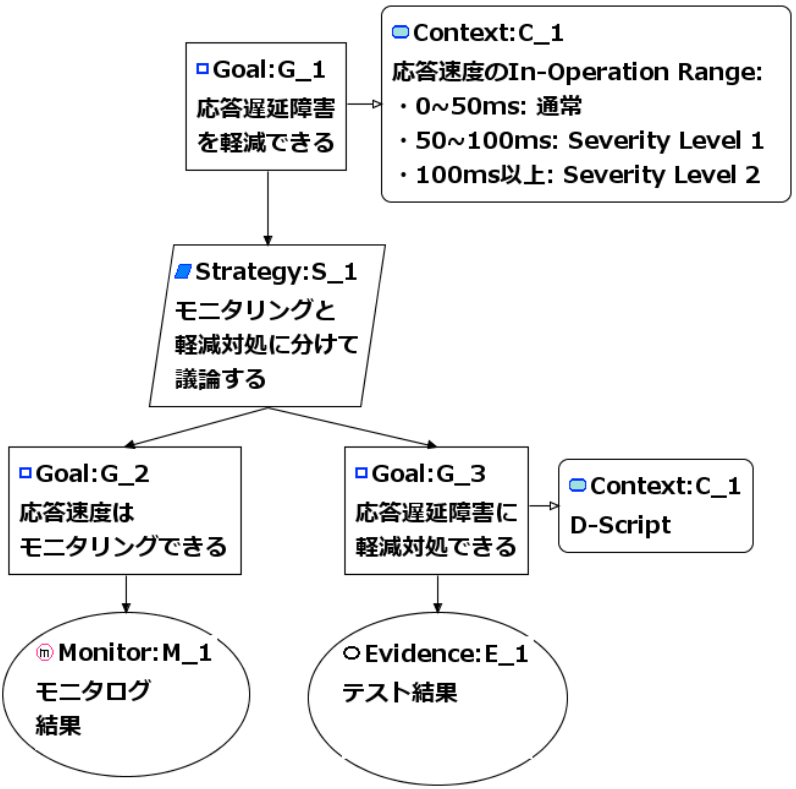


図 1.1-1 D-Case の例

1.2 D-Case の必要性

ディペンダビリティとは、「アベイラビリティ(可用性)性能及びこれに影響を与える要因、すなわち信頼性性能、保全性性能及び保全支援能力を記述するために用いられる包括的な用語である」と、JIS Z8115 (2000)で定義されている。また、Avizienisらはシステムのディペンダビリティを以下の５特性で定義している[12]。

- 可用性: 正しいサービスを提供できること
- 信頼性: 正しいサービスを持続できること
- 安全性: ユーザと環境に破滅的な事態を生じさせないこと
- 一貫性: 不適切な変更がないこと
- 保守性: 修理・修正できる能力

Jackson は、ディパンダブル・システムのソフトウェアには、明示的主張(Explicit claims)、証拠(Evidence)、知識(Expertise)の３つが必要であると指摘している[13]。この理由は、システムがディペンダブルであるというためには具体的な性質を明示的に主張することと、ディペンダビリティの主張を支持する証拠を提示する必要があるからである。システムが実行条件下で重要なディペンダビリティ要求を満足することを示すために、ディペンダビリティケース(D-Case)が必要となる。

D-Case の必要性を規格からの要請、用途、期待効果の観点から説明する。

- 規格からの要請

  ISO 26262 は自動車の電気/電子に関する機能安全についての国際規格である。Part10[14]では、機能安全についてのガイドラインが説明されています。このガイドラインの 5.3 節「セーフティケースについて理解する」で、セーフティケースの表記法として、GSN と CAE(Claims- Argument- Evidence)[3]が紹介されている。このガイドラインでは、開発対象システムとしてのプロダクトについてだけではなく、システム開発やアセスメントのプロセスについてもセーフティケースが必要だと指摘している。

- 用途

  環境と相互作用するシステムや製品が持つ不確実性やリスクに対してシステムや製品が望ましい性質を持ち、危険な状況に陥らないことを確認できる。また、システムや製品の開発プロセスにおける計画や、生産物、人間活動、意思決定

[3] http://www.adelard.com/asce/choosing-asce/cae.html

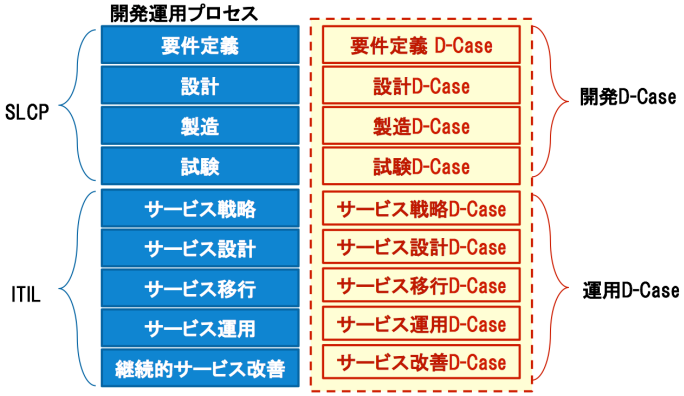
などに対する合意形成の結果を記録する。主張に含まれる不確実性を関係者が許容できるかどうかを D-Case の作成を通じて議論する。したがって、D-Case を作成した結果にも、主張が持つ性質の影響度とその不確実性を反映することになる。

・ 期待効果

D-Case の効果を列挙すると次のようになる。

- 主張するサービス水準を提供できることを示すエビデンスを提供できる
- システム異常の検出と修正を早期化できる
- 開発・運用プロセスと生産物のリスクに対する客観的な管理を推進できる
- システムのディペンダビリティの影響評価を早期化できる
- システム要求の充足性に対して、客観的なエビデンスに基づく確認プロセスを提供できる
- システム開発・運用プロセスを統合的に確認することによりプロセス改善を推進できる

システム開発・運用プロセスに対する D-Case の構成例を図 1.2-1 に示す。



参考）ISO/IEC 12207, IEEE Std 12207-2008, Systems and software engineering — Software life cycle processes
iTSMF, ITIL V3 Foundation Handbook, 2009

図 1.2-1 システム開発・運用プロセスに対する D-Case の構成例

D-Case を用いたディペンダブルな開発・運用プロセスと、現状の開発・運用プロセスを比較すると、図 1.2-1 のようになる。現状でも開発運用文書を用いてシス

テム開発や運用を効率化している。しかしこれらの開発運用文書では、ディペンダビリティについての主張や、主張が成立することを示す明示的な証拠がない。このため、システム障害やシステム改善を実施する上での活動が妥当であることを示すことが困難である。

これに対して D-Case を用いた開発運用プロセスでは、システムのディペンダビリティに対する主張、前提、証拠が明示的に記録されているので、主張が成立することを客観的に論証できる。このため、システム障害やシステム改善の際に D-Case を活用することができ、迅速な障害対応やシステム改善の妥当性を容易に確認できる。

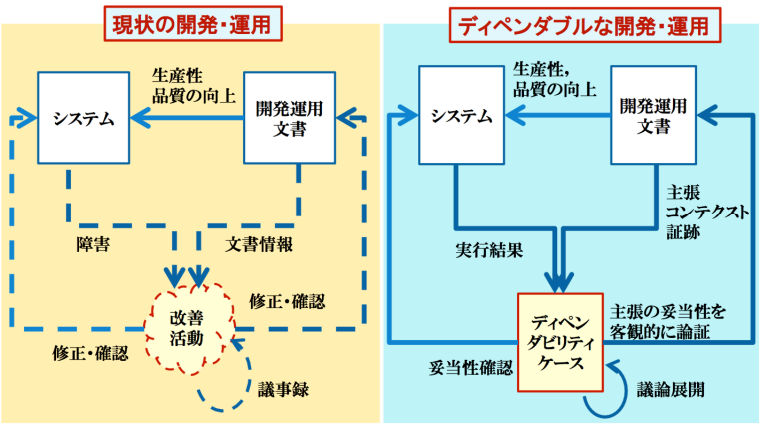


図 1.2-2 開発・運用プロセスの比較

D-Case は上記のようなメリットがあると期待される。多くの企業、研究機関との共同記述実験などを通して、実証を行なっている。

参考文献

[1] Yutaka Matsuno, Jin Nakazawa, Makoto Takeyama, Midori Sugaya, and Yutaka Ishikawa. Toward a language for communication among stakeholders. In Proc. of the 16th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'10), pages 93–100, 2010.

[2] 松野裕、高井利憲、 山本修一郎. D-Case 入門 ~ディペンダビリティ・ケースを書いてみよう!~. 株式会社ダイテックホールディング, 2012. ISBN:

978-4-86293-079-8.

[3] Yutaka Matsuno and Shuichiro Yamamoto. A new method for writing assurance cases. International Journal of Secure Software Engineering (IJSSE), Special Issue on Cybersecurity Scientific Validation, January 2013

[4] Yutaka Matsuno and Shuichiro Yamamoto. Consensus building and in-operation assurance for service dependability. In Proc. of CD-ARES, LNCS 7465, pages 639–653. Springer, 2012.

[5] YutakaMatsuno, HirokiTakamura, and Yutaka Ishikawa. A dependability case editor with pattern library. In Proc. IEEE 12th International Symposium on High-Assurance Systems Engineering

[6] Hajime Fujita, Yutaka Matsuno, Toshihiro Hanawa, Mitsuhisa Sato, Shinpei Kato, and Yutaka Ishikawa. DS-Bench toolset: Tools for dependability benchmarking with simulation and assur- ance. In Proc. IEEE DSN 2012, 2012. 8pages.

[7] J.R. Inge. The safety case, its development and use in the United Kingdom. In Proc. of ISSC25, 2007.

[8] Yutaka Matsuno and Shuichiro Yamamoto. Toward dynamic assurance cases. In Proc. JCKBSE 2012, pages 154–160. IOS Press, 2012.

[9] Shuichiro Yamamoto and Yutaka Matsuno. A review method based on a matrix interpretation of GSN. In Proc. JCKBSE 2012, pages 36–42. IOS Press, 2012.

[10] Yutaka Matsuno and Kenji Taguchi. Parameterised argument structure for GSN patterns. In Proc. IEEE 11th International Conference on Quality Software (QSIC 2011), pages 96–101, 2011.

[11] Tim Kelly, Rob Weaver (2004). The Goal Structuring Notation – a safety argument notation. In Proc. of DSN 2004, Workshop on Assurance Cases, 2004

[12] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secur. Comput., 1(1):11–33, January 2004.

[13] Daniel Jackson, Martyn Thomas, and Lynette I. Millett. Software for Dependable Systems Sufficient Evidence? The National Academies Press,

Washington D.C., 2007.

[14] ISO 26262 Road vehicles -- Functional safety

[15] Mario Tokoro, Editor. Open Systems Dependability: Dependability Engineering for Ever-Changing System, CRC Press, 2012

2　D-Case の基礎知識

2.1　ディペンダビリティについて

本節ではディペンダビリティ(Dependability)について説明する。ディペンダビリティは現在、システムの非機能要求を特に利用者の視点から議論するための中心的な概念となっている。ディペンダブルなシステムとは一言で言うなら利用者が安心し、使いたいと思えるシステムである。どのようにシステムの開発と運用を行えば、ディペンダブルなシステムになるのか、が私達の一番大きな研究目的である。

2.1.1 節においてディペンダビリティの基礎を参考文献[1]をもとに説明する。2.1.2 節では情報システムの規模の拡大が続く現代において、今後のディペンダビリティに重要になるシステム保証(System Assurance)と説明責任達成(Accountability Achievement)を説明する。D-Case は、主にシステム保証と説明責任を達成するための手法とツールである。

2.1.1 ディペンダビリティの基礎

ディペンダビリティはシステムに関する概念である。システムの定義は色々あるが、ここでは参考文献[1]の定義を以下に示す。

システム: 他のシステムと相互作用するもの。ここでいうシステムは、与えられた環境であり、ハードウェア、ソフトウェア、人間、および自然現象を伴う物質的世界などである。システム境界はシステムとその環境が接する境目である。

システムは互いにサービスを介して相互作用を行う。参考文献[1]のサービスの定義を示す。

サービス：システムが(提供者の役割によって)提供するサービスとは、その利用者から見える振る舞いである。ここで利用者とは、提供者からサービスを受ける別のシステムである。提供者のシステム境界のうちサービス提供がなされる部分を提供者のサービスインタフェースという。

上記利用者と提供者のサービスを介した関係を 2.1.1-1 に示す。

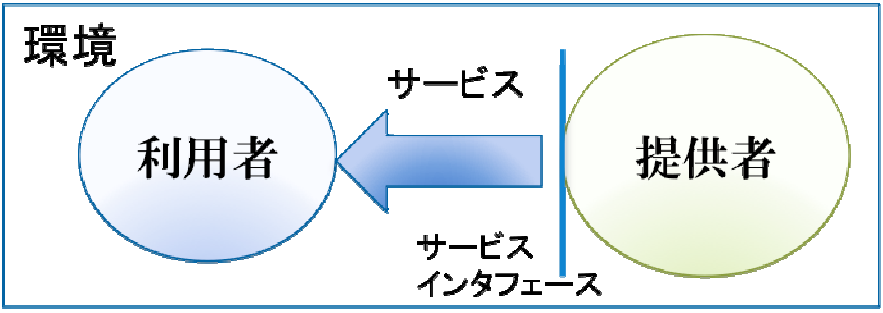


図 2.1.1-1 利用者システムと提供者システムのサービスの受け渡し

ディペンダビリティ(dependability)は、システム間の依存(depend)関係に由来する。システム A のシステム B への依存の度合いが、システム A のディペンダビリティが、どれだけシステム B に影響されるかを表す。参考文献[1]では、この依存関係などをもとに、以下のような定義を行なっている。

ディペンダビリティ: 容認できる以上の頻度と深刻さのサービス障害を防ぐシステムの能力

規格においては、例えば JIS Z 8115 では「可用性性能及びこれに影響を与える要因、すなわち信頼性性能、保全性性能及び保全支援能力を記述するために用いられる包括的な用語」と定義されている。

ディペンダビリティは JIS Z 8115 などにおけるように、可用性や信頼性など、従来のシステムの非機能要求を総合したものとして考えられてきた。しかし、もともとはシステム間の依存関係に由来し、サービス障害などに関連する概念である。可用性や信頼性は、システムがディペンダブルであるためには必要な非機能要求であると考えられるが、システムは環境に置かれ、また利用者によってサービスは使用されるため、必要となる機能、非機能などは環境や利用者によって異なる。またディペンダブルであるために必要となる機能や非機能が提供されなくなった場合(システム障害)、システムは復旧を行い、サービスを継続し無くてはならない。このことから、システムがディペンダブルであるためには、利用者、環境において望まれる性質を持ち続け、サービスを継続しなくてはならないと言える。

上記のディペンダビリティの概念や用語は、1980 年代から始まる IFIP WG “Dependable Computing and Fault Tolerance”などにおいて議論されてきた。Fault Tolerance は日本語では耐故障性であり、ディペンダビリティはもともと耐故障性の分野で考えられてきた。近年ではセキュリティとあわせて議論されている。参考文献[1]は、２０年以上に渡るディペンダビリティの議論をまとめた論文として有名である。

参考文献[1]では、ディペンダビリティを以下の３つの視点より、用語を整理している。

- ディペンダビリティ属性
  - システムがディペンダブルであるために持つべき属性
- ディペンダビリティへの脅威
  - システムがディペンダブルであることを脅かす物、出来事。
- ディペンダビリティへの脅威に対処する手段

以下それぞれを説明する。

ディペンダビリティ属性：ディペンダビリティは、従来の可用性などの非機能要求を統合概念として議論されてきた。以下の５つの属性は、参考文献[1]における定義であるが、この定義以外のもある。例えば、性能(Performance)やユーザビリティ(Usability)などを主なディペンダビリティ属性とすることも考えられる。

- 可用性(Availability)
  - 正しいサービスの即応性。利用者のサービス要求に即座に対応できること。
- 信頼性(Reliability)
  - 正しいサービスの継続性。壊れにくいこと。
- 安全性(Safety)
  - 利用者と環境へ破壊的影響をもたらさないこと。利用者に危害を加えないこと。
- 一貫性(Integrity)
  - 不適切なシステム変更がないこと。
- 保守性(Maintainability)
  - 変更と修理を受け入れられること。

次に、ディペンダビリティへの脅威について説明する。脅威は以下の３種類に分けられる。

・ 欠陥(Fault): 誤りの原因となるとみなされる・推定されるもの、こと。
・ 誤り(Error): 障害が起こりうるシステムの状態。誤りの状態になったからといって、障害が起こるとは限らない。
・ 障害(Failure): サービスが正しいサービスから逸脱する出来事。

欠陥、誤り、障害の関係を示すと図 2.1.1-2 になる。

図 2.1.1-2　欠陥、誤り、障害の関係

この、欠陥、誤り、障害の連鎖関係は、耐故障性、ディペンダビリティの研究においてよく知られているモデルの一つである。

次に、脅威への対処手段を説明する。対処手段は、参考文献[1]では以下の 4 種類に分けられている。

・ 欠陥防止(Fault Prevention): 欠陥の導入や発生を防ぐ。
・ 耐故障性(Fault Tolerance): 欠陥が存在するなかで、障害を防ぐ。
・ 欠陥除去(Fault Forecasting): 欠陥の現在の数、今後の障害、影響などを予測する。
・ (Fault Forecasting): Fault の現在の数、今後の障害、影響などを予測する

参考文献[1]では、様々な障害対策をこの 4 種類に分類している。

ディペンダビリティの属性、脅威、対処手段をまとめると図 2.1.1-3 のディペンダビリティ(とセキュリティ)の木になる。ディペンダビリティと、セキュリティは、明確に対応付けられてきてはいなかったが、参考文献[1]などで、統一して議論されるようになった。

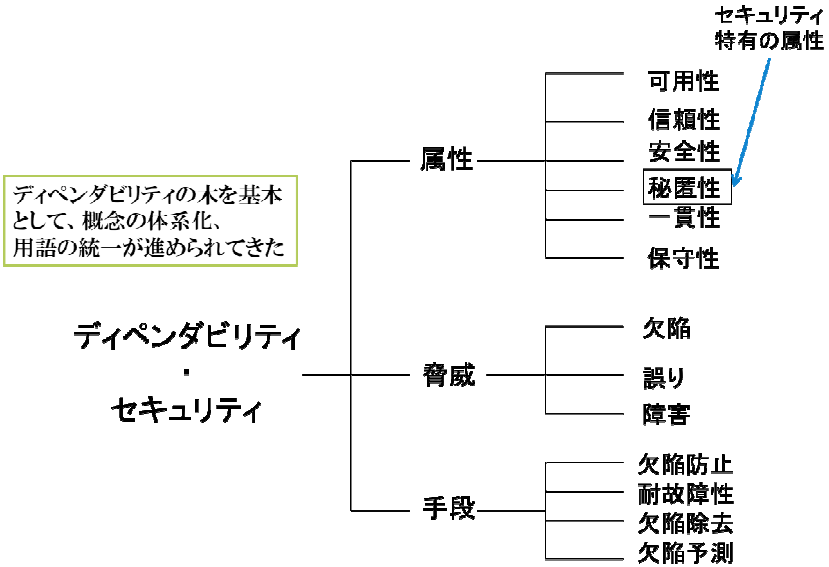


図 2.1.1-3　ディペンダビリティ(とセキュリティ)の木

ディペンダビリティの木を基本として、ディペンダビリティに関わる用語の整理が行われた。例えば、図 2.1.1-4 は欠陥の分類である。欠陥を整理し、分類することにより、システム開発者などが共通の言葉でシステム障害に対応できるようになる。

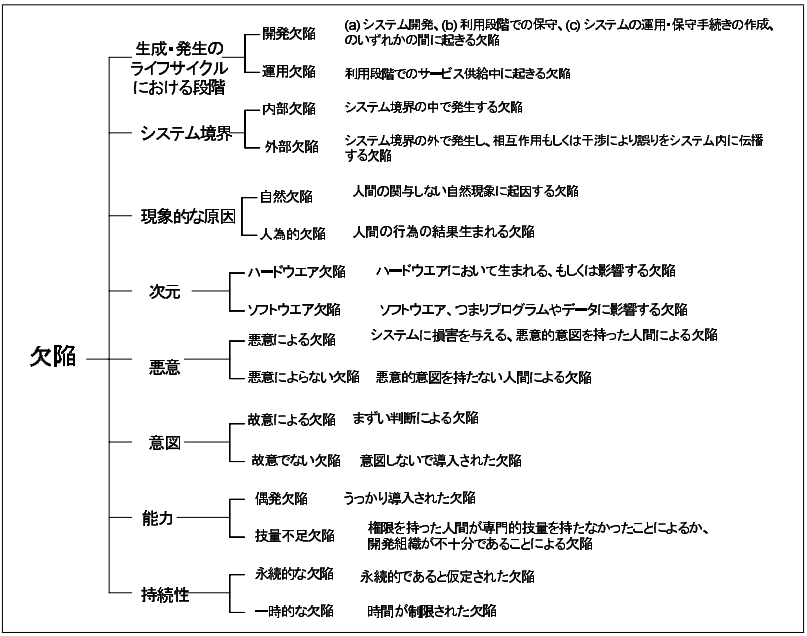


図 2.1.1-4　欠陥の基本的分類

また、図 2.1.1-5 は誤りがシステムのコンポーネント間を伝搬するモデルである。

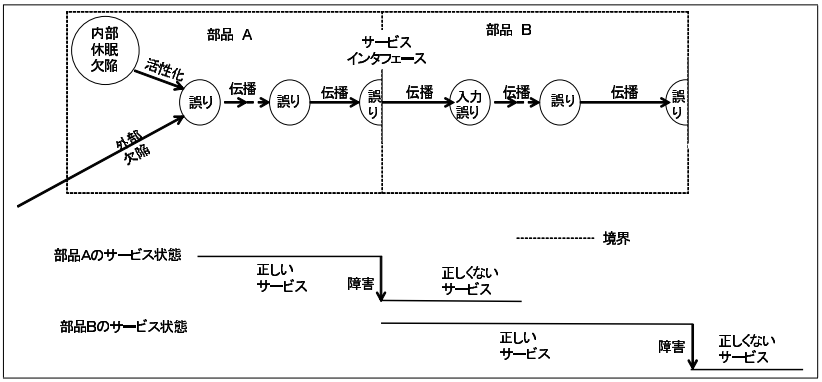


図 2.1.1-5 コンポーネント間の誤り伝搬

部品(コンポーネント)A において、外部もしくは内部の欠陥により誤り状態が遷移し、部品 A と部品 B のサービスインタフェースを超えて部品 B に伝搬すると、部品 B からみると、部品 A のサービス障害が発生したことになる。

参考文献[1]をもとに、ディペンダビリティのこれまでの研究内容を簡単に紹介した。ディペンダビリティは耐故障性研究から始まり、可用性や安全性などを統合した概念として議論されてきた。ディペンダビリティへの脅威、およびそれへの対処手段が議論の中心であり、その仕組などの工学的な一分野として発展してきたといえる。可用性などと異なる点の一つは、利用者の存在を明示し、(利用者が期待する、正しい)サービスの提供の継続に主眼を置いている点である。可用性などよりも、利用者視点にたった概念であるといえる。

2.1.2 ディペンダビリティ保証の必要性

IT・組込みシステムの重要性が増すにつれ、IT システムが我々の生活および社会において欠かせないインフラになり、さらにポータル化した。我々の生活、そして社会は便利になったが、同時に従来のディペンダビリティ、耐故障性などの考え方のみでは、対処できない問題が増えてきた。それらは以下にまとめられる(参考文献[2])。

- システムの大規模化による問題
  - プログラムサイズの巨大化
  - 多機能化
  - ブラックボックス化したコンポーネント
  - 複雑化
- 環境の変化による問題
  - 技術の急速な進化へのキャッチアップの困難さ
  - 接続システムの多様化
- 利害関係者(Stakeholder)の変化による問題
  - 要求の頻繁な変更
  - 要求や合意に対する考え方の違い

これらは、勿論従来から問題になってきたものである。しかしながら情報システムの規模のあまりの拡大により、我々はこれらの問題に対処できなくなりつつあるかもしれない。近年深刻な情報システムの障害が報告されている。例えば、２０１２年６月に発生したレンタルサーバにおける５０００を超す企業などのデータが消

失した問題は、復旧作業中に顧客のデータが他の顧客に流出するなど、さまざまな混乱を引き起こした(朝日新聞２０１２年７月７日)。

情報システムの規模の拡大により、我々はシステムのディペンダビリティを従来のようにコントロールすることが不可能になる分岐点に立っているかもしれない。ではどうすればよいか。従来の耐故障技術、さらにはテストやより精密な形式手法などはこれからますます有用になると考えられる。しかしながら、情報システムの規模の拡大が続けば、このアプローチだけではいずれ限界になると考える。すなわち、障害から逃れることはできない。我々は、障害は完全には防げないという前提にたって、これからのディペンダビリティを考察した(参考文献[3][4])。ディペンダビリティは、可用性や信頼性とは異なり、利用者を中心とした概念と言える。利用者を含む利害関係者に、障害が完全に防げないとしても、ディペンダブルと思ってもらうにはどうすればよいか。我々は以下が重要であると考える。

- システム保証(System Assurance) : システムの安全性やディペンダビリティを利用者などの利害関係者に、説明し、納得してもらうこと。
- 説明責任達成(Accountability Achievement) : システム開発、運用に当たって、利用者などの利害関係者に説明すべきことを正しく説明すること。特に障害が発生した場合、障害の原因、事後対策、利用者などへの影響などを説明すること。

従来のディペンダビリティは、耐故障性など、どちらかというと工学的、技術的側面で研究されてきた。しかしながら、工学的、技術的な側面だけではこれからのシステムのディペンダビリティを保てないとするならば、それ以外の手法を組み合わさなければならない。我々はディペンダビリティが利用者に主眼をおいた概念であることから、利用者、広く言えば利害関係者に対するシステム保証と説明責任達成が重要であると洞察した。

社会的にも、システム保証と説明責任の重要性は高まっている。たとえば、2009 年から 2010 年にかけて問題になった、アメリカにおけるトヨタ車のリコール問題は、最終的にはトヨタ車には欠陥がなかったことが報告されたにもかかわらず、大変な批判をトヨタは浴びた。この問題は政治的な背景が強かったと言われているが[4]、参考文献[5]は、トヨタに説明を行うための体制ができていなかったために問題が大きくなったと分析している。また 2011 年に発行された、自動車の機能安全規格で

[4] http://www.47news.jp/CN/201102/CN2011021001000548.html

ある ISO26262 の「最大のインパクトは、安全性の根拠を、より説明しやすくするように自動車メーカーに迫る点である」と参考文献[6]で指摘されている。ちなみに、ISO26262 では、D-Case の元になっている後に説明するセーフティケースの提出が要求されている。

ディペンダビリティは、耐故障性の研究より始まり、これまで安全性や信頼性を統合した概念として議論されてきた。しかしながら近年のあらゆる面で規模の拡大が続いている IT システムのディペンダビリティを達成するためには、従来の工学的なアプローチだけでなく、システム保証や説明責任達成が重要であると考える。本書で紹介する D-Case は、その観点から出発した。

参考文献

[1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transaction on Dependable and Secure Computing, Vol. 1, No. 1, 2004 (邦訳版は http://ocvs.cfv.jp/tr-data/PS2009-008.pdf)

[2] 屋代眞、ディペンダブルシステム組み込み OS に関する最近の話題(研究開発状況や、標準化、市場の動向など), CEATEC 2009

[3] DEOS Project Whitepaper version 3 , http://www.dependable-os.net/ja/topics/file/White_Paper_V3.0J.pdf

[4] Mario Tokoro ed., Open Systems Dependability-Dependability Engineering for Ever Changing Systems, CRC Press, In Press, 2012

[5] Michael A. Cusumano, Technology Strategy and Management - Reflections on the Toyota Debacle, Communication of the ACM, vol.54, 2011

[6] 日経エレクトロニクス 2011.1.10

2.2　ディペンダビリティケースについて

本節ではディペンダビリティケースの背景、概要を説明する。

2.2.1 ディペンダビリティケースについて

ディペンダビリティケース(Dependability Case)とはシステムのディペンダビリティを証拠(Evidence)にもとづいて議論するためのドキュメントである。その基本構造は木構造などで表される。

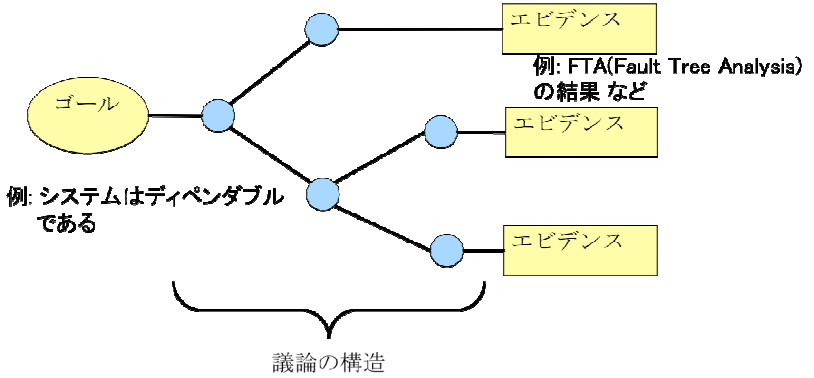


図 2.2.1-1 ディペンダビリティケースの基本構造

図 2.2.1-1 に参考文献[1]にある図をもとにしたディペンダビリティケースの基本構造を示す。ディペンダビリティケースは、「システムはディペンダブルである」など、システムが満たすべき命題をトップゴールとして、それを詳細化し、最終的にエビデンスによって詳細化された部分が成り立つことを保証する。詳細化のためのゴール分割の形を議論の構造と呼ぶ。

ディペンダビリティケースの定義は色々あるが、ここでは[1]の定義を示す。

“A documented body of evidence that provides a convincing and valid argument that a system is adequately dependable for a given application in a given environment.”

和訳すると以下のようになる。

「与えられた適用先と、環境において、システムがディペンダブルであることの確かで、正しい議論を提供する、エビデンスを元にしたドキュメント」

また、ISO/IEC 15026-2:2011 Systems and software engineering – Systems and

software assurance – Part 2: Assurance case において、

- ディペンダビリティケースの構造と内容に対する最低限の要求を規定
- ディペンダビリティケースの内容
  - システムや製品の性質に対する主張(claim)
  - 主張に対する系統的な議論(argumentation)
  - 議論を裏付ける証拠(evidence)
  - 明示的な前提(explicit assumption)
- 議論の仮定で、補助的な主張を用いることにより、最上位の主張に対して証拠や前提を階層的に構成

などが規定されている。

ディペンダビリティケースは、従来は安全性を議論するドキュメントとして、セーフティケース(Safety Case)と呼ばれてきた。議論すべきシステムの性質に応じて、～ケースと呼ばれる。例えばセキュリティを議論する場合はセキュリティケース(Security Case)と呼ばれる。ディペンダビリティケース、セーフティケースなどを総称して、アシュアランスケース(Assurance Case)と呼ばれる。ISO/IEC 15026 においてアシュアランスケースと呼ばれているように、今後アシュアランスケースという言葉が一般的になると思われる。本書はディペンダビリティを中心に議論していることから、ディペンダビリティケースを主に用いる。また、D-Case は、ディペンダビリティケースを記述するための手法およびツールの総称であるが、記述されたディペンダビリティケース自体を D-Case と書くこともある。

セーフティケースは、現在自動車の機能安全規格である ISO26262 などでも要求項目になっているなど、特に欧米では高い安全性が要求されるシステムを開発運用する際には提出が義務付けられるほどに普及している(詳細は 2.2.2 節を参照のこと)。普及した背景には、近年におきた、欧米での深刻な障害事例がある。1988 年におきた北海油田における爆発事故では、167 名が死亡した。従来、システムの安全性確認は安全性に関するチェックリストの項目を満たしているかどうかを認証者などが判定することにより主に行われていた。しかしながら、なぜチェックリストの項目を満たすと、システムが安全であるのか、明示的な議論が行われることが少なかった。北海油田における事故などの反省から、チェックリストの項目にある手順やテストのみではなく、なぜそれらの手順やテストで、対象システムの安全性が保たれるのか、明示された議論で、証拠(エビデンス)をもとに議論する重要性が認識された。セーフティケース(Safety Case)という言葉は、北海油田事故の調査報告

書である[2]が主な初出である。以来セーフティケースはイギリスを中心として欧米でシステム安全性認証(Safety Certification)で義務付けられるほど普及している。アメリカでは、2007 年に National Academy of Science から、近年の深刻な障害事例の調査報告書[3]を出版され、その中でディペンダビリティケースの必要性が言われたことが一つの契機になって、アシュアランスケースがアメリカにおいても認識されはじめた。例えば US. Food and Drug Administration (FDA)では、点滴ポンプなどの医療器具を病院に導入する際にそのセーフティケースを提出することを義務付けている[4]。

セーフティケースは普及しているが、まだ若い分野であり、様々な課題がある。基本的な課題としては、セーフティケースの記述法や評価法が未だ確立されていないことがある。そのため、ときとしてずさんなセーフティケースがあることが報告されている。有名な例としては、Nimrod 軍用機の例がある。2006 年 9 月 2 日、アフガニスタンで作戦飛行中の MR.2 XV230 が、空中給油を受けた直後に火災が発生して墜落する事故が発生した。2007 年 12 月 4 日、イギリス国防省は調査報告を発表し、墜落した機体は、給油後タンクから燃料漏れが生じており、高温空気パイプの熱によって発火、拡大して墜落に至った、と分析した[5]。Nimrod 軍用機のセーフティケースはあったのだが、上記の障害が起こりうることを、十分に議論していなかった、またセーフティケースの記述の十分性のチェックが行えるような管理体制が機能していなかったなどの問題点が報告された。セーフティが適切に管理されていれば、障害に至る欠陥を解析するための有効な材料になり得たはずであると報告された。

ディペンダビリティケースについて簡単に説明した。きちんとした議論で、適切な証拠をもとにシステムの安全性やディペンダビリティを保証することの重要性が世界的に認識されつつあると考える。これからの分野であることから、課題も多い。

2.2.2 セーフティケースの現在

本節は、DEOS のホームページで公開している資料[5]をもとにしている。

セーフティケースがどのように現在使われているのか述べる。セーフティケースが様々な規格、ガイドラインにおいて最も利用されているので、一番参考になると考えられるからである。

[5] http://www.dependable-os.net/tech/D-CaseEditor/Information_J.html

セーフティケース の作成、提出を義務付けている規格はいくつかあり、本資料においては、以下のものを主に参考にした。

・Yellow Book [6]
・EUROCONTROL [7]
・ISO 26262 [8]
・Def-Stan 00-56 [9]

ISO 26262 以外は略称であり、正式名称は参考文献を参照されたい。

Yellow Book は、鉄道システムに対して改変が行われる場合の安全管理に関するガイドラインである。

EUROCONTROL はヨーロッパの 29 カ国が加盟している、航空管制の安全に関与する組織であり、安全で機能的な航空管制管理(Air Traffic Management) を提供するものである。そこでの安全管理のための セーフティケース 作成のガイドラインが文献 [7] である。

ISO 26262 は自動車の電気、電子システムに関する機能安全規格である。機能安全の完全性の保証のために セーフティケース の作成が義務付けられている。

Def-Stan 00-56 は英国防衛省が策定した、防衛システムの安全管理システムに関する規格である。

これらのガイドラインとは別に、 セーフティケース の設計に関する資料としては、以下を参考とした。

・Modular Software セーフティケース Process [10]

上記の資料は 英国における二つの主要な航空機システムインテグレーターである BAE Systems と General Dynamics UK で構成される、Industrial Avionics Working Group が、 セーフティケース 構築のより効率的な方法論の確立のために行われた研究の成果資料である。

本資料においては、セーフティケース と、セーフティケース レポートと呼ばれる成果物を明確に区別する。セーフティケース は対象システムの安全性を保証するための議論の構造を意味するのに対して、セーフティケース レポートの定義としては以下のものを採用する。

*A **Safety Case Report** is a key deliverable that summarises the Safety Case at a particular instant in time. It provides **assurance** to the Duty Holder that safety is being managed effectively, highlights areas of safety-related project risk requiring management attention and gives stakeholders visibility of the status of the Safety Case.*

([9]、page 9)

セーフティケース と セーフティケース レポート を混同して利用している例は多い(例えば、[2])。本資料においては、 セーフティケース は議論の構造であり、 セーフティケース レポート はそれを元に作成された、安全性の保証のための審査に利用される文書であるとする。セーフティケース レポート については第３章で述べられる。

2.2.2.1 セーフティケース の利用方法

セーフティケース の利用方法については、いくつかの分類方法が考えられる。本資料においては、規格やガイドラインで作成が義務付けられているかどうかを最初の大きな分類として扱うことにする。

【作成される前提条件】

- 規格、ガイドライン、法規により作成、提出が義務付けられている。
- 義務付けられていないが作成を行う。

義務付けられていない セーフティケース に関しては、記述方法などは自由であり、またどのようなドキュメントをそこから作成するかは任意であるので、本報告書においては触れない。このような利用方法の目的としては、従来の安全規格などにおける prescriptive (処方箋的)なアプローチが不十分であることから、目的ベースで安全に対する主張を行うために利用するものである。これは、1988 年 7 月における、北海油田における Piper Alpha 事故(167 名死亡(229 名中)、270 億円の被害)に対する、Cullen 卿による事故調査レポートにおける以下の言葉にもうかがえる。

Compliance with detailed prescriptive regulations was not sufficient to ensure safety

しかし、現在においては規格に従い、与えられたパターン(テンプレート)を用いて セーフティケース を記述すれば安全が保証される、といった傾向があり、それはある意味 prescriptive なアプローチであり、反省すべき点であると言える。

規格、ガイドライン、法規において作成、提出が義務付けられている例としては、列車システムに関する Railway Yellow book [6] 、航空管制システムについての EUROCONTROL [7]、車載組み込みシステムの機能安全規格である ISO26262 [8] 、防衛関係 Defense Standard 00-56 [9] などがある。また、これらのガイドライン、規格においては、詳細度の程度の差があるが、どのような セーフティケース を作成する必要があるかが規定されている。

セーフティケース にも様々な種類があり、規格、ガイドラインにより異なる規定が行われている。さらに、単一の セーフティケース だけを作成すれば良いのではなく、複数の性質の異なる セーフティケース の作成が義務付けられている場合もある。

例えば、 Railway Yellow Book においては、Engineering セーフティケース (変更に関する安全性)と Railway セーフティケース(組織の安全管理に関する セーフティケース)の両者の作成が義務付けられている。

また、注意しなければならないのは、これらのガイドライン(例 : [6]、[7])は、 セーフティケース 作成の規定ではなく、より包括的なシステムの安全性の保証、管理に関するガイドライン、規格であり、セーフティケース はその一部であることである。Yellow Book の目的については、以下のように記述されている。

> *The main purpose of the Yellow Book is to help you set up a process that protects you and others from mistakes and gives documented evidence (the engineering safety case) that risk is at an acceptable level.*
>
> ([6]、Vol. 1、Page4)

さらに、Yellow Book では、どのような内容について記述するかを明確に規定し

ている。

Among other things, the railway safety case must describe:

- *the operator's safety policy and arrangements for safety management;*
- *the operator's assessment of the risk;*
- *how it will monitor safety;*
- *how it organises itself to carry out its safety policy; and*
- *how it makes sure that its staff are competent to do safety-related work*

([6] 、Vol. 1、 Page5)

上記の記述により、 Yellow Book におけるセーフティケース の書き方、議論の構造を規定している。

ISO 26262 [8] では、 セーフティケース の記述は、 Part 1、Part 2、そして Part 10 において述べられている。Part 1 の セーフティケース の定義では、以下のように記述されている。

1.106

Safety Case

argument that the safety requirements for an item (1.69) are complete and satisfied by evidence compiled from work products of the safety activities during development

NOTE safety case can be extended to cover safety (1.103) issues beyond the scope of ISO 26262.

([8] Part 1、page 14)

セーフティケース の定義として標準的であると言えるが、 item というシステム(もしくはサブシステム)を表す ISO 26262 独自の用語については注意が必要である。特に注意として記されているのが、ISO 26262 のスコープを超えて利用することも可能である点である。その場合には、規格に規定された内容とは異なる議論

を独自に構築する必要がある。

ISO 26262 における セーフティケース の記述に関しては、以下のように規定されている。

6.4.6 Safety case

6.4.6.1 *This requirement shall be complied with for items that have at least one safety goal with an ASIL (A), B, C or D: a safety case shall be developed in accordance with the safety plan.*

6.4.6.2 *The safety case should progressively compile the work products that are generated during the safety lifecycle.*

([8]、Part 3、page 13)

すなわち、本要件(セーフティケース) は、ASIL (Automotive Safety Integrity Level) A、B、C、D が割り振られた少なくとも一つの安全ゴールを持つ item のために作成され、セーフティケース は 規格で規定された safety plan にしたがって開発されなければならない、という点と、安全ライフサイクルの間に生成された work product (規格で規定された提出すべき成果物)を、漸次的に集めなければならない、ということが示されている。

ここで、分かるのは、Part 3(Concept Phase)(Part 3 では安全分析、評価、安全ゴール、安全機能要件の作成が行われる) で作成される、ASIL が割り当てられた安全ゴールに関する議論に対して利用され、証拠として利用されるのは work product である、ということである。

次に考慮すべき点は、セーフティケース の作成から破棄されるまでのライフサイクルである。 セーフティケース は作成された後、レビューを受け、その完全性についての検証が行われる([8] Part 2、Table 1、page 15)。その後、受理され、改定が必要になれば保守が行われ、最後に必要ななくなると放棄される。ここから分かるのは、セーフティケース は通常のシステムライフサイクルと同等のものを持っている、ということである。

1） 作成

2） レビュー
3） 受理
4） 保守
5） 放棄

保守に関しては、変更部分に関するインパクト分析を行い、その結果に従って既に作成された セーフティケース を改訂し､それに合わせて、セーフティケース から作成されたドキュメントに変更を行う必要がある。

セーフティケース のレビューには様々な条件があるが､ここでは ISO 26262 で規定されたレビューの方法を示す。

C.2 Review of the completeness of the Safety Case (see 6.5.3)

C.2.1 *Confirmation that the work products referenced in the Safety Case are available and sufficiently complete, so that the item's achievement of functional safety can be adequately evaluated.*

NOTE The referenced work products can be the work products that are identified as relevant to support the Safety Case.

C.2.2 *Confirmation that the work products referenced in the Safety Case:*

- *are traceable from one to another,*
- *have no contradictions within or between work products, and*
- *either have no open issues that can lead to the violation of a safety goal, or have only open issues that are controlled and have a plan for closure.*

([8]、Part 2、page 21)

ISO 26262 においては、セーフティケースの中で work product (提出すべき成果物)が参照されており、レビューにおいては、それらが利用可能であり十分に完全であるか確認することで、item の機能安全の達成が、適切に評価されること、さらに、 セーフティケースにおいて参照される work product については、互いにトレーサブルであり、相互に矛盾が無く、安全ゴールを侵害するようなオープンイシューが無いことを確認する必要があること、と記されている。機能安全規格においては、対象システムの開発プロセスにおける成果物間の前方、後方トレーサビリ

ティを必要条件としている。しかし、ここで興味深いのは、認証に関わる成果物間のトレーサビリティを述べている点であり、そのために セーフティケースが利用されている点である。

次にセーフティケース の作成、利用に関して、どのような人間(ロール)が関与するかを分析する。

【Case に関わりがあるロール】

一般的には以下を考慮する必要がある。

1） **開発者**
2） **プロジェクト・マネジャー**
3） **安全マネジャー**
4） **安全アセサ、オーディター**

ここでの開発者はシステムの開発者という意味である。プロジェクト・マネジャーは、開発チームのコンタクト先であり、規格で要求される要件が満たされていることについての責任を持つ。安全マネジャーは、安全性についての責任者である。そして、最後にアセサ(オーディター)は、安全性が効率的に管理されており、安全管理が法規に沿って施行されていることを審査する者である。

様々な規格においては、これらのロールに対して厳密に規定しているものもあるし、非常に一般的に規定している場合もある。また、ロールの独立性について記述している場合もある。例えば、 ISO 26262 では、ASIL のレベルにより独立性(I0 から I3)を規定している([8] Part 2、Table 1、page 15)。Def-Stan 00-56 [9] においては、内部的に審査が行われる場合には、審査対象となる部門とは別のチームにより審査は実行されるべきであると述べている。

ここに上げた全てのロールは セーフティケースに関与している。本案件の セーフティケースから文書を作成する拡張機能に関わりがあるのは、開発者、プロジェクト・マネジャー、安全マネジャーである。どのようなフォーマットの書類(例えば、 セーフティケース レポート)が必要になるかの決定権を持っているのは、安全マネジャーであり、実際に書類を書くのは、開発者と安全マネジャーであると考えられる。以下のものは Yellow Book における Project Safety Manager の定義である。

Person responsible for safety on a project and for producing all safety-related documentation

([6]、Page A-4)

【セーフティケース作成のプロセス】

セーフティケースの作成プロセスを規定している規格も存在する。例えば、EUROCONTROL [7] においては、二つの異なる セーフティケース(Unit セーフティケース と Project セーフティケース)の作成と、開発プロセスが示されている。二種類の セーフティケースの定義は以下のものである。

・Unit セーフティケース

現在、進行中で定常的な運用が安全、かつこれからも安全を保持できること示すセーフティケース

・Project セーフティケース

現在ある安全に関連するサービスやシステム(新しいサービス、システムの導入を含む)が実施されるときに作成される。

そのプロセスで示されているのは、安全管理システム(安全計画、安全分析、安全アセスメントと二種類のセーフティケースとの関連である。

本プロセスでは、安全への考察をしてから、初期的な安全議論を構築し、それから安全計画を立案している。初期的な安全議論を構築した後は、それを利用して Project セーフティケース の構築を行う。安全計画に従って、安全のライフサイクルが回り(中央の黄色い部分)、その結果は Project セーフティケース の根拠資料になる。

安全アセスメントに関しては、FHA (Functional Hazard Assessment)、PSSA(Preliminary System Safety Assessment) と 、 SSA (System Safety Assessment)の三種類のアセスメントを安全計画、操作状況を元に行う。 SSA においては、実装、システム結合に関するアセスメントの結果は Project セーフティケース において利用され、その結果により許可が降りると、運用への移管、運用・保守に関するアセスメントが行われ、それが Unit セーフティケース に用いられる、というプロセスになっている。

これに対して、 セーフティケース 自身の設計についてより詳細に述べているのが Modular Software Safety Case Process [10] である。

[10] においては、プロセスは、「製品のライフサイクルの分析」から始まり、「ソフトウェア設計とセーフティケースアーキテクチャの最適化」に続き、「モジュラーセーフティケース の作成」へと移るプロセスを提案している。最適化は、「安全に関する議論の設計」と「ソフトウェア セーフティケース アーキテクチャの定義」という二つの部分プロセスに分解される。このような設計プロセスが可能なのは、まず セーフティケースの構造がモジュール化されているので、議論の構造の分割統治が容易であることである。

このような考え方は、当然 D-Case の作成においても利用することが可能である。例えば、 ISO 26262 の セーフティケース の考え方からすると、work product をどのように取り扱うかが非常に重要であり、そのためには、初期的な議論の構造は以下の図に示されるようになるはずである。

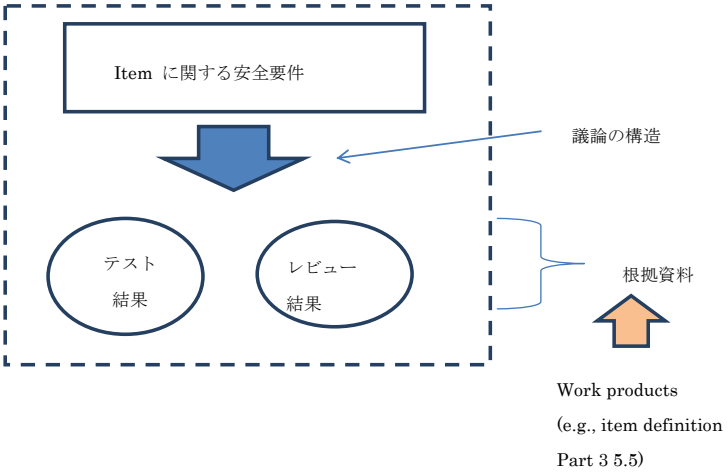


図 2.2.2.1-1 ISO 26262 の セーフティケース の初期的な議論の構造

このような初期の議論の構造を決め、アーキテクチャを決めることで、根拠が正しいテンプレートの開発が可能となる。

2.2.2.2 アシュアランスケースの国際標準化

ここでは、簡単に現時点におけるアシュアランスケースに関する国際標準化の動向について述べる。OMG (Object Management Group) の System Assurance Platform Task Force (SysA PTF) においては、ARM (Argumentation Metamodel) FTF beta [14] と SAEM (Software Assurance Evidence Metamodel) FTF beta [15] が策定され、現在は両者を統合するための SACM (Structured Assurance Case Metamodel) が策定され、改訂が行われている。OMG は 1989 年に設立されたソフトウェアシステムに関連する規格の標準化団体であり、これまでも UML (Unified Modeling Language) や CORBA (Common Object Request Broker) などの標準化で知られている。 SysA PTF は、システム保証に関する標準化を議論する OMG の一部会である。

ここで FTF (Final Task Force) とは OMG の用語で、規格の最終稿を意味する。

ARM は議論の構造、 SAEM は根拠資料についての規格という分け方から、その統合へと規格の策定は進んでいる。これらの規格は、利用方法などについての規定をしている訳ではなく、アシュアランスケースの概念的構造を規定している。

OMG における規格は UML をモデル記述言語としてメタモデルを定義することで行われる。SAEM は 20 のクラス図から構成されており、論理構造としては、Exhibits、Fact Model、 Properties、 Evidence Evaluation、Administration から構成されている。

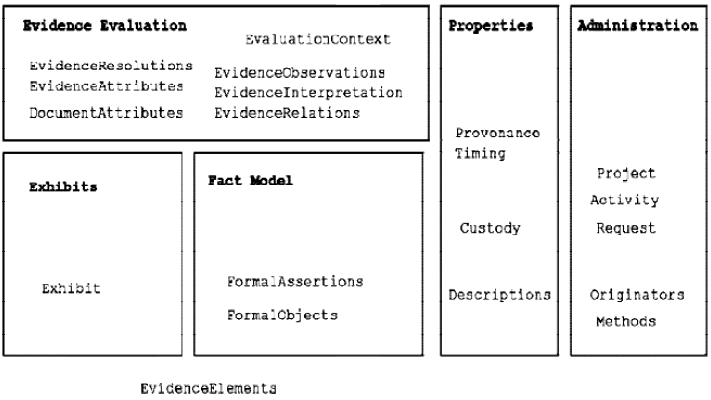


図 2.2.2.2-1. SAEM [14], p15, Figure 7.1

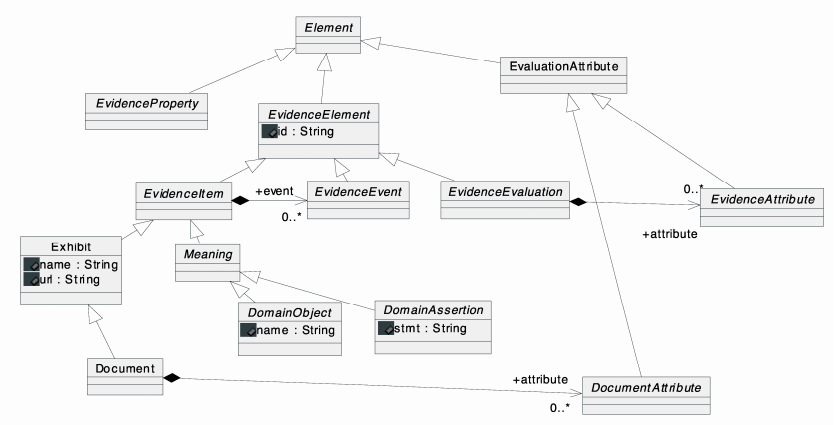


図 2.2.2.2-2 SAEM [14], p16, Figure 7.2

図 2.2.2.2-2 は SAEM における主なクラスを示している。EvidenceElement は根拠(証拠)資料として提出された物理オブジェクトを示すもので、主要構成要素になっている。EvidencProperty は根拠(証拠)の主要構成要素の属性を示している。EvidenceItem は根拠(証拠)として収集された対象を表す。Exibit は根拠(証拠)である物理オブジェクトを示す。DomainAssertion は根拠(証拠)に関する命題である。EvidenceAttribute は評価において主張された根拠(証拠)資料の特性を示す。

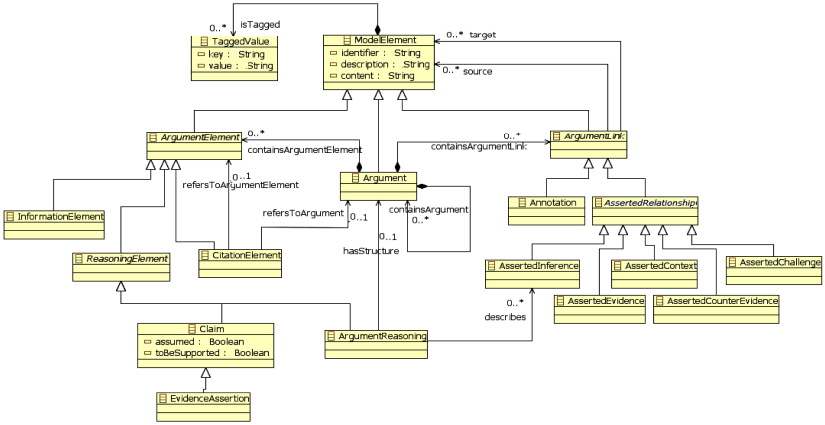


図 2.2.2.2-3. ARM [13]、p21、Figure 8.1

図 2.2.2.2-3 で示されたものが、 ARM のメタモデルであり、 SAEM とは異なり、簡潔に一つのクラス図で表されている。ここでは、Argument クラスが、構造化議論全体を統括している。ArgumentElement において、Argument 全体を構成するクラスを表し、ArgumentElement を ArgumentLink がつないで、構造化議論になるという解釈をしている。ReasoningElement クラスは、推論の基礎になるクラスであり、その部分クラスとして Claim クラス（GSN [11] でいう Goal，CAE [12] でいう Claim）と、 Claim 間の推論関係を規定する ArgumentReasoning クラスにより構成されている。根拠資料のクラスとしては、EvidenceAssertion クラスがあり、本クラスは Claim のサブクラスとして規定されている。

ARM の特徴としては、その簡潔さから拡張が容易になっている点が上げられるが、 GSN と CAE のコア部分だけを規定しており、例えば GSN におけるモジュールや、パターン言語への拡張は全く入っていない。今後、Assurance Case を支援するツールにおいては、相互互換性を考えた場合、 ARM、SACM への準拠が必要になると思われる。

2.2.2.3 ゴール構造の作成方法

本節では、D-case 図の基礎である GSN 記法による議論構造の基本的な作成方法について述べる。ここで示されている作成方法は D-case 図作成にも用いることが出来る。

高度な Case の作成方法としては、アーキテクチャの決定方法やパターンの利用などがあるが、ここではふれない。ここで述べられる方法は、 GSN Community Standard [11] に記述されているものである(元々のソースは、 T. Kelly の博士論文 [16] に記述されたものである)。

セーフティケースを作成する作成者として最初に注意すべき点は、以下の点である([11])。

・文書化された議論の明確さ
・文書化された議論の理解しやすさ
・文書化された議論の真実性(veracity)

議論の明確さとは、各々の主張と参考資料は容易に理解可能であり、議論の論理的流れが明確であることである。議論が理解しやすいとは、作成者と読者の両者が議論における主張に対して､共通理解を持っていることである｡議論の真実性とは、議論は根拠資料と推論の真実の状態を正確に反映していることである。

議論構造を作成する場合には、構造に対してトップダウンのやり方とボトムアップのやり方がある。ここでは、トップダウンによる作成方法について述べる。

【トップダウンアプローチ】

ステップ１） 支持すべき Goal (ゴール)を同定する
ステップ２） ゴールが述べられている Context(文脈)について定義をする
ステップ３） ゴールを支持するために利用される Strategy (戦略)を同定する
ステップ４） Strategy が述べられている基礎について定義する
ステップ５） Strategy を洗練化する(そして、ステップ１に戻り新しいゴールを同定する)かステップ６に移る
ステップ６） Evidence(根拠資料)を同定する。

議論の構造を作成する際に最初に行うのは、トップゴールを同定することである。トップゴールの記述は、適度なレベルの詳細度を持つ必要がある(ステップ１)。主張が真であるためには、それを支持するゴール構造を作成する必要がある。ゴールを支持する大切な情報が、どのような文脈において、その主張がされたかを示す Context ノードである。 Context ノードには、以下の情報を示す必要がある(ステップ２)。

・議論の対象となるシステムに関する情報
・システムの環境に関する情報
・議論に関する情報(用語、規格、等)

次には主張をどのように根拠付けるための戦略を決める必要がある(ステップ３)。戦略には、様々な種類がある。例えば、次章において述べられる最初のテンプレートにおいては、リスク指向の戦略が用いられている。これは、ハザード分析をした結果を用いて、各ハザード毎にその安全性を保証する議論を構築する戦略である。

他にも分割統治(divide and conquer) では、大きなゴールと小さなゴールに分解することで行われる。

ゴールと同様に Strategy がどのような文脈において用いられるかを示す必要がある(ステップ４)。もし、 Strategy によるゴールの分解の粒度が荒すぎる場合には、Strategy を複数作成し、さらなるゴールの分解を行う必要がある(ステップ５)。最後に、それ以上、議論構造にゴールを付加することが出来ない段階になったら、Evidence (根拠資料)を付け加えることで、ゴール構造の作成は終了する(ステップ６)。

このやり方は原則である。次章以降で具体的に、ディペンダビリティケースの初心者にも使いやすい手法を提案する。

参考文献

[1] Bishop, P. & Bloomfield, R. (1998). A Methodology for Safety Case Development, in Proc. of the 6th Safety-critical Systems Symposium, Birmingham, UK. Feb 1998

[2] Cullen, The Hon. Lord. (1990). The Public Inquiry into the Piper Alpha Disaster, Vols. 1 and 2 (Report to Parliament by the Secretary of State for Energy by Command of Her Majesty).

[3] Daniel Jackson, Martyn Thomas, and Lynette I. Millett, Software for Dependable Systems – Sufficient Evidence?, The National Academies Press, Washington D.C., 2007

[4] Richard Chapman, Assurance Cases for External Infusion Pumps http://www.fda.gov/downloads/MedicalDevices/NewsEvents/WorkshopsConferences/UCM217456.pdf, 2010

[5] The Nimrod Review, http://www.official-documents.gov.uk/document/hc0809/hc10/1025/1025.pdf

[6] Railtrack, Engineering Safety Management Issue 3, Yellow Book 3, Volume 1 and 3, Fundamentals and Guidance, 2000

[7] European Air Traffic Management, Safety Case Development Manual, European Organisation For the Safety of Air Navigation, Ed. 2.2, Nov 2006.

[8] ISO 26262, Road Vehicles – Functional Safety, 2011
[9] Ministry of Defence, Defence Standard 00-56, Issue 4 Publication Date 01, June 2007
[10] Industrial Avionics Working Group, Modular Software Safety Case Process, Part A: Process Definition, Oct 2007
[11] GSN Community Standard, 2011
[12] ASCAD, Adelard Safety Case Development Manual, 1998, Adelard
[13] OMG SysA PTF, Argumentation Metamodel (ARM), FTF beta, 2010, Aug
[14] OMG SysA PTF, Software Assurance Evidence Metamodel) (SAEM), FTF beta, 2010, Oct
[15] OMG SysA PTF, Structured Assurance Case Metamodel (SACM), 2012
[16] T. Kelly, Arguing Safety: A Systematic Approach to Managing Safety Cases, D. Phil Thesis, University of York (1998), http://www-users.cs.york.ac.uk/~tpk

2.3 リスク分析

社会的に影響のある事故の原因になることから、ソフトウェアの安全性が注目されている。ソフトウェアの安全性を分析確認する技術が従来から組込みシステム分野で研究されている。

ソフトウェアはその運用環境から要求された機能を実行するための契機に反応して運用環境に対して応答結果をもたらす。このとき、ソフトウェアが運用環境に対して危険な結果をもたらすことがなければ、ソフトウェアは安全であるといえる。そこでソフトウェアが安全であること、すなわち環境に対して危険でないことを分析して確認する技術が必要になる。このとき環境に対してどの程度の危険性があるか、危険性に対する対策が十分であるかなどを分析することも重要になる。

以下では、まずハザードについて述べ、主なリスク分析手法として、故障木解析、故障モード解析、イベント木解析、原因結果解析、HAZOP を紹介する。

2.3.1 ハザード

システム障害を発生させるハザードの構成要素には、①危険な要素、②契機、③対象と脅威がある。ハザードに対する刺激がシステムの危険要素で発生すると、それが契機となって、脆弱な人やモノが損傷を受けることで、不幸な重大事象が起こることになる。

たとえば、爆発物に対して不注意な取り扱いによって爆発による大惨事が発生する。

2012 年秋に、業務用洗剤をコーヒー飲料のあきカンにいれて家に持ち帰ろうとしたところ、地下鉄車内で、その洗剤が爆発して、同じ車両にいた乗客がやけどを負うという事故が発生した。この場合、危険な要素は業務用洗剤、契機はコーヒー飲料のあき缶に入れて地下鉄車内に持ち込んだこと、対象と脅威は、洗剤が爆発して乗客がやけどを負ったことである。

表 2.3.1-1　ハザードの構成要素[1]

要素	説明	例
危険な要素 Hazardous Element	ハザードに対する刺激を発生する危険要素	爆発物
契機 Initiating Mechanism	ハザードが発生する契機となるトリガイベント 休止状態から活性状態への遷移	不注意な操作
対象と脅威 Target and Threat	損傷を受けやすい脆弱な人やモノ 不幸な事象の重大性を記述	爆発による大惨事

2.3.2 故障対策技術

ソフトウェアの故障分析技術として、ハードウェアに対する故障分析技術が適用されている[2]。主な故障分析確認技術を方法、入力、出力、使用される知識の観点から比較すると下表のようになる。

表 2.3.1-2　主なソフトウェア高安全性分析確認技術

技法	方法	入力	出力	知識
FTA	ハザード原因を AND OR 論理で分析 ①システム定義 ②故障木をトップダウン作成 ③定性分析 ④定量分析	トップ故障コード 設計	故障木 分析結果 原因事象	トップ故障
FMEA	①コンポーネント定義 ②コンポーネント故障 ③コンポーネント、システム影響分析 ④故障モード結果確率、深刻度分析	詳細設計	FMEA 分析表 (コンポーネント、故障率、故障モード、モード別故障割合、影響)	故障モード 故障率
ETA	故障対策を成功と失敗により確率的に分析 ①システム定義 ②イベント木をトップダウン作成 ③分岐を追跡して事故をモデル化	潜在故障 設計 分岐確率	イベント木 事故モデル	潜在故障
CCA	①重大事象を定義 ②重大事象の原因結果図を作成 ③重大事象の潜在的影響を伝搬分析	重大事象 判断要素	原因結果図	重大事象 判断知識
HAZOP	①期待する運用意図 ②意図からの潜在的逸脱 ③逸脱の原因 ④逸脱の結果	設計	HAZOP 分析表 (ガイドワード、逸脱、原因、結果)	ガイドワード

以下では、これらの手法について紹介する。

（1） FTA(Fault Tree Analysis) 故障木分析

故障木分析では、ハザード原因を AND ノードと OR ノードから構成される木構造で論理的に分析する。

まず、システムを定義することにより、基本事象を整理する。次いで、発生する脅威をトップ事象として、その原因を探索することにより、故障木を作成する。作成した故障木に基づいて、定性分析と定量分析を実施できる。

図 2.3.2-1 に、列車事故に対する故障木の例を示す。ここでは、危険な現場での速度超過と危険な自然現象下での列車の運行が事故原因であるとした。危険な自然現象の下で列車を運行する原因としては。危険な自然現象であると認識していないか、危険な自然現象対策について考慮が不十分であることが考えられる。これに対して、制限速度を越えて危険な現場で列車を運行して事故になる原因としては、危険な現場に ATS を設置していないことに加えて運行速度制限が遵守されないことが考えられる。

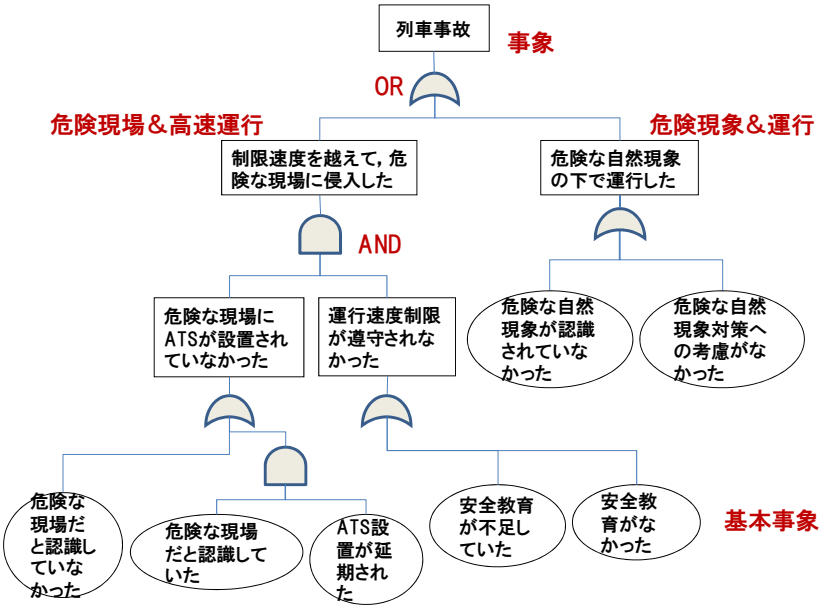


図 2.3.2-1 故障木の例

（２）　FMEA(Failure Modes and Effects Analysis) 故障モード分析

故障モード分析では、コンポーネント定義に基づいて、コンポーネントごとに、コンポーネントの故障モード、故障確率、故障モード結果確率、システムへの影響、深刻度を分析する。

表 2.3.2-2 に示した FMEA 分析表では、運転士も列車の安全運行システムでは、重要な要素であるとして故障モード分析の対象としている。また危険な現場には ATS(自動列車停止装置)が設置されることからコンポーネントとして分析対象としている。ATS の故障モードで「なし」としているのは設置すべきなのに、設置していない状態を示している。また ATS が設置されていてもスイッチが切れている故障モードを「スイッチオフ」で示している。

表 2.3.2-2　FMEA 分析表の例

コンポーネント	故障率	故障モード	割合	影響
運転士	9×10^{-2}	速度超過	５０	5×10^{-3}
	9×10^{-2}	信号見落とし	５０	5×10^{-3}
ATS	1×10^{-4}	なし	９０	5×10^{-3}
	1×10^{-4}	スイッチオフ	１０	5×10^{-3}

なお、上述の数値は、確率を記入することを例示するために用いたものであり、実際のデータではないことを注意しておく。

(３)ETA :(Event Tree Analysis) イベント木分析

イベント木分析では、故障対策の成功と失敗についての分岐を追跡して事故をモデル化できる。すなわち、システムの定義にもとづいて、システム事故の契機となるイベントから出発して、システムで用意されている故障対策が成功するか失敗するかによって、故障木をトップダウンに作成する。

このようにイベント木分析では、故障対策の成功と失敗に対する確率を付与することにより確率的に事故の可能性を分析できる。

図 2.3.2-2 に示したイベント木の例では、危険な現場を通過するときに、ブレーキと ATS による故障対策が時間順序に従って、成功と失敗によって分岐している。危険な現場でブレーキが利かず ATS も機能しないとき、脱線事故になることが分かる。

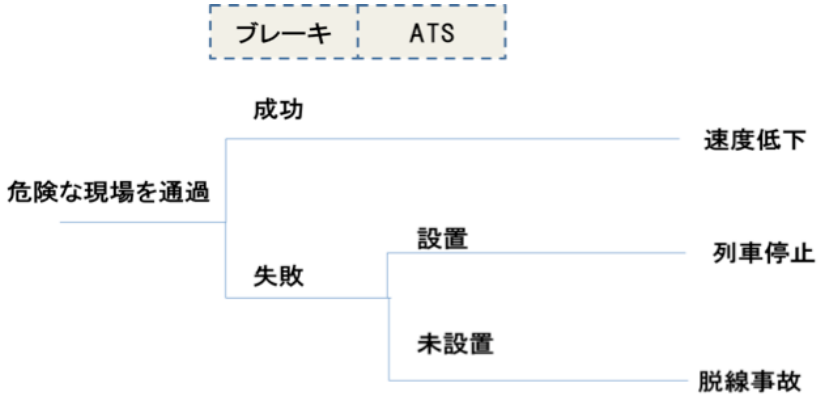


図 2.3.2-2　イベント木の例

(4)CCA (Cause Consequence Analysis) 原因結果分析

原因結果分析では、まず重大事象を定義する。次いで重大事象に対する原因結果図を作成することにより、重大事象の潜在的影響を伝搬分析することができる。原因結果図では、故障対策要素が有効に機能するかどうかの判断結果に対する原因を AND ノードと OR ノードを用いることにより原因を論理的に探索できる。

図 2.3.2-3 に示した原因結果図の例では、危険現場での列車運行に対するクリティカルな重大事象として「速度が高すぎる」を抽出している。この重大事象に対する故障対策である ATS が動作するかどうかの判断を考える。ATS が動作しない原因として、ATS が未設置であるか、設置した ATS が故障している場合があることが分かる。

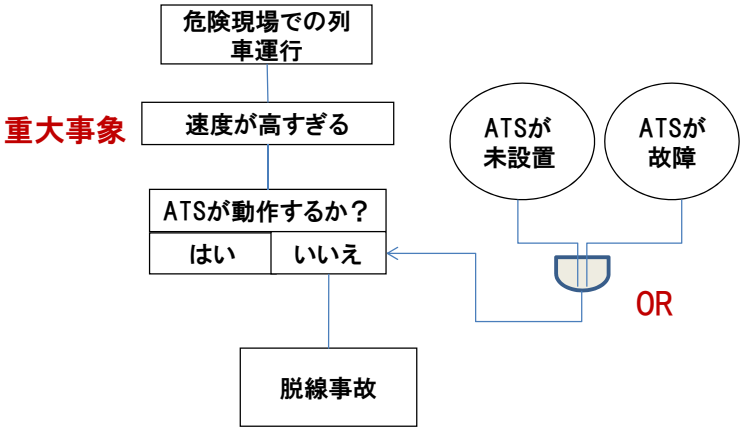


図 2.3.2-3　原因結果図の例

(5)HAZOP(Hazards and Operability Analysis)

HAZOP では、システムの期待する運用意図に対して、ガイドワードを用いて、運用意図からの潜在的逸脱を抽出する。次いで、逸脱の原因と逸脱の結果を分析する。

ガイドワードの例として、「なし」、「過剰」、「縮小」、「対象以外」、「一部」、「逆」、「異なる」などがある。

HAZOP 分析表の例を表 2.3.2-3 に示す。この例では、危険な現場での速度に対して、ガイドワード「過剰」を用いることにより、逸脱「危険な現場での速度超過」を導出する。この原因と結果として、それぞれ、「安全教育不足」「脱線事故」を抽出している。

また、危険な現場への ATS 設置に対して、ガイドワード「なし」を用いることにより、「危険な現場への ATS 設置漏れ」を導出する。この原因として、それぞれ、「安全意不足予算不足」と「脱線事故」を抽出している。この結果として、「脱線事故」を抽出している。

表 2.3.2-3　HAZOP 分析表の例

ガイドワード	逸脱	原因	結果
過剰	危険な現場での速度超過	安全教育不足	脱線事故
なし	危険な現場への ATS 設置漏れ	安全意識不足 予算不足	脱線事故

2.3.3 まとめ

HAZOP はソフトウェアが外部環境に与えるハザードに着目した分析技術である。これに対して FTA、FMEA などはハードウェアやソフトウェア内部の故障に着目した分析技術である。本書で解説するディペンダビリティケースの基礎になっている GSN(Goal Structuring Notation)はシステムが環境やユーザに対して安全であることを保証するための手法である。

この観点から下図に示すようにソフトウェアの高安全性分析技術を分類できることが分かる。ここで、追跡性(Tracability)や設計理由(Design Rational)の研究はソフトウェア内部に着目した保証技術である。

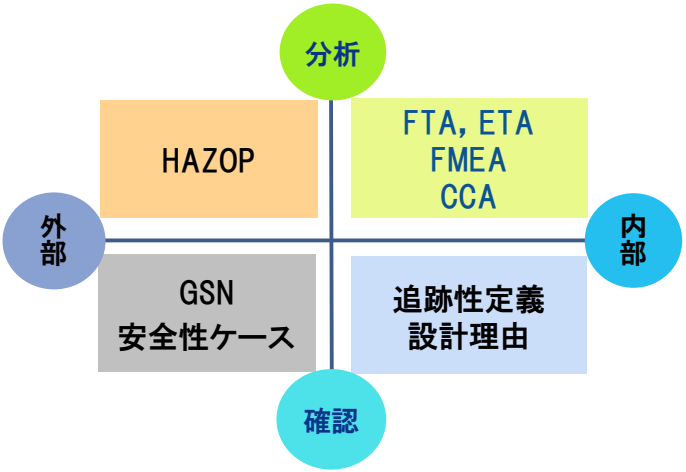


図 2.3.3-1　ソフトウェア高安全性分析保証技術の関係

表 2.3.3-1　分析表の構成の比較例

項目	FMEA	HAZOP
対象	構成要素 システム階層、タスク階層	パラメータ システム状態
根拠	故障モード 過去の故障実績	意図からのパラメータの逸脱 ガイドワード
影響	上位システムへの影響	外部へのハザード
重大性	故障の重大性	外部への影響の重大性
原因	故障の原因	ハザードの原因
対策	影響の緩和策	影響の緩和策

高安全性分析技術の手順には表 2.3.2-1 から分かるように、リスク(ハザード)識別とその分析、影響の定義、安全性確認というような共通点がある。これをまとめると図 2.3.3-2 のような高安全開発のプロセスにまとめることができる。

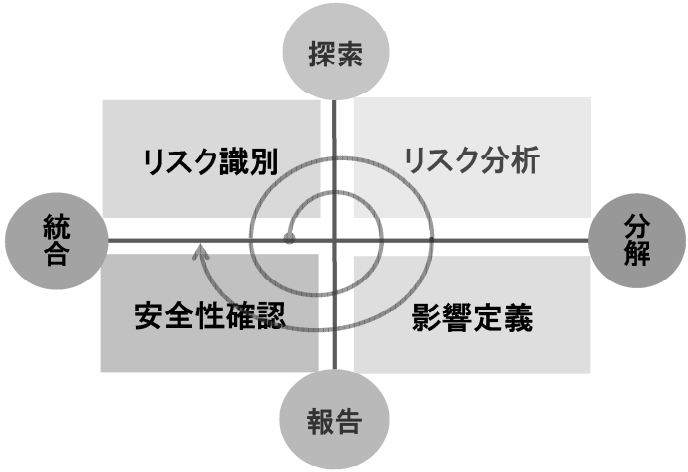


図 2.3.3-2　ソフトウェア高安全性開発プロセス

【リスク識別】

まずシステムの内部構造を分析するために、①システムアーキテクチャ②コンポーネント構成③コネクタ関係④相互作用分析などを実施する。次いで、ハザード、故障モードを抽出し、その妥当性、完全性を確認するために、①抽出準備②抽出活動指揮③抽出結果の文書化④抽出結果確認などを実施する。

【リスク分析】

リスクに対して、システムと環境への影響、重大性を分析するために、①対象要素定義②判断根拠の分析③影響分析④重大性分析⑤原因分析⑥対策定義などを実施する。

【影響定義】

リスク分析結果に基づいて、リスクの影響を定義するとともに、残存リスクを記録する。

【安全性確認】

リスク対策と安全性要求との適合性を評価し、安全性リスクを摘出するために、①リスク対策評価②リスク緩和策割付③組織準備判断④安全性要求判断⑤安全性評価確認などを実施する。

参考文献

[1] Clifton A. Ericson Ⅱ, Hazard Analysis Techniques for System Safety, John Wiley & Sons, Inc., 2005.

[2] Nancy Leveson, Safeware– System Safety and Computers, Addison-Wesley, 1995, 松原友夫監訳，セーフウェア，翔泳社，2009

3　D-Case/GSN の基礎

本章では D-Case の記法である GSN(Goal Structuring Notation)の記法の説明および演習を行う。

3.1　D-Case/GSN の基礎

D-Case は GSN Community Standard [1]をベースに、DEOS で考えられてきた拡張を行った表記法を用いる。D-Case の仕様は、モジュールなどを含めて策定中である。ここでは[1]をベースに基本的なところを説明する。

D-Case は主に以下のノードにより構成される。このうち、モニタと外部接続は D-Case の GSN からの拡張である。ノードの名前の()内は、GSN における英語による表記である。簡単な D-Case の例を示す。

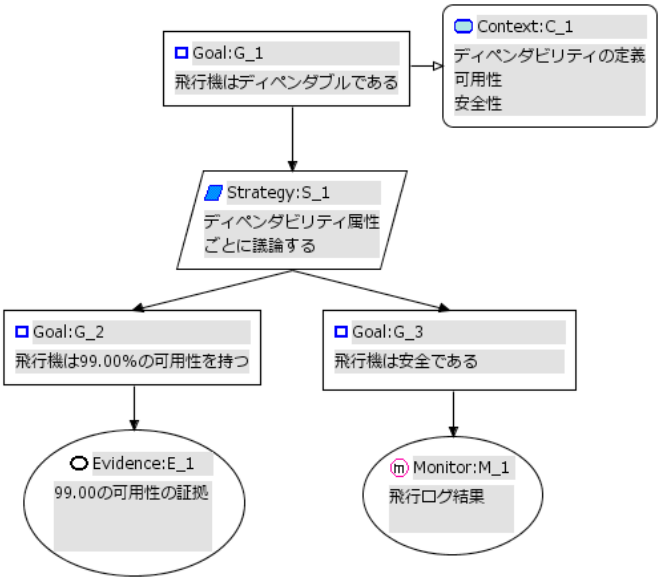


図 3.1-1 簡単な D-Case の例

A ノードの種類

・ ゴール（Goal）

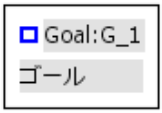


図 3.1-2 ゴール

対象システムに対して、議論すべき命題である。例えば「システムはディペンダブルである」とか「システムは適切な安全性をみたす」などである。

・ 戦略（Strategy）

図 3.1-3 戦略ノード

ゴールが満たされることを、サブゴールに分割して詳細化するときの議論の仕方である。例えば、「システムは安全である」というゴールに対して、現時点で識別されているハザードに対処できていることによって議論したいとき、戦略ノードとして「識別されたハザードごとに場合分け」を用いると、例えばひとつのサブゴールは「システムはハザード X に対処できる」となる。

・ 前提（Context）

図 3.1-4 前提ノード

ゴールや戦略を議論するとき、その前提となる情報である。例えば、運用環境や、システムのスコープ、あるいは「識別されたハザードのリスト」などである。

・ 証拠（Evidence）

図 3.1-5 証拠ノード

詳細化されたゴールを最終的に保証するものである。例えばテストや形式手法による検証結果などである。

・ 未達成（Undeveloped）

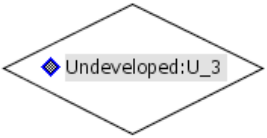


図 3.1-6 未達成ノード

ゴールを保証するための十分な議論もしくはエビデンスがないことを表す。

・ モニタ（Monitor）（D-Case の GSN からの拡張）

図 3.1-7 モニタノード

運用中のシステムより取得可能なエビデンスである。たとえばインターネットのアクセスログなど。D-Case ツールはシステムと連携し、モニタリングを用いながらディペンダビリティの保証を支援する。

・ 外部接続（External）（D-Case の GSN からの拡張）

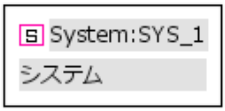


図 3.1-8 外部接続ノード(現時点の D-Case Editor ではシステムノード)

他のシステムの D-Case へのリンクである。システムは単一でディペンダビリティを保つことはなく、助けあいながら保つ。現在の D-Case Editor ではシステムノードと言う名前になっていって、他の D-Case への単純なリンクができるようになっている。

B ノードのつなげ方、矢印

・ ゴールは戦略を通して分解される。

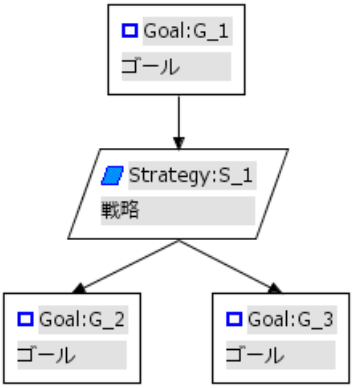


図 3.1-9 ゴールの分解の仕方

・ D-Case の葉は、証拠、モニタ、外部接続、未達成のいずれかである(現時点では外部接続ノードはシステムノード)である。未達成は、ゴールもしくは戦略ノードにリンクされる。

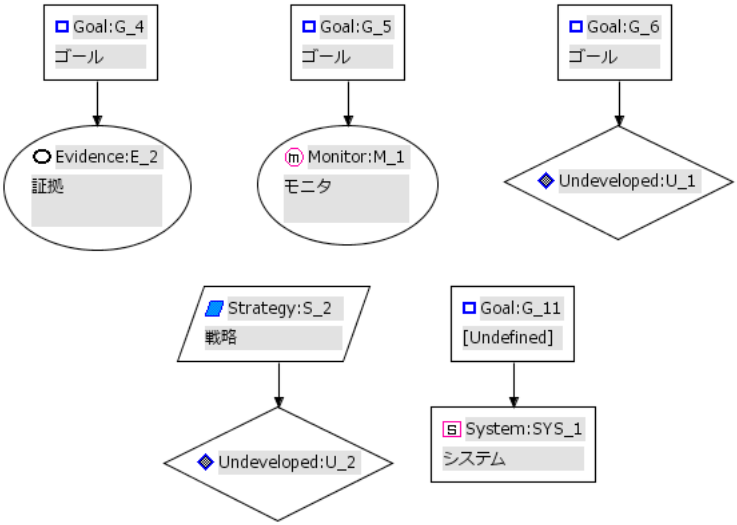


図 3.1-10 D-Case の葉

- 前提は、ゴール、もしくは戦略につなげられる。
- 矢印は２種類ある。
  - 支援リンク(Supported By): ゴールから戦略、戦略からゴール、ゴールから証拠、ゴールからモニタ、ゴールから外部接続、ゴールから未達成、戦略から未達成。例えば下図。

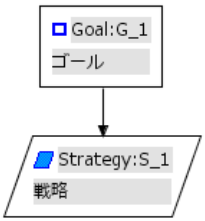


図 3.1-11 支援リンク

  - 前提リンク(InContextOf): ゴールから前提、戦略から前提をつなぐ。

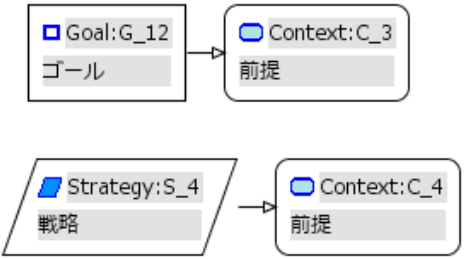


図 3.1-12 前提リンク

Quiz 1 下のノードを適切につなげて、正しい D-Case を作ってください。

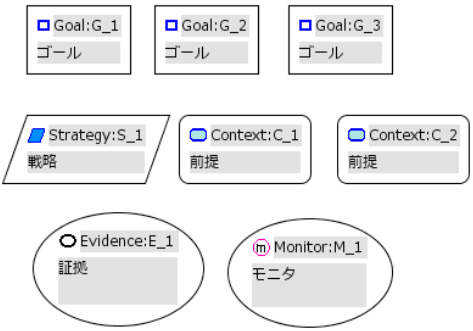


図 3.1-13 Quiz 1

答えはこの節の最後にある。

C ノードの呼び方、文章の書き方

ノードの日本語訳はディペンダビリティケースの普及が、日本ではこれからであることからまだ定まってはいない。いくつかの呼び方を紹介する。これらの中から好きな呼び方をしてほしい。ノードの文章は、自然言語で記述する(テンプレートなど、構造化された文書なども検討している)。推奨する文体を紹介する。大切なことは、文体のゆらぎをなるべくなくすことである。例えば、戦略ノードに対してある場所では、「～ごとに議論する」と書いておいて、他のノードでは「～によって説明する」などのゆらぎをできるだけ無くすことが、D-Case を見やすくする一つのコツである。

- ゴールノード(他に「主張」など)。ゴールは命題の形をしていなければならない。文体としては以下がある。この限りではないが、できるだけ文体は統一したほうがよい。
  - 「システム」が「状態」である。例　ウエブサーバはディペンダブルである
  - 「活動」が「状態」である。例　システム運用は通常運用状態である
  - 「システム」は「条件」を満たす。例　システムは国際規格の要件を満たす。
- 戦略ノード。ゴールを分解するときの考え方、観点を記述する。ゴールとそのサブゴール郡の関係を理解するための助けとなるように記述する。
  - 「観点」によって説明(もしくは論証、議論、確認)する。例　Data Flow Diagram の構造によって説明する。
  - 「観点」に分けて説明(もしくは論証、議論、確認)する。例　障害ごとに分けて説明する。

戦略ノードを使うとき注意すべきは、その前後にあるゴールとサブゴール群と、論理的な整合性を保つように使うことである。例えば、「システムは安全である」を議論するときには、戦略は複数考えられるが、もし「想定される障害ごとに議論する」を戦略ノードとして選んだ場合は、サブゴールは例えば「システムは障害 A に対処でき、安全である」などと戦略ノードに応じてサブゴールの文章を設定する。

- 前提ノード(他に「コンテクスト」など)。前提ノードには文章ではなく、前提として置くドキュメント名を置く。参照先を明確にするために、ドキュメントの中の参照部分を明記しても良い。
  - リスク分析結果ドキュメント
  - 国際規格 ISO 26262 の Part 8 の部分
- 証拠ノード(他に、「証跡」、「証憑」、「エビデンス」など)。証拠となるドキュメントの名前を書く。例えば、「テスト結果」、「レビュー結果」などである。
- 未達成ノード(他に「保留」など)。ゴールが達成することの議論がその時点でできない場合、とりあえず未達成ノードを置く。未達成ノードの文章には、なぜ現時点でゴールをサポートできないのか、サポートするために必要な条件などを記す。例えば、「ウエブサーバにおけるリスポンス時間が 3 秒以内であるための証拠が必要である」など。
- モニタノード。証拠となる、運用時のシステムから得られるドキュメントの名

前を置く。例えば、「アクセスカウントのログ結果」など。

Quiz 2 以下の文章はそれぞれ、どの種類のノードの文章にふさわしいですか(一つだけどれにも不適切な文章があります)。

- システムは十分に安全である
- 故障ごとに議論する
- システム構成図
- 故障木解析(Fault Tree Analysis)の結果
- おはようございます
- ネットワークのアクセスカウンタのログ

Quiz 3 下の D-Case は、システムの安全性を議論しています。いま、リスク分析の結果により、このシステムには障害 A と障害 B が起こりうることがわかりました。障害 A と障害 B への対策ができていることを、それぞれ分けて議論したとするとき、下の D-Case で、文章が書いていないノードの文章を書いてください。

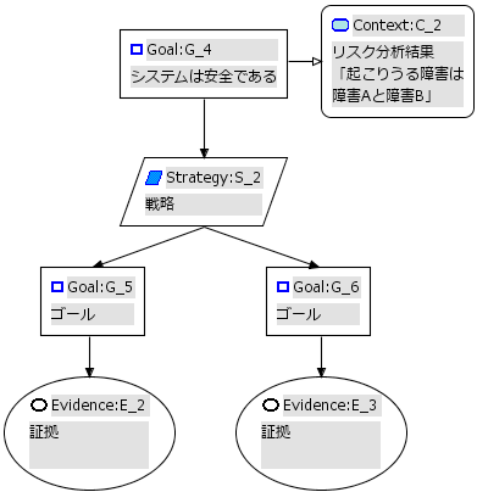


図 3.1-14 Quiz 3

Quiz の答え

Quiz 1 解答は複数あり得るが、一例を示す。

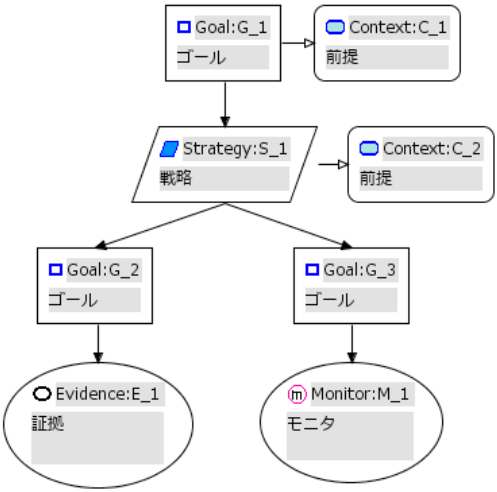


図 3.1-15 Quiz 1 の答えの例

Quiz 2

- システムは十分に安全である → ゴール
- 故障ごとに議論する → 戦略
- システム構成図 → 前提 (証拠にもなりうる)
- 故障木解析(Fault Tree Analysis)の結果 → 証拠
- おはようございます → 不適切
- ネットワークのアクセスカウンタのログ → モニタ

Quiz 3 解答の一例を示す

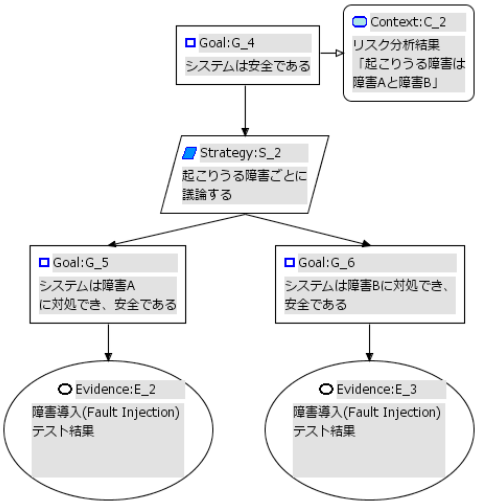


図 3.1-16 Quiz 3 の答えの例

参考文献

[1] GSN Community Standard version 1.0
http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf

3.2 GSN の基礎演習

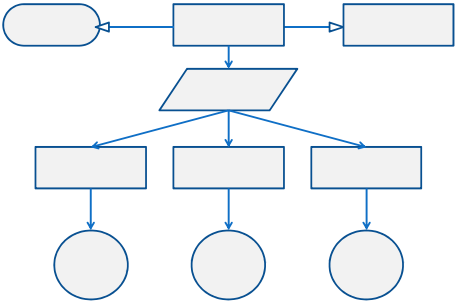
【問題１】次の表の図式要素の欄に GSN の基本となる 7 個の記号を示しなさい

名称	図式要素	説明
ゴール		システムが達成すべき性質を示す。下位ゴールや戦略に分解される
戦略		ゴールの達成を導くために必要となる論証を示す。下位ゴールや下位戦略に分解される。
コンテクスト		ゴールや戦略が必要となる理由としての外部情報を示す
未展開要素		まだ具体化できていないゴールや戦略を示す
証跡		ゴールや戦略が達成できることを示す証跡
コンテクスト関係		コンテクストとゴール、戦略との関係
ゴール関係		ゴールと戦略、ゴールと証跡の関係

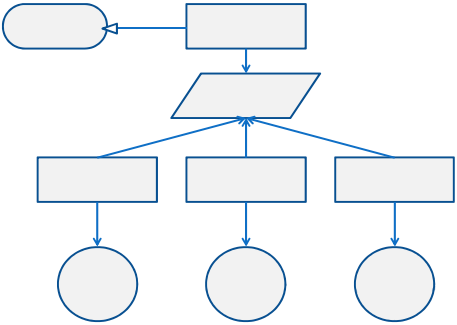
【問題２】GSN の基本的な関係を示しなさい

関係	図式表現	説明
ゴール・コンテクスト関係		ゴールのコンテクストを示す関係
戦略コンテクスト関係		戦略のコンテクストを示す関係
ゴール戦略関係		ゴールを戦略によって分解することを示す関係
戦略ゴール分解関係		戦略によって下位ゴールに分解されることを示す関係
ゴール証跡関係		ゴールが証跡によって達成できることを示す関係
ゴール保留関係		ゴールが保留されていることを示す関係
戦略保留関係		戦略が保留されていることを示す関係

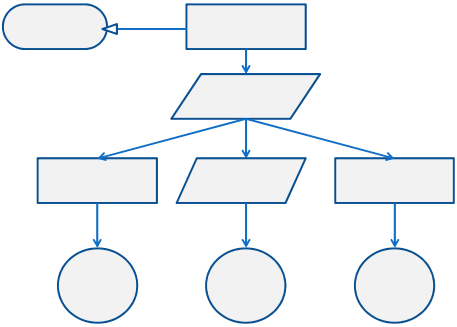
【問題３】次の GSN 構造の誤りを指摘しなさい



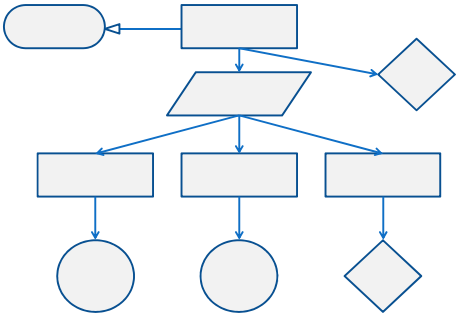
【問題４】次の GSN 構造の誤りを指摘しなさい



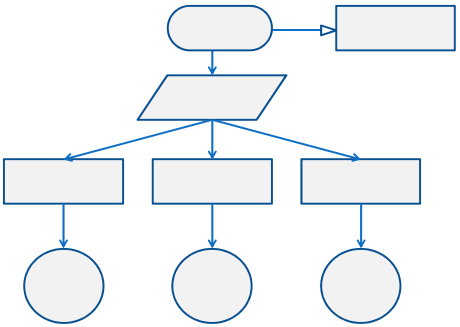
【問題５】次の GSN 構造の誤りを指摘しなさい



【問題６】次の GSN 構造の誤りを指摘しなさい



【問題７】次の GSN 構造の誤りを指摘しなさい



【問題８】次の中で、正しい記述を指摘しなさい

1. コンテクストを手順だと考えて順番に繋いでよい
2. サブゴールを手順だと考えて順番に繋いでよい
3. 戦略ノードを省いて、コンテクストやサブゴールを分岐・合流してもいい
4. 戦略を分岐条件だと考えて記述するとよい
5. コンテクストによって手順の実行状況を逐次的に表現してはいけない

【問題９】次の中で、正しい記述を指摘しなさい

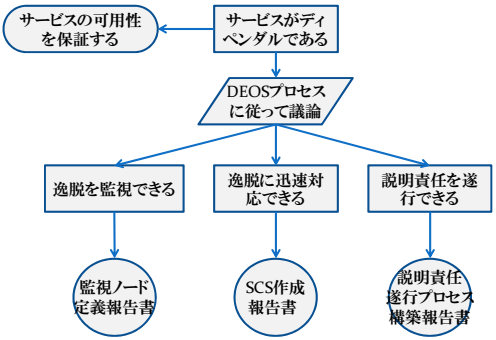
1. ゴールには実行文を書いてはいけない
2. ゴールには疑問文を書いてもよい

3. ゴールでは根拠となる理由を述べなくてはならない
4. ゴールでは主張を書くのであるから、信念や主義を述べるべきである

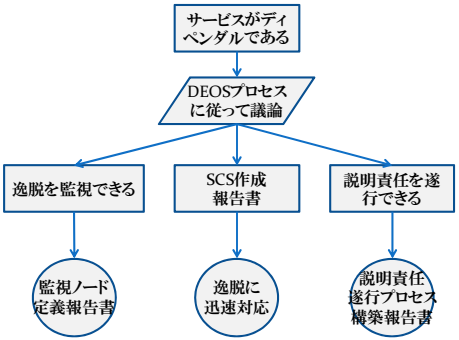
【問題１０】次の中で、正しい記述を指摘しなさい

1. 戦略では条件文を書いてよい
2. 戦略には証跡を接続してもよい
3. 戦略では主張を分解する根拠となる観点を記述しなくてはならない
4. 戦略では主張を分解するのであるから、命題として真偽が判定できるように記述すべきである

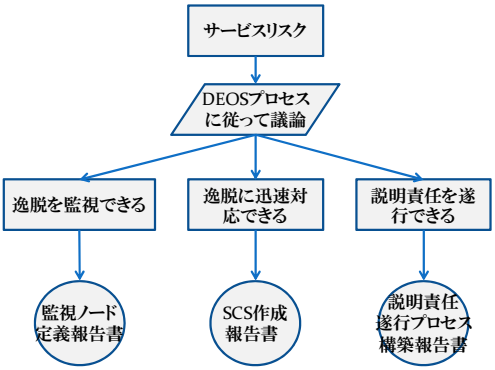
【問題１１】次の GSN の誤りを指摘しなさい



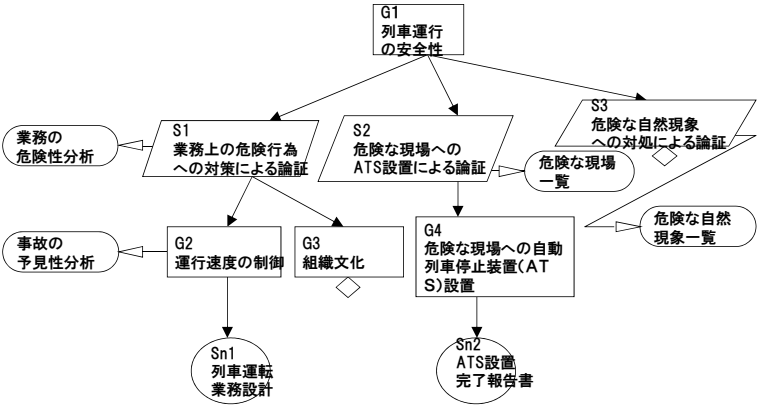
【問題１２】次の GSN の誤りを指摘しなさい



【問題１３】次の GSN の誤りを指摘しなさい



【問題１４】次の GSN で、論証が省略されている部分はどこか？　指摘して、省略された論証を復元しなさい。



【問題１５】問題１４の GSN で省略されているスコープは何か？　省略されているスコープの例を指摘しなさい。

4　D-Case 作成法

4.1　D-Case 作成法

前節で D-Case の記法である GSN について説明した。本節では実際のシステムに対してどのように D-Case を書いていけばよいか、説明する。ディペンダビリティケースを記述する手法は、これまで色々提案されているが、まだ広く認められる手法はない。本節では、名古屋大学で開発した手法を紹介する[1]。まだ発展途上である。現在、様々な企業や研究機関の方々と記述実験を行なっている。それらから得られた知見をもとに、今後改善していく予定である。また、これまでの記述実験で得られた GSN を書く上でのノウハウをいくつか紹介する。

4.1.1 D-Case 記述ステップ

[1]で提案した D-Case/GSN 記述ステップを示す。

1，　システムライフサイクルを整理し、それぞれのフェーズの入力、出力ドキュメントをまとめる
2，　入出力ドキュメントを分類する
3，　トップゴールを置く：「システムはディペンダブルである」
4，　ディペンダビリティ要求、環境情報、語彙定義を前提としてトップゴールにおく
5，　大まかな D-Case の構造を考える
6，　必要なドキュメントを前提として置く
7，　ドキュメントから D-Case のサブツリーを作る
8，　サブツリーができていない部分を典型的な議論構造を使って作る
9，　上記を必要なだけ繰り返す

それぞれのステップを解説する。

ステップ１　システムライフサイクルを整理し、それぞれのフェーズの入力、出力ドキュメントをまとめる。

D-Case はシステムのライフサイクルにおいて生成されるドキュメントをもとに

作る。このことは、D-Case は従来のシステム開発、運用で生成されるドキュメント群を置き換えるものではなく、基本的にはそれらドキュメントを用いて、システムがディペンダブルであることを示すためのドキュメントであるからである。D-Case を記述することは、従来のリスク分析や、要求分析手法を置き換えるものではない[2]。それら従来の手法より得られたドキュメントを用いて、D-Case を記述する。将来的にはまず D-Case を記述し、D-Case で要求されるドキュメントを生成するための活動をシステムライフサイクルで行うなどの手法が考えられるが、まずは D-Case の基本を確認してほしい。

簡単な例を図 4.1.1-1 に示す。

図 4.1.1-1 システムライフサイクルの例

システムのライフサイクルが上図のように定義されていたとき、それぞれのフェーズでの入出力ドキュメントは例えば表 4.1.1-1 のようになる。

表 4.1.1-1 システムライフサイクルの入出力ドキュメントの例

フェーズ	入力	出力
要求定義	利用者インタビュー文書	要求定義文書
アーキテクチャ設計	要求定義文書	アーキテクチャ仕様書、運用定義書
実装	アーキテクチャ仕様書	プログラムコード
テスト	プログラムコード	テスト結果
運用	運用定義書	運用ログ

D-Case はこれらのドキュメントをもとに生成される。実際のシステムライフサイクルでは、当然もっと多くのドキュメントがある。それらのドキュメントの中から、システムのディペンダビリティに関わるドキュメントを選択し、D-Case を記述する(すべてのドキュメントが D-Case に関係する可能性もある)。

ステップ２　ライフサイクルの入出力ドキュメントを分類する

ライフサイクルの入出力ドキュメントが、D-Case、すなわちシステムのディペンダビリティの議論にどのように関係するのか考える。これまでの経験から、D-Case に関係するドキュメントの種類は以下のようなものがあると考える。

（１）　規格。ISO 26262、 ISO 12207 など、システムが適合することを要求される国際規格など。

（２）　リスク分析結果。システムのサービス継続を脅かすリスクを解析した結果。ハザード解析、故障木解析(Fault Tree Analysis)の結果など。

（３）　ディペンダビリティ要求。例えば 99.999%の可用性などの要求。ディペンダビリティ要求はシステムごとに異なり、明確にする必要がある。上の例では、利用者インタビュー文書、要求定義文書が相当する。

（４）　システムのライフサイクルに関するドキュメント。ステップ１にあるような、システムのライフサイクルドキュメントは、システムのディペンダビリティを議論する上で重要である。

（５）　システムアーキテクチャモデル。システムのアーキテクチャは、UML などを用いて記述される。システムのコンポーネントがそれぞれどのようにシステムのディペンダビリティに寄与するか議論する必要があるときに参照する。上の例では、アーキテクチャ仕様書が相当する。

（６）　運用情報。障害はシステムの運用時に起こる。そのため、システムがどのように運用されるかは非常に重要である。システムのログ情報は、システムの現時点でのディペンダビリティの状態を知るために重要である。上の例では運用定義書、運用ログが相当する。

（７）　環境情報。システムが置かれた環境を特定しないと、システムのディペンダビリティは議論できない。

（８）　テスト、検証結果。これらは、D-Case において、証拠ノードにおいて参照される。システムのディペンダビリティを最終的に保証するものである。

（９）　プログラムコード。特に、障害対応を行うプログラムコードは、システムのディペンダビリティを議論する上で重要になる。

ステップ３ トップゴールを置く:「システムはディペンダブルである」

ステップ１、２は、D-Case を書くための準備である。ステップ３でいよいよ D-Case を書く。まずトップゴールを考える。システムのディペンダビリティはシ

ステムおよびその環境によって異なる。まず、「システムはディペンダブルである」という決まり文句をトップゴールにおき、そのシステムのディペンダビリティを、ディペンダビリティ要求に関係するドキュメントをもとに定義する。例えば、「システムは十分に安全である」や「システムにおいて、すべての識別された障害は適切に軽減されている」などが考えられる。しかしながら D-Case を書き始める時点では明確でないこともある。その場合は、決まり文句を仮置きのまま、はじめてもよい。D-Case を詳細化することによって、トップゴールが具体化することも多い。

ステップ４　ディペンダビリティ要求、環境情報、語彙定義を前提としてトップゴールにおく

トップゴールを議論する上で必要なディペンダビリティ要求の詳細、環境、語彙定義などを前提ノードに記述する。ゴールノードにはできるだけ簡単な文章を置いたほうがわかりやすい。例えばトップゴールを「システムはディペンダブルである」としたとき、そのディペンダビリティの定義を、要求定義文書の、ディペンダビリティ要求の部分を前提として参照したほうがわかりやすい。また、システムの環境情報を明確にする。特に、対象システムのスコープを明確にする。そうでないと、議論が発散してしまう。

ステップ５　大まかな議論構造を考える

従来の手法では、ゴールを演繹的に、一つ一つ設定し、ディペンダビリティケースを記述していく[3]。我々のこれまでの経験から、D-Case の大まかな議論構造をまず考えたほうが、全体を見ながら議論を詳細化できるので良いと考える。これまでの D-Case の記述実験から、我々はいくつかの典型的な議論構造を見つけてきた(詳しくは 5 章参照のこと)。例えば以下がある。

- ライフサイクルに沿った議論構造
- システム機能に沿った議論構造
- システム構造に沿った議論構造
- ワークフローに沿った議論構造
- 障害、リスク低減に沿った議論構造

これらの議論構造を組み合わせ、まず大まかな議論構造を考える。

ステップ６　必要なドキュメントを前提ノードにおく

大まかな議論構造が決まったら、その議論構造のために必要なドキュメントを前提におく。

例えば、ライフサイクルに沿った議論構造を選択した場合、ライフサイクル情報の前提ノードを戦略ノードにリンクさせる(図 4.1.2)。

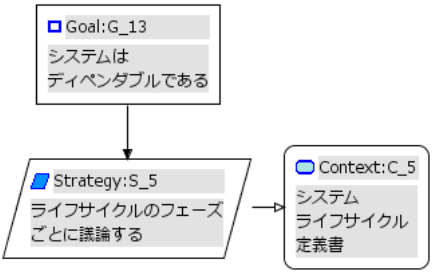


図 4.1.1-2 議論展開のための前提ノード

別な例として、障害対応に関する D-Case を考える。下図の D-Case では、障害 X にシステムが対処できることを議論している。トップゴールには障害 X の定義が前提ノードとして置かれている。障害 X を低減できることを、障害検知と対応に分けて議論している。障害 X に対応するためのプログラムコードを前提ノードとしてリンクしている。

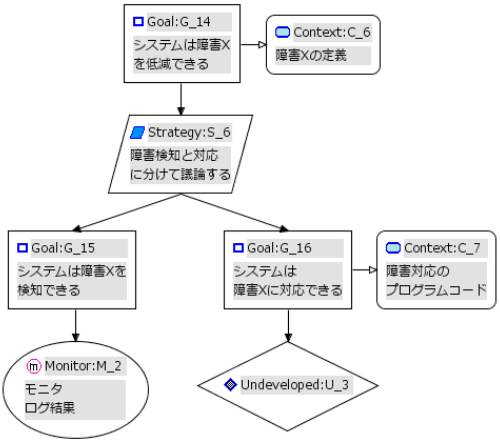


図 4.1.1-3 障害対応に関する D-Case

ステップ７　ドキュメントから D-Case のサブツリーを作る

ステップ６で前提ノードとしてリンクされたドキュメントを前提として議論を作っていく。このとき、ドキュメントの詳細を議論する必要がある場合、そのドキュメントを展開して D-Case のサブツリーを作る。多くのドキュメントは(半)自動的に D-Case に変換できる。例を２つあげる。詳しくは５章を参照してほしい。

例１　プロセス。一般にプロセスは、ゴール(目的)、1 から N 個のステップ、それぞれのステップの入力と出力により定義される。プロセスが定義されているならば、下図のように D-Case のサブツリーが自動的に生成できる(図 4.1.1-4)。

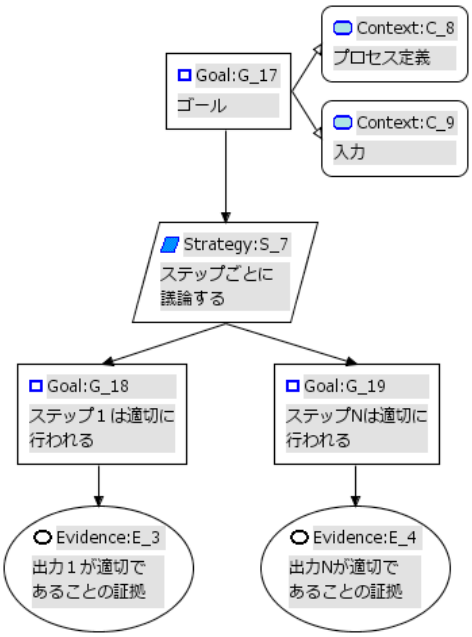


図 4.1.1-4 プロセス定義による D-Case

例２　ディペンダビリティ属性。ディペンダビリティが可用性など複数の属性により定義されている場合、それらの属性ごとにサブゴールを分けることができる(図 4.1.1-5)。

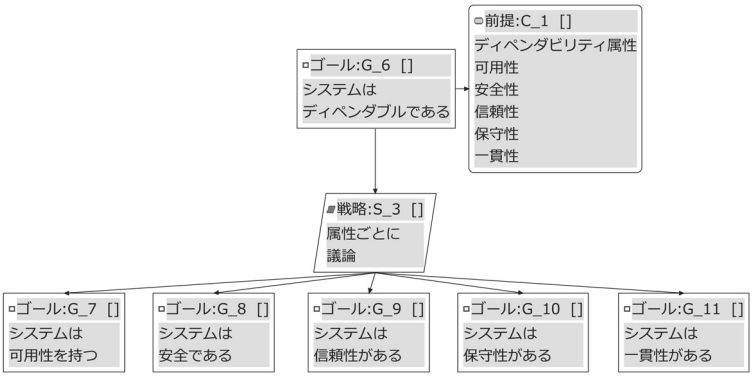


図 4.1.1-5 ディペンダビリティ属性定義による D-Case

ステップ８　サブツリーができていない部分をできるだけ典型的な議論構造を用いて作る。

大まかな構造を考え、必要なドキュメントを前提として置き、ドキュメントを展開することにより、(半)自動的にサブツリーを作成した後、まだできていない部分は、自分たちで作る必要がある。しかしながら全く独自に議論構造を考えサブツリーを作ると、我々の経験からは、他の人にわかりにくいことが多い。自分たちで議論構造を考えるにしても､これまで使われてきた議論構造から選択したほうが良い。５章で詳しく説明するが、例えば City University London の Robin Bloomfield らは､ディペンダビリティケースに用いられる議論の典型を下の表のように分類した。

表 4.1.1-2 議論分解のパターン

項番	パターン	説明
1	アーキテクチャ分解	システム構成に従って分解
2	機能分解	主張を機能構成に従って分解
3	属性分解	特性を複数の属性に分解
4	帰納分解	説明対象の場合分けによる分割
5	完全分解	説明対象のすべての要素による分割
6	単調分解	新システムによる旧システムの改善点による分解
7	修正分解	曖昧性の明確化による分解

ステップ９　上記を必要なだけ繰り返す

D-Case は形式的なものではなく、非形式的な議論により成り立っている。そのため、「よい」D-Case を一つに決めることはとても難しい(ディペンダビリティケースの定性的・定量的評価に関する研究はいくつか行われているが、広く認められた評価法はまだない。本質的に明確な基準では測れない部分がある。)　一つに決めることは難しいが、議論を尽くして、よりよい D-Case を目指すことが大切である。システムのディペンダビリティを D-Case を書くことによってステークホルダ間で理解を深めることも、D-Case によって得られることの一つであると考える。

上記のステップに沿った記述例を示す。下図は DEOS センターで開発したウエブサーバのデモシステムである(図 4.1.1-6)。

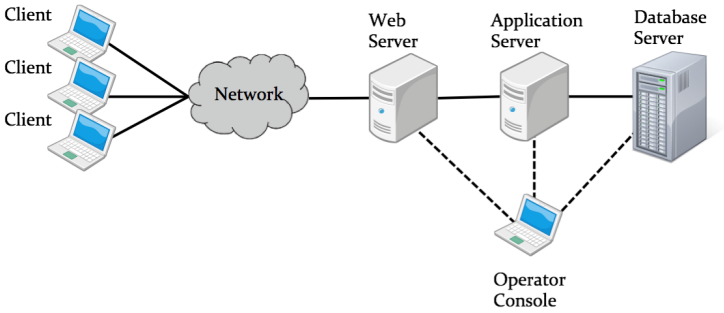


図 4.1.1-6 ウエブサーバのデモシステム

このシステムの主なコンポーネントはウエブサーバ、アプリケーションサーバ、データベースサーバよりなり、それらを運用者がオペレータコンソールを通じて管理している。利用者はネットワークを介して、それぞれのクライエント PC などでアクセスする。このシステムの D-Case を書いてみよう。以下は、DEOS センターでの記述実験をもとにしている。

ステップ１　システムライフサイクルを整理し、それぞれのフェーズの入力、出力ドキュメントをまとめる。

このウエブサーバシステムは、既存のサーバ PC を統合して開発した。そのライ

フサイクル、入出力ドキュメントは下のようであったとする。ただしデモシステムなので、いくつかのドキュメントには、仮定したものもある。ここでは特に運用ワークフロー定義文書に注目する。

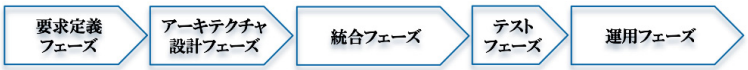


図 4.1.1-7 ウエブサーバデモシステムのライフサイクル

表 4.1.1-3 ウエブサーバデモシステムのライフサイクル入出力ドキュメント

フェーズ	入力	出力
要求定義フェーズ	ユーザインタビュー文書	要求定義文書、SLA文書
アーキテクチャ設計フェーズ	要求定義文書、SLA文書	アーキテクチャ設計文書、運用ワークフロー定義文書、リスク分析文書
統合フェーズ	アーキテクチャ設計文書、運用ワークフロー定義文書、リスク分析文書, サーバ仕様書	統合されたプログラムコード
テストフェーズ	統合されたプログラムコード	テスト結果
運用フェーズ	運用ワークフロー定義文書	運用ログ、システムログ

ステップ２　入力、出力ドキュメントを整理する。

ライフサイクルで生成されるドキュメントがシステムのディペンダビリティにどのように関わるか考え、分類する。

（１）　規格。このシステムはデモシステムなので、順守すべき国際規格などは想定していない。あるシステムが、順守すべき国際規格に適合していることを示すことは、従来からのセーフティケースの主要な使われ方であり、実際のシステムの D-Case を書く場合は重要になる。

（２）　リスク分析結果。アーキテクチャ定義フェーズ出力ドキュメントである「リスク分析文書」が該当する。

（３）　ディペンダビリティ要求。要求定義フェーズでの、「ユーザインタビュー文書」、「要求定義文書」、「SLA 文書」などをもとに、システムのディペンダビリティを考える。

（４）　システムのライフサイクルに関するドキュメント。ステップ１にあるようなライフサイクル定義文書、それぞれのフェーズの入出力ドキュメントを整理しておくとよい。

（5） システムアーキテクチャモデル。「アーキテクチャ設計文書」が相当する。システムのスコープを設定する、構成をもとに議論するときに参照する。

（6） 運用情報。「運用ワークフロー定義文書」、「運用ログ」、「システムログ」が相当する。運用ログや、システムログは、システムがどのように運用されているかの証拠として重要である。

（7） 環境情報。今回はデモシステムであったため、具体的な環境情報は考慮していなかった。実際のシステムでは運用情報と一緒にシステムがどのような環境で、どのように運用されているかを議論するために重要である。

（8） テスト、検証結果。テストフェーズの「テスト結果」が相当する。

（9） プログラムコード。「統合されたプログラムコード」が相当する。

ステップ3　トップゴールを置く：「システムはディペンダブルである」

このウエブサーバのユーザはエンドユーザであるが、システム開発会社が受注を受けるのは、ウエブサービス提供会社であったとした。エンドユーザに対する D-Case も考えられるが、ここではシステム開発会社がウエブサービス提供会社に、開発したウエブサーバがディペンダブルであることを D-Case で示すことを考える(実際のウエブサーバでは、さらにウエブサーバ運用会社なども考える必要がある)。ウエブサーバ開発会社とウエブサービス提供会社などの間では、一般に SLA(Service Level Agreement)文書で非機能要件などの取り決めを行う。このことから、トップゴールは「ウエブサーバは SLA を十分に満たす」とした。

ステップ4　ディペンダビリティ要求、環境情報、語彙定義などを前提としてトップゴールに置く。

「ウエブサーバは SLA を十分に満たす」というトップゴールを議論するための前提となる情報を、前提ノードとして置く。トップゴールに「ウエブサーバ」を議論するために必要な、アーキテクチャ設計文書を置き、スコープしているシステムが、ウエブサーバ、アプリケーションサーバ、データベースサーバであること、「SLA」の内容である「SLA 文書」を置く。ここまでで、図 4.1.1-8 のような D-Case ができる。

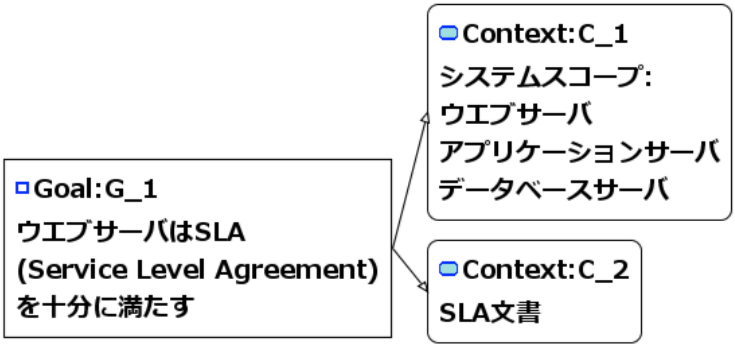


図 4.1.1-8 前提ノードをつけたトップゴール

ステップ５　大まかに D-Case の構造を考える

この例では以下のような議論を元に、構造を考えた。「サーバ PC の中身は開発していない(購入して統合した)ので技術的詳細を確認することはむずかしい。また最近の PC 技術は十分に成熟しているので、特にこのような小規模なシステムでは PC 内部の欠陥による障害はそれほど考えなくてよいだろう。最近では、サーバ運用上のミスによって重大な情報損失事故などが起こっている。運用ワークフローにそって、それぞれのステップにおいて、リスク分析により得られた、起こりうる障害に対処できるか議論することが重要なのではないか。その上で障害即応、変化対応の議論をしよう。」ここで障害即応、変化対応という議論の仕方は、DEOS プロジェクトで議論されてきたことで、障害が発生したらできるだけ即応する、またシステムとその環境になにかしらの変化が起こった時、それに対応することが、重要であるという考え方に基づいている。詳しくは DEOS プロジェクトのホワイトペーパーをご参照ください[4]。図 4.1.1-9 が、上記議論より得られた大まかな議論構造である。

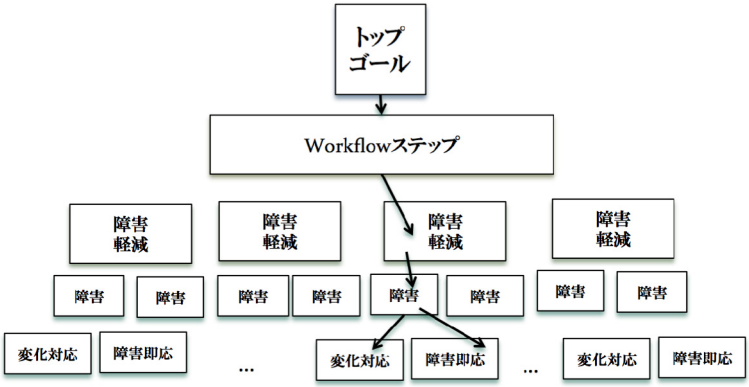


図 4.1.1-9 ウエブサーバシステムの D-Case の大まかな構造

ステップ６　必要なドキュメントを前提として置く

運用ワークフローにそって議論するためには、ワークフローを定義しているドキュメント、つまり運用ワークフロー定義文書を前提ノードに置く。運用ワークフロー定義文書では、ユーザログイン、ショッピングカート処理、クレジットカード認証、終了処理、配達、クレーム処理の 6 ステップからなり、それぞれのステップがさらに詳細なステップにわかれていると定義した。

ステップ 7 ドキュメントから D-Case のサブツリーを作る

この例の D-Case のトップゴールは、運用ワークフロー定義文書にしたがって、ステップごとに、図 4.1.1-10 のように展開できる(最初と最後のステップのみ展開)。

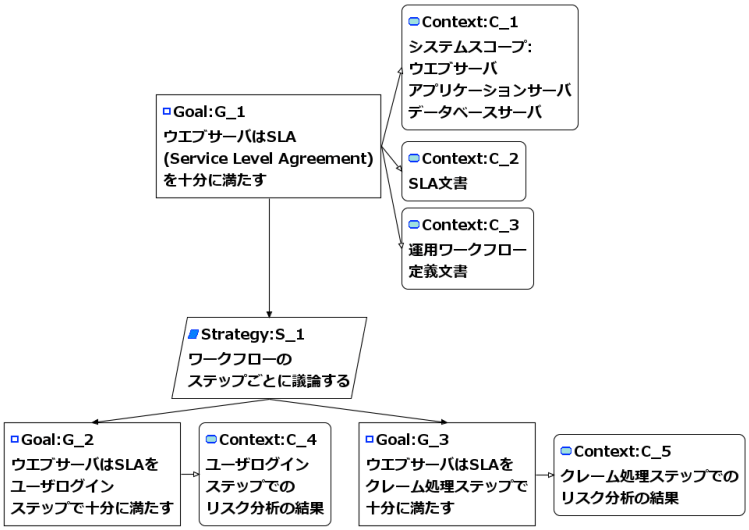


図 4.1-10 ウエブサーバシステムの D-Case トップレベル

4.1.2 D-Case 記述のためのノウハウ

これまで、デモシステムや実際のシステムを対象にして D-Case 記述実験を行なってきた。

- ウエブサーバなどのデモシステム
- エレベータ(DEOS 富士ゼロックス恩田グループ)
- センサネットワークシステム(DEOS 慶応大徳田チーム)
- ET ロボットコンテスト(DEOS 富士ゼロックス恩田グループ)[6]
- ソフトウエアバージョン管理システム(DEOS 産総研木下チーム)
- 日本未来科学館の受付ロボット(DEOS 産総研加賀美チーム)
- スーパコンピュータ運用マニュアル(DEOS 名古屋大山本グループ)
- TOGAF エンタープライズ・アーキテクチャ(DEOS 名古屋大山本グループ)
- 自動車のエンスト問題(トヨタ自動車石崎氏らとの共同研究)[7]

など。これらの成果はウエブページなどで公開する予定である。特に、富士ゼロックスによる、ET ロボットコンテストにおける D-Case の適用は、モデル部門で最優

[6] http://dcase.jp/pdf/20120914_03.pdf
[7] http://dcase.jp/pdf/20120914_02.pdf

秀賞(エクセレントモデル)を受賞した。D-Case によって、システムのディペンダビリティゴールを満たすことを議論し、そのエビデンスとのつながりを明示したことが評価された点の一つである。ディペンダビリティケースは ET ロボットコンテストで初めて採用され、最優秀賞を受賞したことは、非常に意味がある。下記ではこれらの記述実験から得られたことをいくつか示す。今後こうしたノウハウをより集め、公開していきたい。

1. 前提ノードについて

前提ノードは主に２通りの使い方がある。

(ア) 議論する上での前提となる情報を提供する。ウエブサーバの例では、「アーキテクチャ設計文書」が相当する。これまでの記述実験で、どのようなドキュメントが前提ノードとして参照されたか、いくつかあげる。

① 自動車のエンスト問題

1. 「エンジンの運用条件情報」。エンジンが動作する環境に関わる条件(パラメター)のリスト。エンストが起きないことを議論するにあたって、前提となる情報である。トップゴールにつけられる。
2. 「DRBFM 法の説明書」。DRBFM とは、Design Review Based on Failure Mode の略であり、トヨタなどで用いられているリスク分析手法である。DRBFM に基づくリスク分析を用いたエンジン開発が、適切に行われていることを議論するために参照する。

② 日本未来館の受付ロボット

1. 「２次元地図による自己位置推定機能仕様書」。受付ロボットは２次元地図を常に更新し自己位置を推定し続ける。この機能が適切に実装されているかを議論する時に参照する。
2. 「想定する静的障害物のリスト」。あるフロアにある静的障害物を地図に記録できるか議論するときに参照する。

③ ET ロボットコンテスト

1. 「ET ロボコン競技規約」。ロボットコンテストは競技規約にしたがって行われる。競技規約を前提として、ロボットのディペンダビリティを議論する。トップゴールに前提ノ

ードとして参照される。

2. 「想定するハザード」。競技コースをロボットが走行する上でのハザードを列挙し、それらに対処できることを議論するために参照する。想定するハザードとしては、「通信障害によりスタートできない」、「ライントレースに時間が掛かる」、「ラインから外れる」、「他のロボットに追突する」などがあげられる。

(イ) ドキュメントをもとにゴールを分解するとき、そのドキュメントを前提ノードで参照する。上の例では、「運用ワークフロー定義文書」が相当する。図 4.1.1-2 のように、ゴールではなく、対応する戦略ノードにつけると分かりやすい。

前提ノードには、それにリンクされるゴールおよびサブゴールにおける議論の前提となるドキュメントが置かれる。あるゴールを議論する上で前提として参照したいドキュメントは、そのゴールか、その上のゴールの前提ノードにおけばよい。しかしあまり上のほうにおくと、ゴールと前提の関連がわかりづらくなる。前提ノードはできるだけ下の方においたほうがよい。例えば、図 4.1.2-1 において、もし前提 C_1 が、ゴール G_2 を議論するためだけに用いられるのならば、左の置き方よりも、右の置き方のほうが、範囲が限定されてわかりやすい。

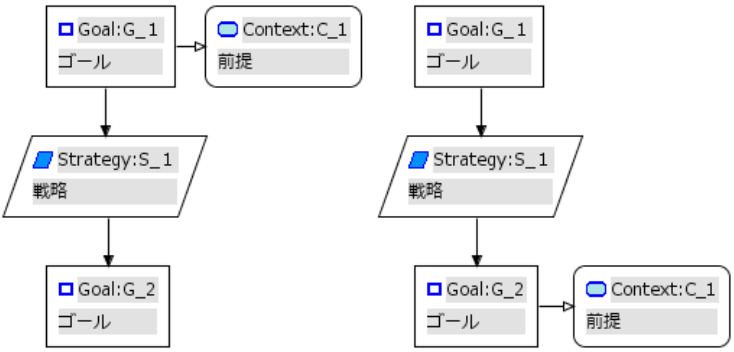


図 4.1.2-1 前提ノードの置き方

2. サブゴールへの分け方

・ ゴールを、戦略ノードを通してサブゴールに分解するとき、分解の仕方が複数考えられる場合が多い。例えば、あるシステムのディペンダビリティを議論する場合、まずそのシステムの機能毎に分けて議論し、それぞれの機能ごとにリスクに対処できることを議論するか、あるいはまずリスクごとに議論し、それぞれのリスクごとに、関係する機能がそのリスクに対処できることを議論してもよい。また、システムのライフサイクルで生成されるドキュメントの中で、どのドキュメントに着目するかで、議論の構造は大きく変わりうる。例えば、ウエブサーバの D-Case を記述するとき、「アーキテクチャ設計文書」にまず着目し、ウエブサーバ、アプリケーションサーバ、データベースサーバごとに分けて議論することもできる。あるいは「SLA 文書」に着目し、SLA として要求されている項目ごとに議論することもできる。ディペンダビリティケースの現在の研究においては、サブゴールをどのように分けるべきか、明確な基準を定義するに至っていない。さらに、D-Case と他のドキュメントがどのように違うのかなど、基本的な質問を受けることが多い。ここでは、我々がこれまで考えてきたこと(当たり前ではあるが)を示す。

・ **D-Case を見せる人の立場にたって、議論構造、内容を考える。**D-Case を用いて、利害関係者に説明するとき、その利害関係者が理解できない、興味がないことを説明しても理解は得られない。例えば、相手企業の上層部の人に対して、システムの技術の詳細を説明しても仕方がない場合がある。上層部の人に対しては、システムのディペンダビリティ要件(SLA 文書など)がサブゴールごとに議論されているトップレベルの議論構造を見せ、詳細は省くなど、工夫したほうがよい。逆に、相手企業の技術部門の人に対しては、システムの技術的詳細に関する D-Case を示し、その技術が関わるディペンダビリティ要件が適切なテストやベンチマークによって達成されていることを議論したほうがよい。そしてトップレベルの議論構造と、技術的詳細を説明する部分がつながり、一つの D-Case を構成するのが理想である。

・ **システムのディペンダビリティにとって重要なものから議論する。**システムのディペンダビリティは、システムライフサイクルで生成されるすべてのドキュメントに関係しうる。それらすべてのドキュメントを用いて D-Case を記述すると、D-Case は巨大になりすぎるかもしれない。まずシステムのディペンダビリティを考える上で、重要な観点(運用ワークフローに沿った観点など)を決

め、それに従って議論を展開し、必要なドキュメントを選びながら D-Case を作っていったほうがよい。

- **D-Case と既存ドキュメントの違い。**よく聞かれる質問である。基本的には、D-Case は既存のリスク分析解析ドキュメントや、仕様書を置き換えるものではない。D-Case は既存ドキュメントを参照し、あるいは既存ドキュメントを展開して構成される、ディペンダビリティを保証するドキュメントである(図 4.1.2-2)。
- **D-Case と他のゴール指向要求分析手法との違い。**D-Case はゴール指向要求分析手法の一種にもクラス分けされうる。他のゴール指向要求分析手法との違いは、簡単には以下のようにまとめられる。
  - 目的
    - D-Case: システムのディペンダビリティの確認、保証
    - 他の手法：要求されるゴールを達成するための手段などの分析
  - ゴールの分解の仕方
    - D-Case: 複数あり、分解の観点が重要
    - 他：分解の観点はほぼ 1 通り

他のゴール指向要求分析手法と、木構造であることは同じであり、厳密に他の手法と分けることは難しいが、大切なことは D-Case を含め様々な手法を、目的に合わせてうまく使い分け、共用していくことである。

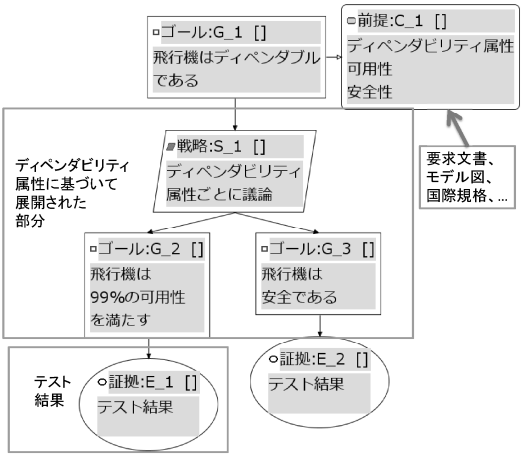


図 4.1.2-2 D-Case と他の種類のドキュメントとの関係の例

サブゴールの分け方、議論構造の決め方などは、一意に決められるものではない。我々は、これまでの記述実験を通して、システム領域ごとに適切なサブゴールの分け方、議論構造を考えてきており、だいぶまとまってきた。今後それらをまとめ、公開していく予定である。他の手法との利用用途の違いを明確にし、他の手法とうまく組み合わせた手法などに、D-Case 手法を発展させていきたい。

参考文献

[1] Yutaka Matsuno, Shuichiro Yamamoto, A New Method for Writing Assurance Cases, to appear in International Journal of Secure Software Engineering, 2013.

[2] Alexander, R., Hawkins R., & Kelly, T. (2011). Security Assurance Cases: Motivation and the State of the Art. Technical Note CESG/TR/2011/1, High Integrity Systems Engineering, Department of Computer Science, University of York.

[3] GSN Community. GSN Community Standard Version 1.0, 2011

[4] DEOS プロジェク whitepaper v3.0. http://www.dependable-os.net/ja/topics/file/White_Paper_V3.0J.pdf

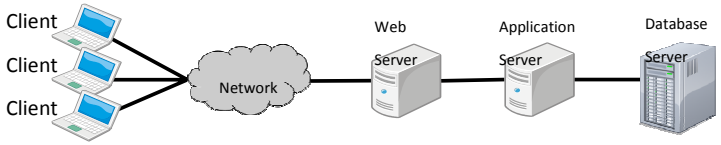
4.2　D-Case 作成法　演習

【問題１】ウエブサーバシステム

以下のドキュメントを使って、D-Case のトップゴールとそのサブゴール群を書け。

トップゴールは、SLA ドキュメントを参考にせよ。

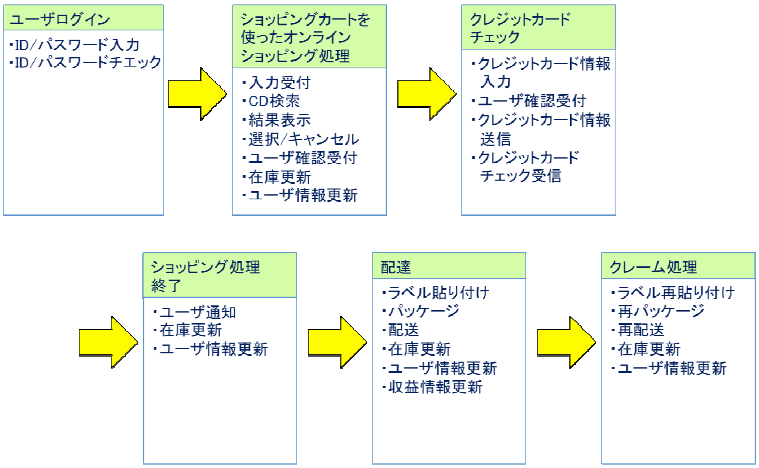
・　システム構成図



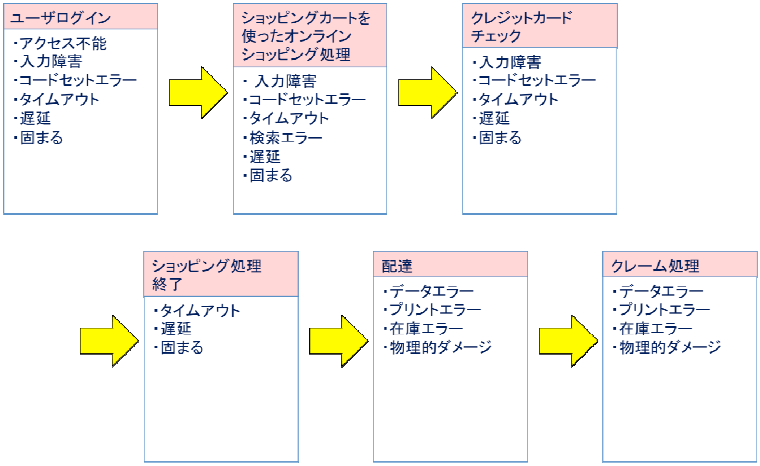
・　SLA(Service Level Agreement)ドキュメント

- ダウンタイム
  - 最大　10 分/年
- 性能
  - アクセス許容量：2500 access/min
  - レスポンスタイム：800 msec/access
- データベース容量
  - 最大登録ユーザ数: 1,000,000

・　運用ワークフロー定義書



・　運用ワークフローの各ステップのリスク分析結果



【問題２】ロボットレース

下図のようなコースで行うロボットレースがあるとする。ロボットは Lego Mindstorm で作られたものである。コースに書かれたラインを読みながら(ラントレース)、走行するとする。与えられたドキュメントをもとに適切なトップゴールを設定して、D-Case を書け(証拠ノードまで書くこと。「他のサブゴールと同様」などの省略は用いてよい)。

ロボットのイメージ

ドキュメント

- ロボットレース競技規約
  - ベーシック区間は４０秒以内にクリアすること
  - 障害区間では、与えられた条件をクリアすること
- レースコース仕様書
  - ベーシック区間、障害区間と与えられた条件
- ロボット仕様書

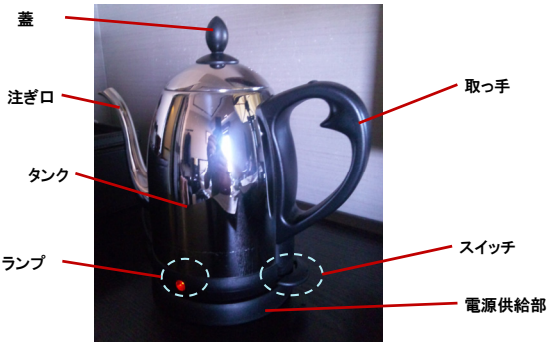
- ロボットの設計書およびレビュー結果
    - 通信、ライントレース、階段を登る機能の設計書およびレビュー結果
- ロボットコンポーネントのユニットテスト結果
    - 通信、ライントレース、階段を登る機能のユニットテスト結果
- ベーシック区間、障害区間走行テスト結果
    - 通信、ライントレース、階段を登る機能の走行テストログ
- リスク分析結果
    - ベーシック区間でのリスク分析結果
        - 通信障害によりスタートできない
        - ライントレースに時間が掛かる
    - 障害区間でのリスク分析結果
        - 階段を登る衝撃でロボットが転倒する

【問題３】地下鉄の車内安全性

地下鉄車内で発生する危険に対して、十分な対策ができていることを D-Case を用いて説明しなさい

【問題４】電気ケトル

下図に示すような電気ケトルのディペンダビリティに対する D-Case を作成しなさい



【問題５】組込みシステムの論理参照モデル

下図に示すような組込みシステムの論理参照モデルの制御システムに対するディペンダビリティケースを作成しなさい。

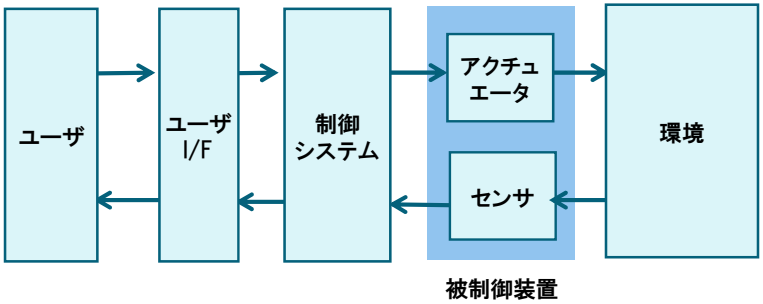


図　組込みシステムの論理参照モデル

【問題６】預貯金システム

下図に示した預貯金システムの D-Case を書きなさい

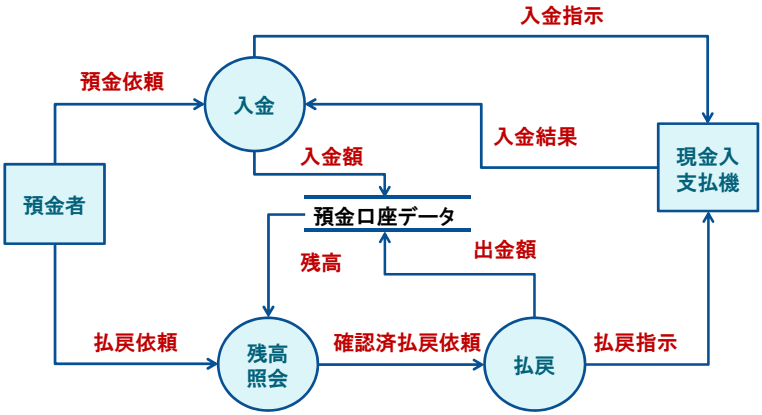


図　預貯金システムのデータフロー図

【問題７】デジタル地図作成システム

下図に示すデジタル地図作成プロセスに対する D-Case を書きなさい

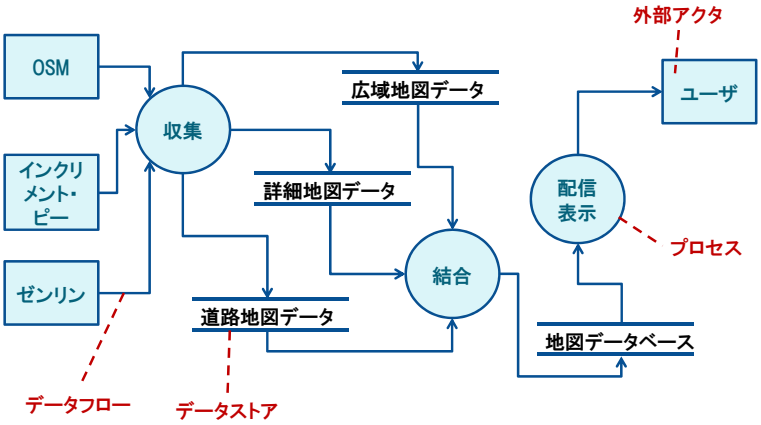


図　デジタル地図作成システムのデータフロー図

【問題８】模擬衛星(応用問題)

この問題は、超小型衛星の開発に参加されている慶応大学の田中康平氏より頂いたものである。この問題では、実際のシステムに近い例を考える。

模擬人工衛星 CanSat の電源部の D-Case を、添付の資料の機能図と仕様書をもとに書け。

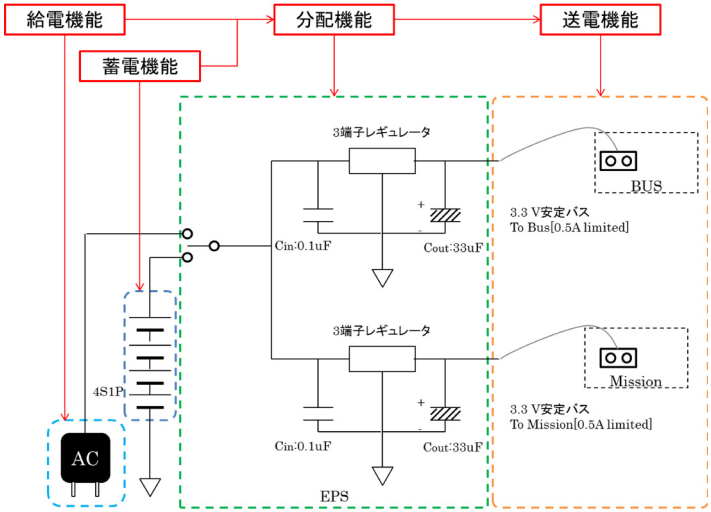
トップゴールは以下に設定すること。

・CanSat の電源部は他の系からの要求を満たす。

補足：CanSat は人工衛星と同様の機能を持つ、地上で実験用に用いられる空き缶サイズの模擬衛星であることから、CanSat と言われている。

CanSat は通常電源を OFF 状態で打ち上げるものだが、今回対象としている CanSat はロケットの振動環境を計測するためのものであり、常時搭載機器が動作状態であることが求められている。

発展課題：添付の仕様書通り製造したところ、実験中に不具合が生じた。書いた D-Case を使って、指摘せよ。



電源部 要求・設計仕様書

1 記述内容

本文章では、CanSat の電源部製作にあたっての要求及びその要求に対する設計結果を記述する。

2 他系から電源部への要求

1. 取り扱いが容易な電池を選定すること
   (ア) 購入が容易であること
   (イ) 買い出しの観点から、何度も購入しなくて済む電池を選定のこと
   (ウ) 過放電保護機能が不要な電池であること
2. 電源部は運用期間中に搭載機器へ電力を供給し続けること
   (ア) 運用期間中とはロケットに搭載してから打上げ、CanSat を放出してから回収するまでの期間(90 分)を示す。常時加速度を計測する。
   (イ) 機器へと供給する電力は表 1 の要求を満たすこと
3. 電源部は 135x43 mm の面積上に搭載できること

4. 電源部は 100 g 以下の質量であること
5. スイッチを ON/OFF できること
6. 1 セットのコストが 5000 円以下であること
7. 外部給電によって各機器に電力を供給できること
8. 振動によって電源部の機能が損失しないこと

表 1　搭載機器と消費電力について

系	機器名	型番	消費電圧[V]	消費電流[A]	消費電力 [Wh]
電気系	OBC(Bus)	SH7145	3.3	160mA	0.792
	GPS	GT723-F	3.3-6.0	50mA@初期補足 30mA@通常測位	0.198
	Tx	TS02EJ-S mdm3LDM2	2.1-7.0	26mA@送信時 18mA@受信時	0.066
	FP	N/A		1mA 以下	0
	光センサ	S9648-100	12V (max)	5mA (max)	0.09
	SDカード(Bus)	Micro SD Card	3.3	数十mA (未確認)	0.099
Mission系	OBC(MIssion系)	PIC18LF4550(クリスタル16MHz)	2.0-5.0	200mA	0.99
	加速度センサ	ADIS16223	3.3	52mA	0.117
	SDカード(Mission)		2.7-3.6	数十mA(未確認)	0.099
電源系	BAT	HR-4UTGB	1.2 * 3	750 mAh	
地上系	Rx	TS02EJ-S mdm3LDM2	2.1-7.0	26mA@送信時 18mA@受信時	

3 電源部の仕様

1. 取り扱いが容易な電池を選定すること

(ア) 購入のしやすさという観点から、家電量販店で購入可能な電池を選定する。

(イ) 二次電池から選定する。この際、充電器も合わせて入手可能な電池から選定する。

(ウ) 取り扱いがしやすく、過放電保護機能を要しない、という観点から、Ni-MH 系バッテリから搭載バッテリを選定する。

上記より、eneloop を搭載バッテリとして選定した。

CanSat クラスでの出場のため、小さい体積でかつ電圧が高いことが望ましい。ここで、単 3 形電池と単 4 形電池などの搭載が考えられる。電圧を高く、つまり直列数を同じにする場合には、電池そのモノのサイズが小さい方が、バッテリモジュールの体積としては小さくて済む。よって、電池サイズの小さい単 4 形電池で検討する。

2. 電源部は運用期間中に搭載機器へ電力を供給し続けること

表 1 に搭載する機器の消費電力量を求めると、2。451Wh となる。これは、消費電圧 x 消費電流 x 稼働時間を計算することで算出した。この消費電力を満たすためには単 4 形電池の場合は、eneloop が 3 本以上あれば良い[表 2]。

表 2　CanSat 消費電力及び供給電力

	個数	電力量 [Wh]	体積 [mm]
消費電力	ー	2.451	
供給電力	3	2.7	11*33*45
	4	3.6	11*44*45

表 1 に示す搭載機器の要求から、電源部は 3.3 V の電源を給電する必要がある。

直列数に対する供給電圧の範囲を表 3 に示す。3.3V 給電を行う場合は、電圧が 3.3V 以上ある必要があるため、直列数は 4 直列が望ましい。

1 直列 3 並列として昇圧することも考えられる。昇圧することによって発生するノイズが、各搭載機器に与える影響を評価することは今回困難である。これは、遠隔開発を行なっているために、時間的に何度も噛み合わせ試験を行う余裕がないためである。

よって、今回は 4 本搭載することとした。

表 3　供給電圧の範囲

	最高電圧	最低電圧
3直列	3.6 V	3.0 V
4直列	4.8 V	4.0 V

3. 電源部は 135x43 mm の面積上に搭載できること

搭載予定の電池の仕様を表 4 に示す。電池を 4 本搭載することは可能である。

表 4　搭載電池仕様

型番	HR-4UTGB
電力容量	Min. 750mAh Typ. 800mAh
体積	(D) 10.5 x (L) 44.5 mm
質量	13 g

4. 電源部は 100 g 以下の質量であること

電池の質量は

・電池ケース　　：16 g (4 g/個)

・電池　　：52 g (13 g/個)

から、68 g となる。

回路の質量は 30 g であるため、電源部の質量は 98 g となる。

これは、構造系からの要求である 100 g 以下であることを満たす。

5. スイッチを ON/OFF できること

スイッチを取り付ける。

6. 1 セットでのコストが 5000 円以下であること

コストについては 5 章を参照のこと。

7. 外部給電によって各機器に電力を供給できること

AC アダプタを用いて、外部から各機器へ給電する。

8. 振動によって電源部の機能が損失しないこと

(ア) ロケットの打上げ振動によって電池・電源基板が外れないこと

(イ) ロケットの打上げ振動によって各基板への給電ケーブルが断絶しないこと

4 電源部仕様まとめ

4.1 電源部のシステム構成

図 1 に電源部のシステム構成を示す。4 直列 1 並列の電池から、非安定バスと 3.3V 安定バスにて各機器へ電力を供給する。外部から給電するための“給電機能”、搭載品を動作させるための“蓄電機能”、各システムに制御した電力を供給するための“分配機能”、各システムに電力を受け渡す“送電機能”から機能は構成される。

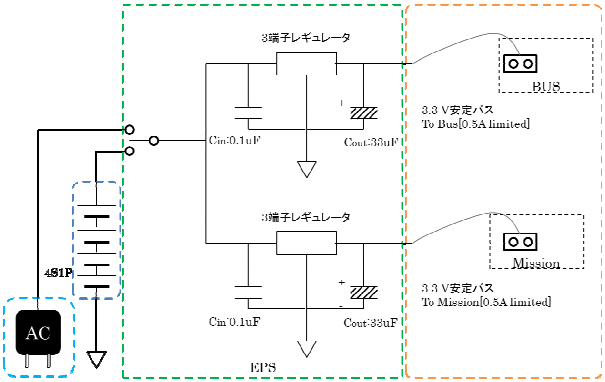


図 1　電源部のシステム構成図(回路図)

4.2 他系との I/F

電源の I/F は表 1 に従う。電源の安定度要求は±0.3 V。

構造の I/F は M3 のボルトで固定する。電源部の寸法を図 2 に示す。

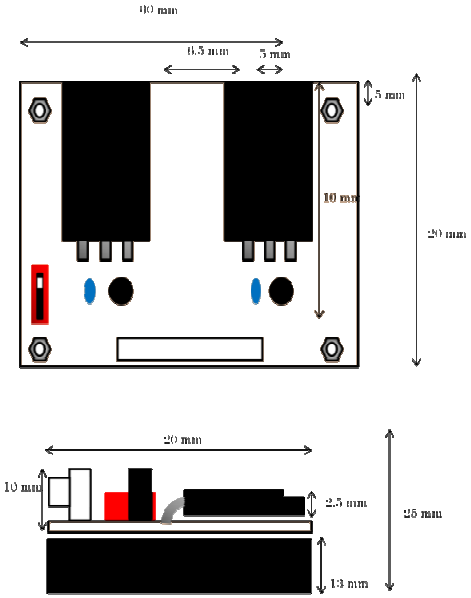


図 2　電源部寸法

4.3 電源部の機能

1. 電力供給機能
   (ア) 3.3V 安定電源及び非安定電源の供給ができる
   (イ) スイッチによって電力供給を遮断できる
   (ウ) 配線によって他の基板へ電力を受け渡しできる
2. 部品保持
   (ア) 電池を所定の構造体に固定できる
   (イ) 電子基板を所定の構造体に供給できる
   (ウ) 振動によって取りはずれない
3. 外部給電機能
   (ア) AC アダプタから 5V/1A を給電し、各機器に 3.3V 安定電源を供給する。

5 実施試験項目

1. 機能確認 [参照：機能確認試験結果報告書]
   1.1.要求通りの電圧を給電することを確認した
      試験環境：常温下、単体試験
      試験条件：抵抗を接続し、規定の電圧を供給できることを確認した
   1.2.スイッチが正常動作することを確認した
      試験環境：常温下、単体試験
      試験条件：給電の ON/OFF が可能なことを確認した
   1.3.供給段での電圧レベルが要求通りであることを確認した
      試験環境：常温下、単体試験
      試験条件：抵抗を接続し、規定の電圧を供給できることを確認した
   1.4.外部電源から給電できることを確認した
      試験環境：常温下、単体試験
      試験条件：外部電源を接続し、供給段での電圧を確認した
2. 充放電試験結果
   2.1.システムを噛みあわせて 2 時間動作することを確認した
      試験環境：常温下、全システム噛み合せ
      試験条件：全機器動作状態で 2 時間動作することを確認した
3. 振動試験結果
   3.1.振動によって部品が外れないことを確認した

試験環境：常温下、電源は OFF 状態
試験条件：給電はしない状態で振動試験を実施した

6 コスト

購入予定物品：

eneloop(充電器込み)	4 本	2,000 円
電池ケース	4 つ	50 円
分配機能		1,000 円
スライドスイッチ	1 つ	
コネクタ(8 ピン)	1 つ	
三端子レギュレータ	2 つ	
40x40 mm 基板		1 枚
スペーサ	4 つ	
ハーネス		50 円
AC アダプタ		500 円

【問題 9】ISO 26262 (応用問題)

この問題は、ISO 15026 などの標準化活動を行なっている奈良先端科学技術大学院大学の高井利憲氏より頂いたものである。ディペンダビリティケースの用例としては、ある国際規格にシステムが適合していることを、認証者に示すことが主にあげられる。この問題では自動車の機能安全規格である ISO 26262 を例にとり、D-Case を書いてみる。ただしこれはあくまで例であって、ISO26262 で要求されているセーフティケースとは直接には関係しない。

以下の資料を参考にして、ISO26262 の安全分析手順にしたがってエアバッグのリスク分析などを行なっていることを示す D-Case(セーフティケース)のトップレベルを書け。

まず関連用語を説明する。

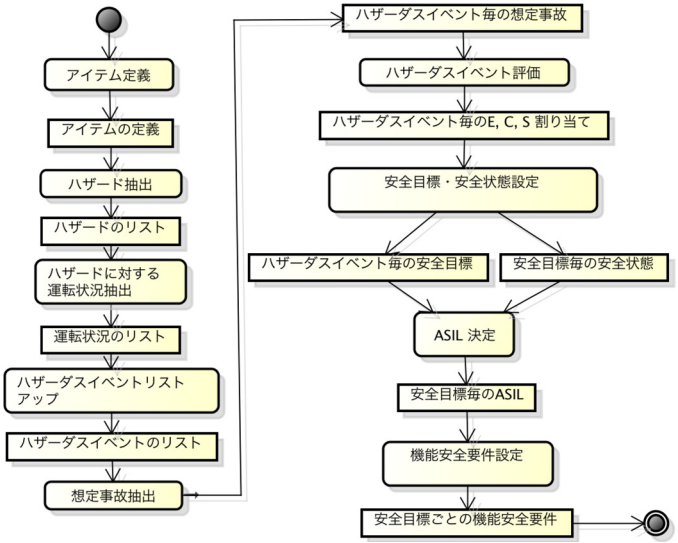
- アイテム: 対象とするシステム
- ハザード: アイテムの意図しない振る舞いによって引き起こされる事故の原因
- ハザーダスイベント： ハザードと自動車の運転状況のペア

・　ASIL：自動車向け安全度水準。A~D の 4 段階で安全性をレベル分けする

ISO26262 の安全分析手順はおおまかには以下のとおりである。

1，アイテム定義
2，ハザード抽出
3，ハザードに対する運転状況抽出
4，ハザーダスイベントのリストアップ
5，想定事故抽出
6，ハザーダスイベント評価
7，安全目標・安全状態設定
8，ASIL 設定
9，機能安全要件設定

この手順をフローで表すと下図のようになる。



ISO 26262 安全分析手順フロー

ASIL を決定するためには、下記の遭遇頻度(E)、回避難易度(C)、過酷度(S)を用いる。

- E : 遭遇頻度

	Class				
	E0	E1	E2	E3	E4
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

- C : 回避難易度

	Class			
	C0	C1	C2	C3
Description	Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

- S : 過酷度

	Class			
	S0	S1	S2	S3
Description	No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

- ASIL 決定表

Severity class	Probability class	Controllability class		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

ASIL 決定表

上記をもとに、エアバッグの安全性分析を以下のように行う。

1. アイテム定義
   - エアバッグシステムの定義記述
2. ハザード抽出
   - 例: 想定しないエアバッグ(助手席用)の起動
3. ハザードに対する運転状況抽出
   - 例: 停止中、走行中、助手席に人あり、なし
4. ハザーダスイベントリストアップ
   - 例：走行中、助手席に人がありで助手席のエアバッグが起動
5. 想定事故抽出
   - 例：エアバッグが助手席の子供に衝突
6. ハザーダスイベント評価
   - E、C、S を評価
7. 安全目標・安全状態設定
   - 上記運転状況ごとに安全目標を設定(ゴールとして採用)
   - 安全目標を満たす安全状態を設定
8. ASIL　設定
   - 上記 E、 C、 S から ASIL を決定

9. 機能安全要件設定
   - ハザードの原因などを考慮し、
     安全目標を実現する安全要件を設定

本問では、上記手順が以下のように行われたとする。

2. ハザードの列挙
   1) 想定しないエアバッグの起動
   2) 事故想定衝撃検知時にエアバッグが起動しない
3. 運転状況の列挙
   a) 通常停止中
   b) 通常走行中
   c) 事故発生後
4. ハザーダスイベントのリストアップ
   a) 通常停止中、想定しないエアバッグの起動
   b) 通常走行中、想定しないエアバッグの起動
   c) 事故発生後、事故想定衝撃検知時にエアバッグが起動しない
5. 想定事故抽出
   4 で得られたハザーダスイベントにたいして、想定事故を以下のように抽出したとする。

ハザーダスイベント	想定事故
a)	予期しないエアバッグ起動によるエアバッグとドライバとの接触
b)	予期しないエアバッグ起動によるドライバ視界不良などによる交通事故の誘発
c)	交通事故時にエアバッグが起動しないことによるハンドルとドライバとの接触

6. ハザーダスイベント評価
   以下のように評価したとする。

ハザーダスイベント	想定事故	E	C	S
a)	予期しないエアバッグ起動によるエアバッグとドライバとの接触	E3	C2	S1
b)	予期しないエアバッグ起動によるドライバ視界不良などによる交通事故の誘発	E1	C2	S3
c)	交通事故時にエアバッグが起動しないことによるハンドルとドライバとの接触	E1	C3	S3

7. 安全目標・安全状態設定

以下のように設定したとする。

ハザーダスイベント	安全目標	安全状態
a)通常停止中想定しないエアバッグの起動	想定外の起動は起きない	起動システムに異常を検知したら、ドライバに警告を発する
b)通常走行中想定しないエアバッグの起動	想定外の起動は起きない	起動システムに異常を検知したら、ドライバに警告を発するとともに、非常用アナログシステムに切り替える
c)事故発生後事故想定衝撃検知時にエアバッグが起動しない	事故想定衝撃検知時には必ず起動する	起動システムに異常を検知したら、非常用アナログシステムに切り替える

8. ASIL 設定

ハザーダスイベント	E	C	S	ASIL
a)	E3	C2	S1	なし
b)	E1	C2	S3	A
c)	E1	C3	S3	C

9. 機能安全要件設定

機能安全要件を以下のように設定したとする。

ハザーダスイベント	安全目標	安全状態	機能安全要件
a)通常停止中想定しないエアバッグの起動	想定外の起動は起きない	起動システムに異常を検知したら、ドライバに警告を発する	起動システムの異常を検知する機能
			ドライバに警告する機能
b)通常走行中想定しないエアバッグの起動	想定外の起動は起きない	起動システムに異常を検知したら、ドライバに警告を発するとともに、非常用アナログシステムに切り替える	起動システムの異常を検知する機能
			ドライバに警告する機能
			非常用アナログシステム
			非常用アナログシステムへ切り替える機能
c)事故発生後事故想定衝撃検知時にエアバッグが起動しない	事故想定衝撃検知時には必ず起動する	起動システムに異常を検知したら、非常用アナログシステムに切り替える	起動システムの異常を検知する機能
			非常用アナログシステム
			非常用アナログシステムへ切り替える機能

5　議論分解パターン

議論分解パターンでは、表 5-1 に示したように、①解決したいディペンダビリティケース作成上の問題、②議論分解パターンが適用される典型的な議論状況、③議論上の問題に対する議論分解パターンによる解決方法、④議論分解パターンの利点と欠点からなる特徴、⑤議論分解パターンを活用してディペンダビリティケースを作成するときの留意点や適用事例を記述する。

議論分解パターンを用いて、ディペンダビリティケースを作成する上での分解問題の種類と、それに適用できるディペンダビリティケースの種類を対応付けることができるので、ディペンダビリティケースの分解方針と解決策の再利用が期待できる。

表 5-1　分解パターンの構成要素

構成要素	説明
分解問題	解決が必要とされる分解上の課題
分解状況	問題と解決策が配置される世界の前提となる状況
解決策	問題に対して具体化された解決策
特徴	解決策が持つ利点と欠点
留意点	分解パターンを適用する際の注意事項

5.1　基本パターン

表 5.1-1 では、ブルームフィールドが示した保証ケースに関する 7 個の分解パターン[2]をまとめている。

表 5.1-1　議論分解のパターン

項番	パターン	説明
1	アーキテクチャ分解	システム構成に従って分解
2	機能分解	主張を機能構成に従って分解
3	属性分解	特性を複数の属性に分解
4	帰納分解	説明対象の場合分けによる分割
5	完全分解	説明対象のすべての要素による分割
6	単調分解	新システムによる旧システムの改善点による分解
7	修正分解	曖昧性の明確化による分解

5.1.1 アーキテクチャ分解パターン

【分解問題】複雑なシステムに対してディペンダビリティケースを作成する必要がある。

【分解状況】アーキテクチャ分解を適用する場合の前提となる状況は、アーキテクチャが明確化されていることである。

【解決策】コンポーネントをシステム構成に従って複数の下位コンポーネントに分解

対象システムが２つのサブシステム A と B で構成されるとき、図 5.1.1-1 に示すように、分解することができる。図 5.1.1-1 では、分解の根拠となるアーキテクチャを示すために、前提として「システム構成の定義」を記述している。またサブシステム間に相互作用がある場合、この相互作用についての主張も記述する。

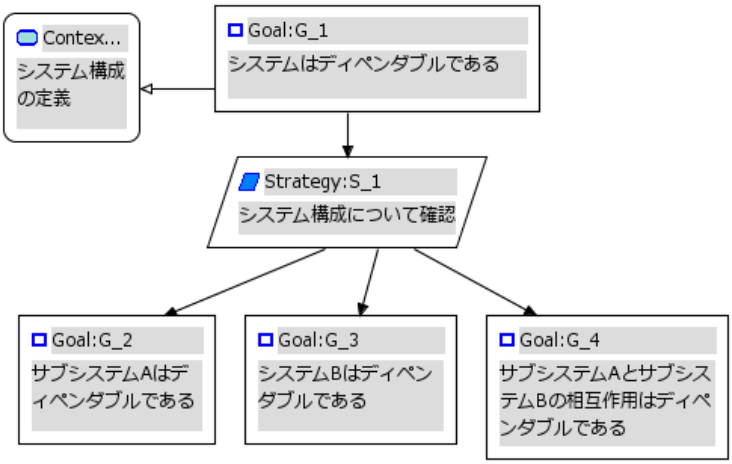


図 5.1.1-1　アーキテクチャ分解の例

【特性】

(利点)システムのアーキテクチャに従って、ディペンダビリティケースを自然に作成できる。このため、システムのアーキテクチャとディペンダビリティケースとの対応関係の一貫性を確認しやすい。

(欠点)システムのアーキテクチャが定義されていなければ、アーキテクチャ分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 3 点がある。

(1)コンポーネント間の相互作用を明らかにする必要がある。

(2)アーキテクチャが満たすべき性質を主張として明確に定義する必要がある。

(3)分解根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.1.2 機能分解パターン

【分解問題】システムの機能に対してディペンダビリティケースを作成する必要がある。

【分解状況】機能分解を適用する場合の前提となる状況は、システムの機能が明確になっていることである。

【解決策】主張を機能構成に従って分解

対象システムが複数の機能で構成されるとき、機能ごとに下位の主張を作成することにより上位の主張を分解できる。

たとえば、図 5.1.2-1 に示すように、検索システムが提供する機能に従って分解することができる。図 5.1.2-1 では、分解の根拠となる機能構成を示すために、前提として「検索システムの機能定義」を記述している。この検索システムの機能には、キーワード入力機能、データ管理機能、キーワード検索機能、検索結果出力機能があることが分かる。

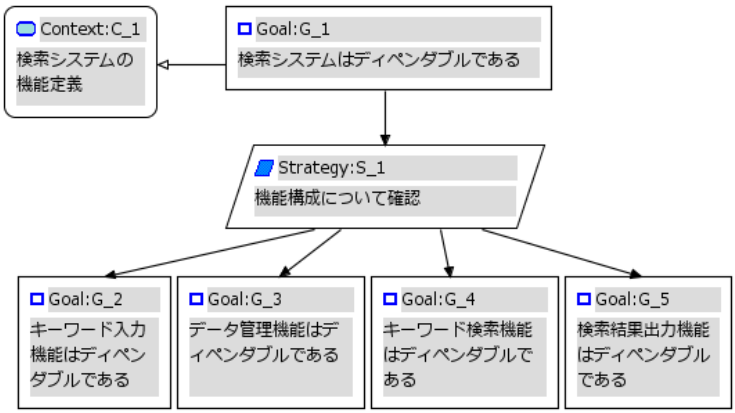


図 5.1.2-1　機能分解パターンの例

【特性】
(利点)システムが提供する機能に従って、ディペンダビリティケースを自然に作成できる。このため、システムの機能構成とディペンダビリティケースとの対応関係の一貫性を確認しやすい。
(欠点)システムが提供する機能構成が定義されていなければ、機能分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 2 点がある。
(1)機能が漏れないように、機能構成の網羅性を確認する必要がある。
(2)機能分解根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.1.3 属性分解パターン

【分解問題】システムの属性に対してディペンダビリティケースを作成する必要がある。
【分解状況】属性分解を適用する場合の前提となる状況は、確認すべきシステム属性が明確になっていることである。
【解決策】主張を機能構成に従って分解
対象システムの特性が複数の属性で構成されるとき、属性ごとに下位の主張を作成することにより上位の主張「システムは特性(を持つ)状態である」を分解できる。
たとえば、図 5.1.3-1 に示すように、ディペンダブル特性を構成する属性に従って分解することができる。図 5.1.3-1 では、分解の根拠となる属性構成を示すために、前提として「ディペンダビリティ属性定義」を戦略ノードに接続している。ここで、ディペンダビリティ属性には、可用性、信頼性、安全性、一貫性、保守性があるとしている。

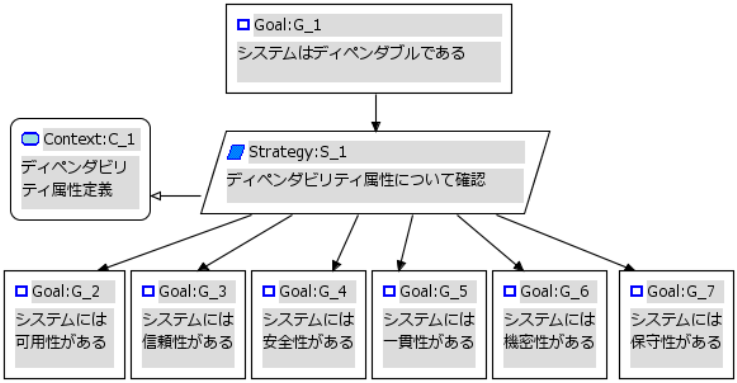


図 5.1.3-1　属性分解パターンの例

【特性】

(利点)特性を構成する属性に従って、ディペンダビリティケースを自然に作成できる。このため、システムが持つべき属性構成とディペンダビリティケースとの対応関係の一貫性を確認しやすい。

(欠点)システムが持つべき属性構成が定義されていなければ、属性分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 2 点がある。

(1)重要な属性が漏れないように、属性構成の網羅性を確認する必要がある。

(2)属性分解根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.1.4　帰納分解パターン

【分解問題】システムの属性に対してディペンダビリティケースを作成する必要がある。

【分解状況】帰納分解を適用する場合の前提となる状況は、確認すべきシステムが扱う場合が明確になっていることである。

【解決策】システムが扱う必要のある場合ごとに主張を分解

対象システムが扱う必要のある複数の場合を明確に定義できるとき、場合ごとに下位の主張を作成することにより上位の主張を分解できる。

たとえば、図 5.1.4-1 で示すように、ディペンダブル特性を帰納法に従って分解

することができる。帰納法では、第１段階と帰納段階の２つの場合に分けている。すなわち、第１段階で主張が成立すること(N=1 の場合)と、第 N=K で主張が成立するとき N=K+1 でも主張が成立すること(帰納段階 N=K の場合)を確認する。

図 5.1.4-1 では、分解の根拠となる場合分けを示すために、前提として「数学的帰納法」を戦略ノードに接続している。同様にして、時間帯などによる場合分けで帰納分解するときには、根拠となる時間帯の場合分けの定義を前提ノードで明記する必要がある。

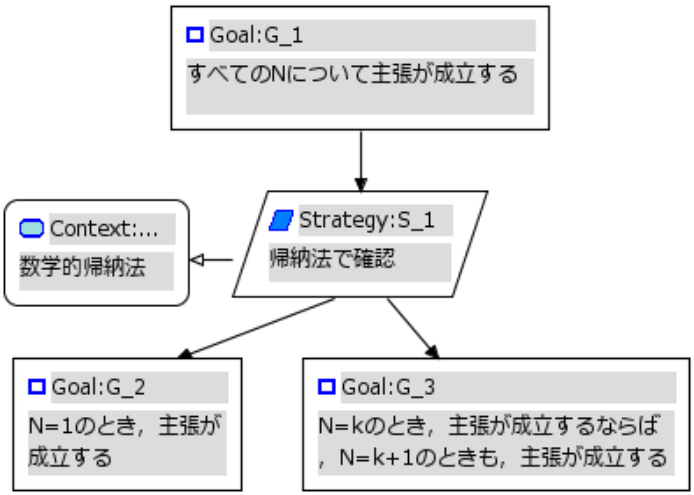


図 5.1.4-1　帰納分解パターンの例

【特性】

(利点)場合分けに従って、ディペンダビリティケースを自然に作成できる。このため、場合分けの構成とディペンダビリティケースとの対応関係の一貫性を確認しやすい。

(欠点)場合分けの構成が定義されていなければ、帰納分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 2 点がある。

(1)重要な場合が漏れないように、場合分け構成の網羅性を確認する必要がある。

(2)帰納分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.1.5 完全分解パターン

【分解問題】システムが対象とする集合に属するすべての要素に対してディペンダビリティケースを作成する必要がある。

【分解状況】完全分解を適用する場合の前提となる状況は、確認すべきシステムが扱う集合の要素が明確になっていることである。

【解決策】説明対象のすべての要素による分割

対象システムが扱う必要のあるすべての要素を明確に定義できるとき、要素ごとに下位の主張を作成することにより上位の主張を分解できる。

たとえば、図 5.1.5-1 で示すように、システムのリスク要素ごとに分解することができる。システムには、入力、処理、出力があることから、システムリスクは、入力リスク、処理リスク、出力リスクに完全に分解できる。

図 5.1.5-1 では、分解の根拠となる場合分けを示すために、前提として「システムリスクには、入力リスク、処理リスク、出力力リスクがある」を戦略ノードに接続している。同様にして、時間帯などによる場合分けで帰納分解するときには、根拠となる時間帯の場合分けの定義を前提ノードで明記する必要がある。

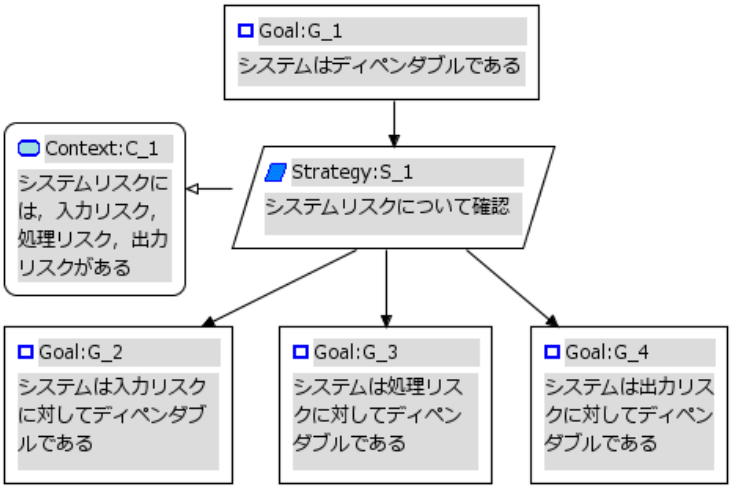


図 5.1.5-1　完全分解パターンの例

【特性】
(利点)完全分解に従って、ディペンダビリティケースを自然に作成できる。このため、説明対象を構成する要素とディペンダビリティケースとの対応関係の一貫性を確認しやすい。
(欠点)要素の完全性が定義されていなければ、完全分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 2 点がある。
(1)必要な要素が漏れないように、要素構成の網羅性を確認する必要がある。
(2)完全分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.1.6 単調分解パターン

【分解問題】対象システムの問題を改善することに対してディペンダビリティケースを作成する必要がある。
【分解状況】単調分解を適用する場合の前提となる状況は、対象システムの課題とその改善策が明確になっていることである。
【解決策】旧システムの問題点を新システムによって解決する改善策に従って分解
対象システムが持つ課題とその改善策に従って、下位の主張を作成することにより上位の主張を分解できる。

たとえば、図 5.1.6-1 で示すように、現行システムの問題、その解決策、解決策の実現性に分解することができる。図 5.1.6-1 では、分解の根拠となる場合分けを示すために、前提として「現行システム」を親ノードに接続している。

なお、この分解パターンの名称について、筆者がブルームフィールドに直接聞いたところ、問題解決では、修正対象となる現行システムから、修正結果となる将来システムへの一方向性があるので単調分解というとのことだった。

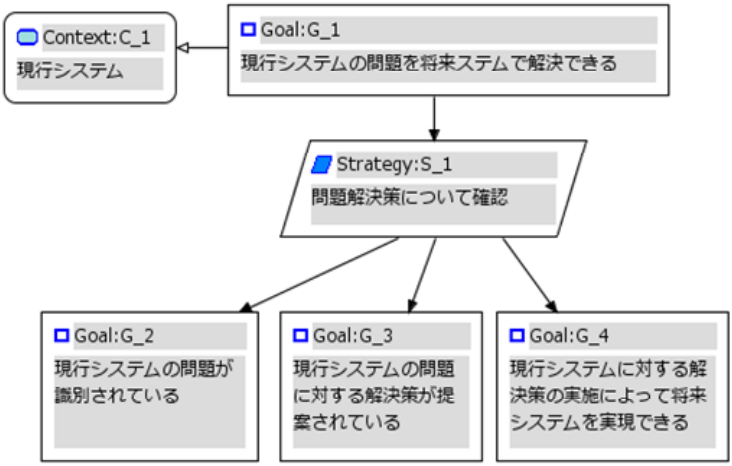


図 5.1.6-1　単調分解パターンの例

【特性】

(利点)単調分解パターンに従って、現行システムの問題状況を解決する活動に従ってディペンダビリティケースを自然に作成できる。このため、問題とその解決策との対応関係をディペンダビリティケースで確認しやすい。

(欠点)要素の完全性が定義されていなければ、完全分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 3 点がある。

(1)問題とその解決策が明確化できていることを確認する必要がある。

(2)現行システムの問題を主張ではなく親主張の前提として単調分解パターンを作成することもできる。

(3)単調分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.1.7　修正分解パターン

【分解問題】対象の曖昧さを修正することに対してディペンダビリティケースを作成する必要がある。

【分解状況】修正分解を適用する場合の前提となる状況は、修正対象の曖昧性とその修正法が明確になっていることである。

【解決策】曖昧性の明確化による分解

対象が持つ曖昧さとその具体化による修正策に従って、下位の主張を作成することにより親主張を分解できる。

たとえば、図 5.1.7-1 で示すように、対象の曖昧性、それを解消するための具体化、具体化による曖昧性の解消の確認に分解することができる。図 5.1.7-1 では、分解の根拠となる対象を示すために、前提として「対象の定義」を親ノードに接続している。

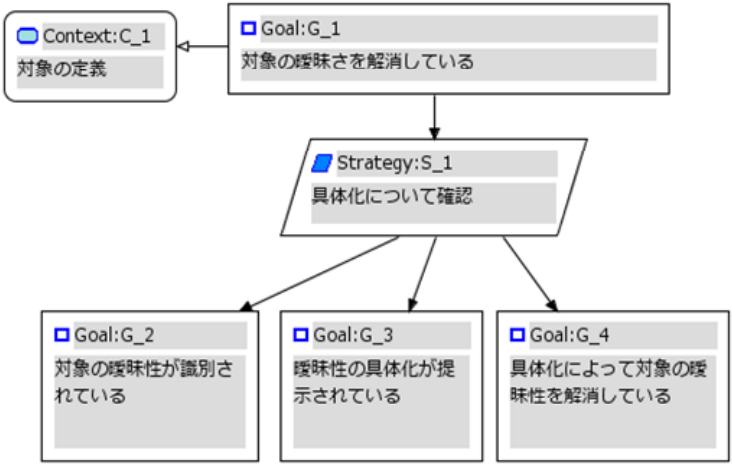


図 5.1-7　修正分解パターンの例

【特性】

(利点)修正分解に従って、曖昧さを解消することを確認するためのディペンダビリティケースを自然に作成できる。

(欠点)対象の曖昧性が識別でき、その解消策が定義できなければ、修正分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 2 点がある。

(1)対象の曖昧性を前提として修正分解パターンを作成することもできる。この場合、対象の曖昧性の識別が下位の主張ではなく親主張に接続する前提ノードとなる。

(2)修正分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.2 応用パターン

名古屋大学におけるこれまでのディペンダビリティケースに関する研究に基づいて整理した 10 個の分解パターン[3]を表 5.2-1 でまとめている。

表 5.2-1　議論分解の応用パターン

項番	分解パターン	説明
1	プロセス分解	プロセスの入力、処理、出力に対して主張を分解
2	階層分解	対象の階層構成に基づいて、主張を分解
3	DFD 階層分解	DFD の階層構成に基づいて、主張を分解
4	プロセス関係分解	先行プロセスと後続プロセスの関係に基づいて主張を確認
5	ECA 分解	イベント、条件、活動に対して主張を分解
6	条件判断分解	条件判断に対して、主張を分解
7	代替案選択分解	代替案選択に対して、主張を分解
8	矛盾解決分解	矛盾とその解決策に対して、主張を分解
9	ビュウ分解	ビュウ構成に基づいて、主張を分解
１０	ユースケース分解	ユースケースに基づいて、主張を分解

5.2.1 プロセス分解パターン

【分解問題】プロセス構造に基づいてディペンダビリティケースを作成する必要がある。

【分解状況】プロセス分解を適用する場合の前提となる状況は、プロセスの定義として、入力、処理、出力が明確になっていることである。

【解決策】プロセスの入力、処理、出力に基づいて、主張を分解する

プロセスの入力、処理、出力に従って、下位の主張を作成することにより親主張を分解できる。

たとえば、図 5.2.1-1 で示すように、システムのディペンダビリティを、入力のディペンダビリティ、処理のディペンダビリティ、出力のディペンダビリティに分解することができる。図 5.2.1-1 では、分解の根拠となる対象を示すために、前提として「プロセス定義」を親ノードに接続している。

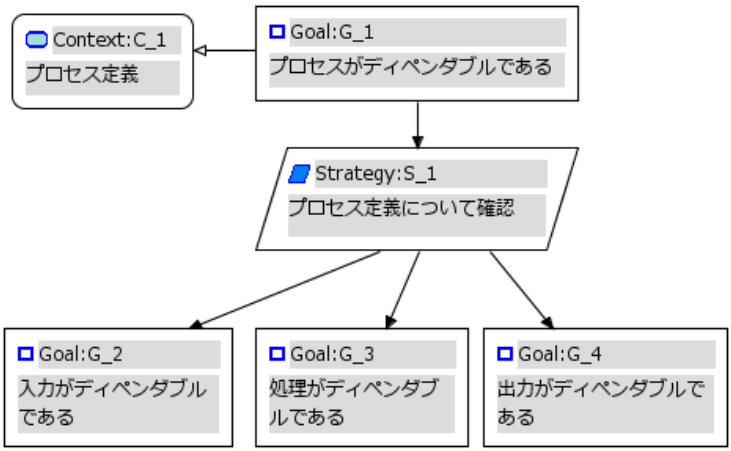


図 5.2.1-1　プロセス分解パターン

【特性】
(利点)プロセス分解に従って、プロセスに対するディペンダビリティケースを自然に作成できる。
(欠点)プロセスの入力、処理、出力が定義できなければ、プロセス分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の点がある。
　プロセス分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.2.2　階層分解パターン

【分解問題】ディペンダビリティケースの階層の深さを統一的に管理する必要がある。
【分解状況】階層分解を適用する場合の前提となる状況は、階層の複雑さの制御が必要となる大規模なディペンダビリティケースであること、階層ごとに記述すべき主張が明確にできることである。
【解決策】システムを構成するプロセスごとにリスク分類とその対策に基づいて、階層的に主張を分解する。
　たとえば、図 5.2.2-1 に示す階層分解パターンでは、プロセス層、リスク分類層、

リスク分解層、リスク対策主張層、リスク対策証跡層によって、ディペンダビリティケースを階層的に作成できる。まず、親ノードで、プロセスごとの分解の根拠となる対象を示すために、前提として「システムのプロセス構成定義」を親ノードに接続している。次いで、プロセスごとに、リスクを分類して定義しておき、リスク分類ごとにディペンダビリティを確認するために、前提ノードを記述する。たとえば、図 5.2.2-1 では、システムがプロセス A と B から構成されているので、「プロセス A に対するリスク分類」と「プロセス A に対するリスク分類」を前提ノードとして記述している。

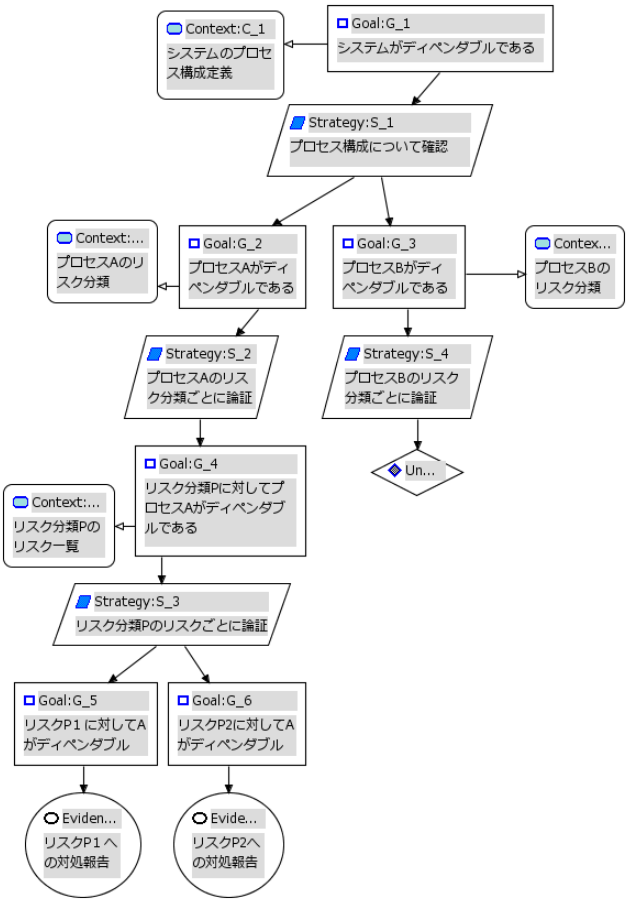


図 5.2.2-1　階層分解パターン

さらに、プロセス分類に属するリスクごとに、プロセスでリスク対策できていることを確認する。図 5.2.2-1 では、プロセス A には、リスク分類 P があり、P のリスク P1 と P2 がある。これらのリスクに対して、それぞれ「リスク P1 に対して A がディペンダブルである」と「リスク P1 に対して B がディペンダブルである」の主張が必要となるから、それらの主張の証跡を記述している。

【特性】
(利点)システムのプロセス構成とリスク分類に従って、システムに対するディペンダビリティケースを自然に作成できる。
(欠点)システムのプロセス定義とプロセスに対するリスク分類が定義できなければ、階層分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 2 点がある。
(1)階層分解では、ディペンダビリティの各階層で記述すべき内容を予め定義する必要がある。
(2)階層分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.2.3 DFD 階層分解パターン

【分解問題】データフロー図(DFD、 Data Flow Diagram)で分析されたシステムに対してディペンダビリティケースを作成する必要がある。
【分解状況】DFD 階層分解を適用する場合の前提となる状況は、データフロー図によってシステムが明確に定義できることである。
【解決策】DFD の階層構成に基づいて、主張を分解する
データフロー図は、□システムの外部とシステムとのデータフローを定義する最上位の**コンテクスト図**と、□コンテクスト図をプロセスによって階層的に分解する下位の**データフロー図**、□それ以上分解できないプロセスに対する**プロセス仕様**、□データを格納する**データストア**からなる。
したがって、DFD 階層に対して、以下の手順でディペンダビリティケースを作成できる。
【手順 1】コンテクスト図に基づいてプロセス分解パターンにより、主張「システムはディペンダブルである」を分解する。このとき、前提として「データフロー

図の定義」をこの主張に接続する。

【手順 2】プロセスが分解できる場合。

プロセスごとに、プロセス分解パターンにより、ディペンダビリティケースを作成する。このとき、上位のプロセスに対応する親ノードがディペンダビリティケースに必ず存在することから、プロセス構成の定義を親ノードに前提ノードとして接続する。

【手順 3】プロセスが分解できない場合、以下の 2 つの場合がある

【手順 3-1】プロセス仕様が定義されている場合

プロセス仕様に対するディペンダビリティケースを作成する。このとき、プロセス仕様では処理の内容が記述されているから、処理手順ごとにリスク分析を実施して、そのリスクに対応できる証跡を示すことができる。

【手順 3-1】プロセス仕様が定義されていない場合

主張に対して、未定義要素を接続する。

上述した手順によって作成できるディパンダビリティケースとデータフロー図の階層的な対応関係の例を図 5.2.3-1 に示す。なお、PxCnf は「プロセス Px がディペンダブルである」、I x は「入力がディペンダブルである」、Ox は「出力がディペンダブルである」、PxSpec は「プロセス仕様 Px がディペンダブルである」ことをそれぞれ示している。

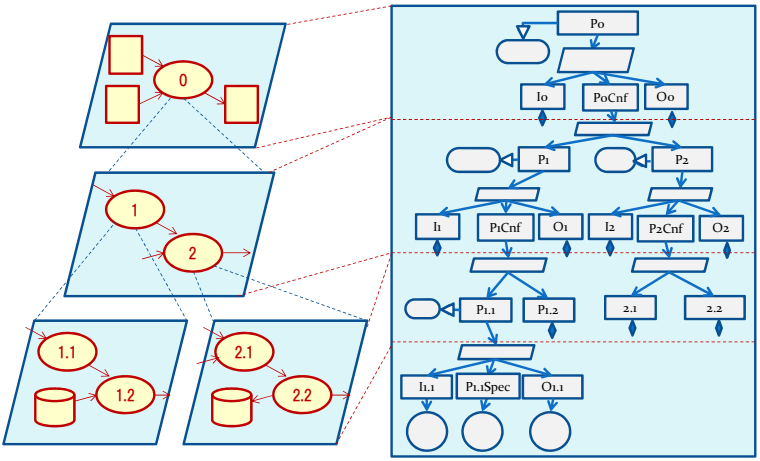


図 5.2.3-1　DFD 階層分解パターンの例

【特性】
(利点)システムに対するデータフロー図に従って、システムに対するディペンダビリティケースを自然に作成できる。
ビジネスプロセスやワークフローを構成する一連のプロセスに対しても同様にしてディペンダビリティケースを作成できる。
(欠点)システムに対するデータフロー図が定義できなければ、DFD 階層分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 3 点がある。
(1)DFD 階層とディペンダビリティケースの階層を対応させる必要がある。
(2)DFD 階層分解の根拠を示す前提ノードの接続方法を、以下のいずれにするかを決める必要がある。
－DFD 全体に対応させることで親主張ノードだけに接続する
－DFD 階層ごとに前提ノードを用意しておき、対応する主張ノードに接続する
(3)データストアの扱いを明確にしておく必要がある。たとえば、「データストアがディペンダブルである」などの主張をどのようにして確認するかを決めておく必要がある。具体的には「データストアの更新がディペンダブルである」「データストアの管理がディペンダブルである」などに分解できる。

5.2.4 プロセス関係分解パターン

【分解問題】依存関係を持つプロセスに対してディペンダビリティケースを作成する必要がある。
【分解状況】プロセス関係分解を適用する場合の前提となる状況は、先行後行関係によってプロセス間の関係が明確に定義できることである。
【解決策】先行プロセスの出力証跡が、後続プロセスの入力についての主張の前提になる
先行プロセスと後続プロセスの関係に基づいて主張を確認することができる。
たとえば、図 5.2.4-1 では、「先行作業の出力結果報告」が先行作業の証拠ノードと、後続作業の前提ノードとで用いられている。図では、同じ名前であることを分かりやすくするため、点線でこのことを示している。ただし、GSN には、このような表記法はないことを注意しておく。

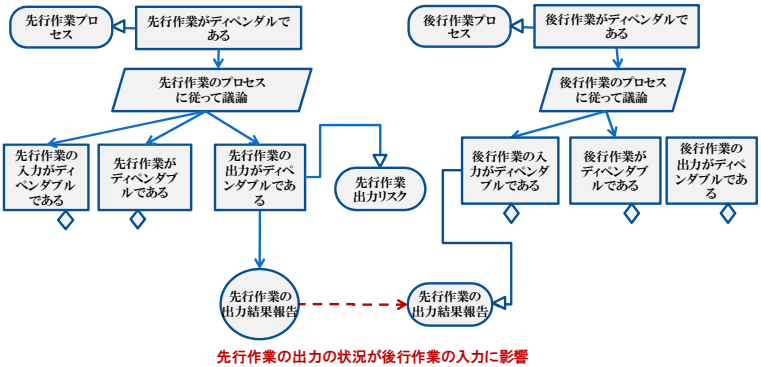


図 5.2.4-1　プロセス関係分解パターンの例

【特性】

(利点)プロセスの先行後行関係に従って、システムに対するディペンダビリティケースを自然に作成できる。

(欠点)プロセスの先行後行関係が定義できなければ、プロセス関係分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の点がある。

証拠と前提の接続関係はディペンダビリティケースの下になっているＧＳＮでは定義されていないことに注意が必要である。したがって、先行作業の結果が後行作業の入力になる場合には対応関係を明示するために、名前を一致させるなどの工夫が必要である。

5.2.5　ECA 分解パターン

【分解問題】イベント(E、Event)、条件(C、Condition)、処理(A、Action)に対してディペンダビリティケースを作成する必要がある。

【分解状況】ECA 分解を適用する場合の前提となる状況は、イベント、条件、処理が明確に定義できていることである。

【解決策】イベント、条件、活動に対して主張を分解する。

ECA 分解パターンに対するディペンダビリティケースの例を図 5.2.5-1 に示す。

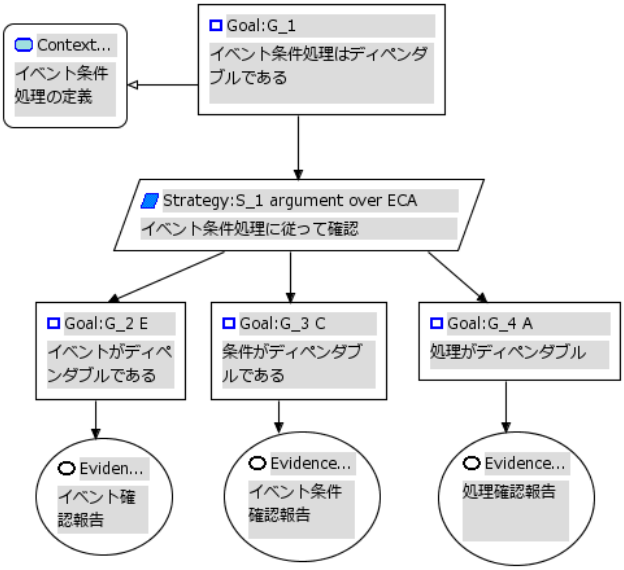


図 5.2.5-1　ECA 分解パターンの例

【特性】

(利点)ECA の定義に従って、システムに対するディペンダビリティケースを自然に作成できる。

(欠点)ECA が定義できなければ、ECA 分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の点がある。

ECA ごとにディペンダビリティケースを作成する必要である。したがって、複数の ECA の組合せに対して「ECA 一覧」を親主張に対する前提ノードとして接続することにより、ECA ごとに分解する必要がある。

5.2.6　条件選択分解パターン

【分解問題】条件判断に基づいて処理に対するディペンダビリティケースを作成する必要がある。

【分解状況】条件選択分解を適用する場合の前提となる状況は、選択条件ごとに処理が明確に定義できていることである。

【解決策】条件判断に対して、主張を分解する。

条件選択分解パターンの例を図 5.2.6-1 に示す。

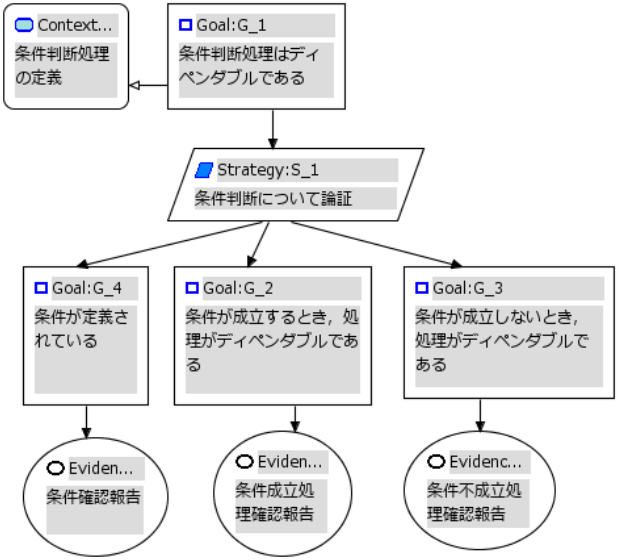


図 5.2.6-1　条件選択分解パターンの例

【特性】

(利点)条件選択処理に対するディペンダビリティケースを自然に作成できる。

(欠点)条件選択処理が定義できなければ、条件選択分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の点がある。

(1)条件が成立する場合だけでなく不成立の場合に対してもディペンダビリティを確認する必要がある。

(2)階層分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.2.7　代替案比較分解パターン

【分解問題】代替案比較に基づいてディペンダビリティケースを作成する必要がある。

【分解状況】代替案比較分解を適用する場合の前提となる状況は、代替案ごとに内容が明確に定義できていることである。

【解決策】代替案選択に対して、主張を分解する。

代替案比較分解パターンの例を図 5.2.7-1 に示す。

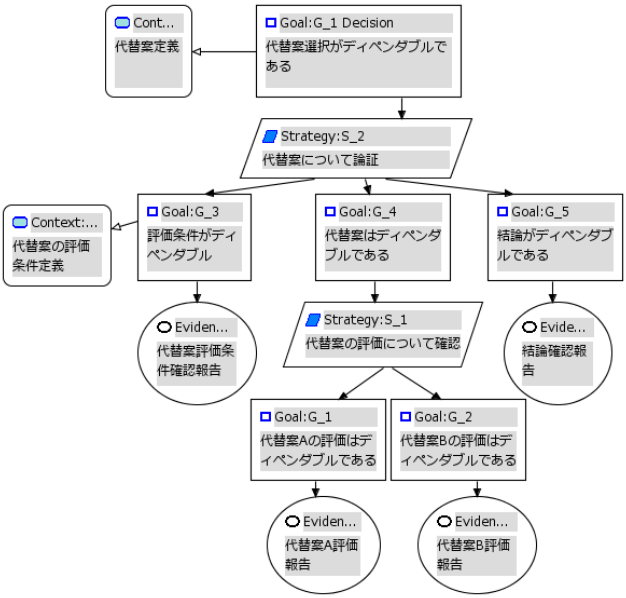


図 5.2.7-1　代替案比較分解パターンの例

【特性】

(利点)代替案に対するディペンダビリティケースを自然に作成できる。

(欠点)代替案が定義できなければ、代替案比較分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の点がある。

(1)すべての代替案に対してディペンダビリティを確認する必要がある。

(2)代替案比較分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.2.8　矛盾解決分解パターン

【分解問題】矛盾解決に基づいてディペンダビリティケースを作成する必要がある。

【分解状況】矛盾解決分解を適用する場合の前提となる状況は、矛盾内容とその対策が明確に定義できていることである。

【解決策】矛盾とその解決策に対して、主張を分解する。

矛盾解決分解パターンの例を図 5.2.8-1 に示す。この例では、矛盾には解決できる矛盾、解決できないが解消できる矛盾、回避できる矛盾があることに基づいて、分解している。

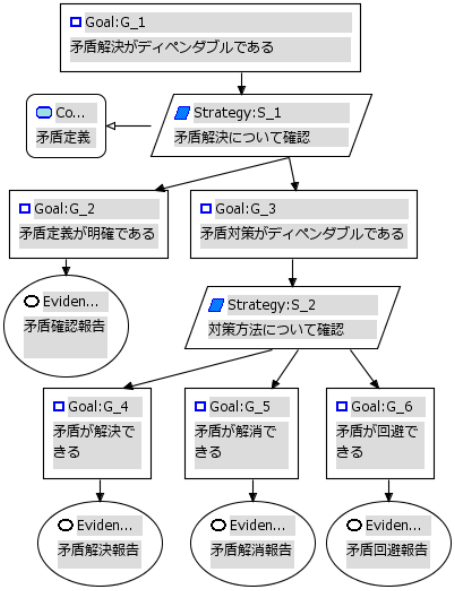


図 5.2.8-1　矛盾解決分解パターンの例

【特性】
(利点) 矛盾とその解決策に対するディペンダビリティケースを自然に作成できる。
(欠点)矛盾とその解決策が定義できなければ、矛盾解決分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の点がある。
矛盾解決分解の根拠を示す前提ノード「矛盾定義」を上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

5.2.9 ビュウ分解パターン

【分解問題】システムのビュウに基づいてディペンダビリティケースを作成する必要がある。

【分解状況】ビュウ分解を適用する場合の前提となる状況は、ビュウが明確に定義できていることである。

【解決策】ビュウ構成に基づいて、主張を分解する。

矛盾解決分解パターンの例を図 5.2.9-1 に示す。この例では、ソフトウェアアーキテクチャを構成する論理ビュウ、プロセスビュウ、配置ビュウ、物理ビュウ、ユースケースビュウからなる**4＋1 ビュウモデル**に従って分解している。なお、ビュウに対する主張の分解については省略している。

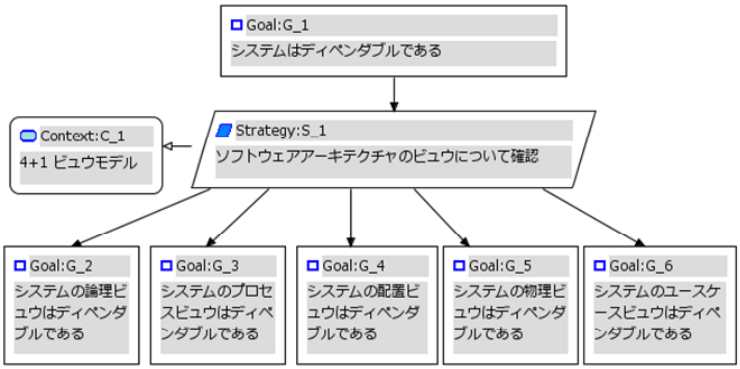


図 5.2.9-1　ビュウ分解パターンの例

【特性】

(利点)システムのビュウに対してディペンダビリティケースを自然に作成できる。

(欠点)ビュウが定義できなければ、ビュウ分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の点がある。

ビュウ分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。図 5.2.9-1 では、ビュウモデルは、具体的なシステムとは独立に定義されることから、戦略ノードに接続している。

5.2.10　ユースケース分解パターン

【分解問題】システムのユースケースに基づいてディペンダビリティケースを作成する必要がある。
【分解状況】ユースケース分解を適用する場合の前提となる状況は、ユースケースが明確に定義できていることである。
【解決策】システムのユースケースに基づいて、主張を分解する。

表 5.2.10-1　ユースケースの構成要素[4]

構成要素	説明
目的	ユースケースによって達成しようとしていること
アクター	だれ(あるいはシステム)がユースケースを起動するのか
事前条件	ユースケースが起動できるための条件
事後条件	ユースケースが完了したときに満たされる条件
基本シナリオ	刺激を受けてシステムに対して実行される
代替シナリオ	指定された特別な条件に対して実行される
例外シナリオ	例外条件に対して実行される

ユースケース分解パターンの例を図 5.2.10-1 に示す。この例では、システムが複数のユースケースから構成されると仮定して、ユースケースごとに表 5.2.10-1 で示したユースケースを構成する要素に従って分解している。図中の記号 U1 や Un は、これらのユースケースの識別子である。なお、図では 2 個のユースケースしか示していないが、ユースケースの個数だけ分解する必要がある。

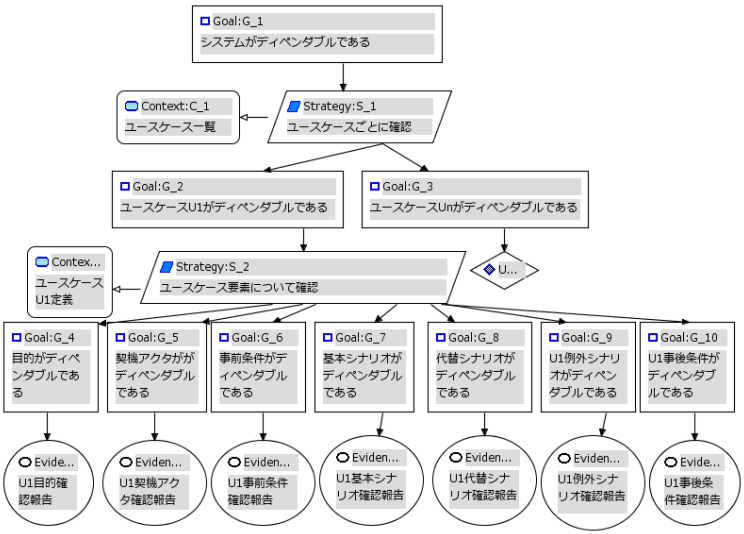


図 5.2.10-1　ユースケース分解パターンの例

【特性】

(利点) システムのユースケースに対してディペンダビリティケースを自然に作成できる。

(欠点) システムのユースケースが定義できなければ、ユースケース分解パターンを適用できない。

【留意点】適用する際の注意事項として、次の 2 点がある。

(1)図 5.2.10-1 では、基本シナリオ、代替シナリオ、例外シナリオに対する証拠をこれらに対する主張に対して直接接続しているが、シナリオも処理で有ることから、これらの処理に対するリスクを分析してプロセス分解パターンによってより詳しく分解することもできる。

(2)ユースケース分解の根拠を示す前提ノードを上位の主張ノードに接続するか、戦略ノードに接続するかを決める必要がある。

参考文献

[1] F.ブッシュマン，R.ムニエ，H．ローネルト，P.ゾンメルラード，M.スタル，ソフトウェアアーキテクチャ，ソフトウェア開発のためのパターン体系，Pattern-Oriented Software Architecture：A System Of Patterns，1996，John & Wiley & Sons, Ltd.，近代科学社，2000

[2] Robin Bloomfield and Peter Bishop, Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective

[3] 松野裕，山本修一郎，高井利憲， D-Case 入門，～ディペンダビリティ・ケースを書いてみよう！～，ダイテックホールディング，2012，ISBN 978-4-86293-079-8

[4] Eriksson, H., and Penker, M., UML Toolkit，杉本他監訳，UML ガイドブック，トッパン，1998

6 まとめ

本書の内容は、CREST「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」研究領域(DEOS プロジェクト)の支援を受けて、筆者らが名古屋大学で実施した研究に基づいている。とくに、D-Case 実証評価研究会で筆者らとともにディペンダビリティケースの講習会に参加された方々をはじめ、ご協力をいただいた皆様に、この場を借りて深く感謝したい。

また、本書では、筆者らによる「D-Case 入門～ディペンダビリティ・ケースを書いてみよう！～」で紹介した内容を、読者による自習ができることを目的として、できるだけ分かりやすく説明することを試みている。この理由は「D-Case 入門」を研修用テキストとして執筆したことから、講師による解説がないと同書だけでは理解が難しいという問題があったためである。これまでに同書を用いて実施してきたディペンダビリティケース講習会の経験を本書に可能な限り反映することを試みているが、扱っている内容の先端性のために必ずしも内容が十分成熟しているとはいえない可能性があることをお断りしておきたい。しかし本書の内容が現時点でのディペンダビリティケースに関する世界最高水準にあると筆者らが自負していることも、ここで指摘しておきたい。この理由は、ディペンダビリティケースについて本書ほど具体的に作成手順を解説するだけでなく、GSN の基礎的な文法の解説とその確認問題を示しているような教科書がないことである。この点で、本書は世界初のディペンダビリティケースのまとまった教科書であるといえる。これまで、ディペンダビリティケースの技法を学ぶためには、英語で記述された断片的な論文を個別的に読むしかなかった。研究者にはできても現場で多忙な技術者にとって、このような余裕を見つけることは難しい。本書を執筆した動機の一つがこの点にある。

ディペンダビリティケースは、本文で紹介したように、ISO26262 機能安全規格によって制度化が進んでいる。今後、現場の技術者がディペンダビリティケース技術を習得して活用する機会が拡大する可能性が高い。本書の内容がディペンダビリティケース技術を必要とする現場の技術者の皆さんのお役に立つことを期待している。

最後に、本書を執筆することによって明らかとなった新たな課題もまた多い。今後もこれらの課題に対して研究を継続していく予定である。

演習問題の答え

3.2 演習問題の回答例

【問題１】省略

【問題２】省略

【問題３】最上位の主張から他の主張への関係線が前提関係を示す白抜き矢線になっている

【問題４】戦略と下位の主張の関係を示す矢線の方向が逆である

【問題５】戦略に対して証拠が関係付けられている

【問題６】最上位の主張が下位の主張に分解されているのに、保留関係も記述されている

【問題７】前提から戦略への関係がある

【問題８】５

【問題９】１

【問題１０】３

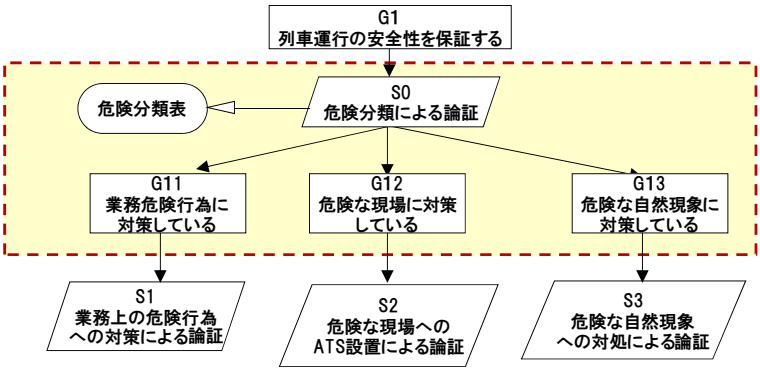
【問題１１】前提として「サービスの可用性を保証する」という主張が記述されている。また、前提への矢線が白抜きではない。

【問題１２】「SCS 作成報告書」という証拠に用いる名称が主張名になっている

【問題１３】「サービスリスク」は主張名として不適切である

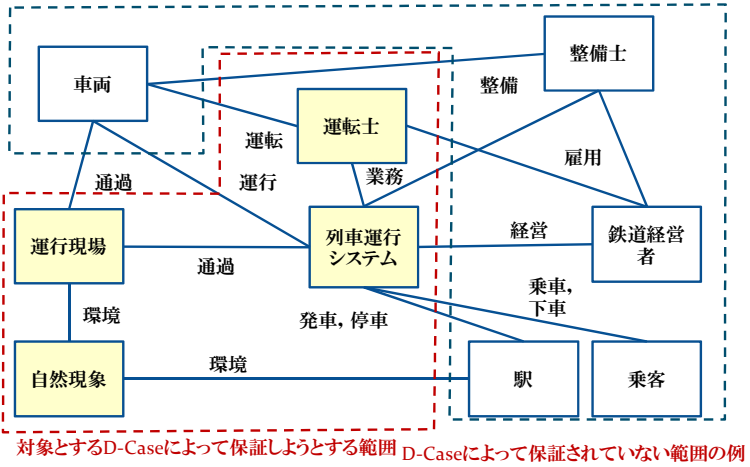
【問題１４】

主張 G1 から戦略 S1、S2、S3 への議論が省略されている。このため、以下のように、戦略 S0、主張 G11、G12、G13 を補うとよい。また、このとき、前提として危険分類表をきじゅつすることにより、なぜ、G1 を主張 G11、G12、G13 に分解したのかを説明できるようにする。



【問題１５】

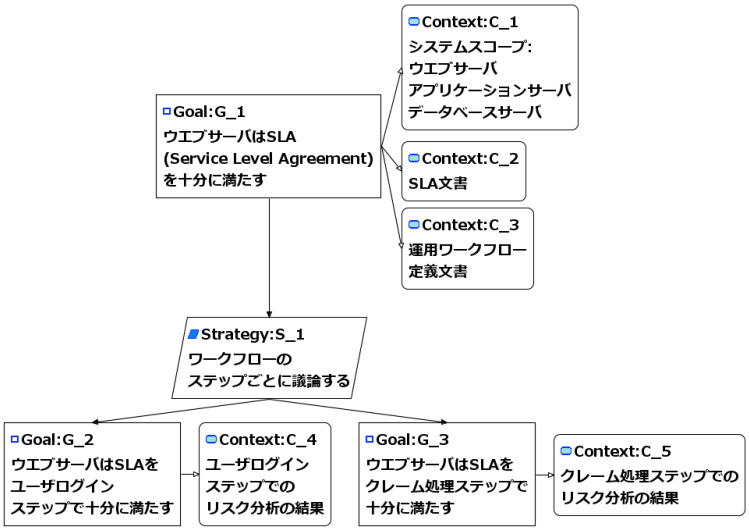
以下のようなコンテクスト図を用いて、対象としている D-Case のスコープを明示することができる。この図から、列車運行システムの車両、駅、乗客、鉄道経営者、整備士などについてリスク分析をしていないことが分かる。したがって、これらのリスクについては、別の D-Case を用いて安全性を確認する必要がある。



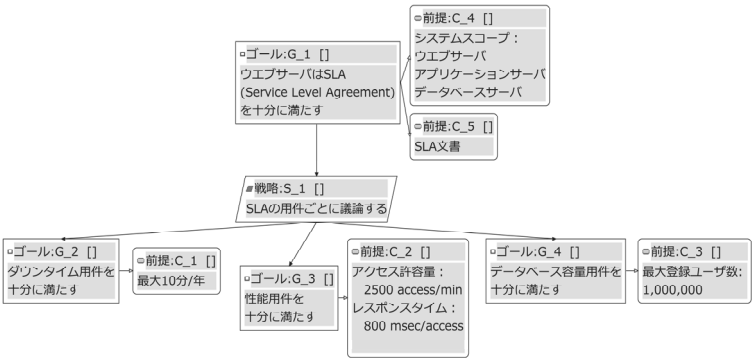
4.2 演習問題の解答例

【問題１】ウエブサーバシステム

本問は 4.1 節で例題として使った例である。解答例を下に再掲する。



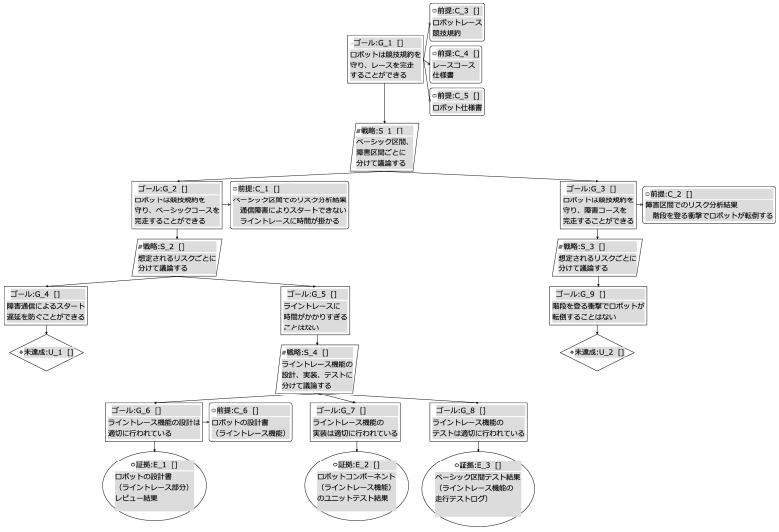
ここでは、4.1 節の答案例以外の D-Case の議論構造を考える。上の解答例では、トップレベルの議論は、ワークフローのステップごとであったが、SLA を十分に満たすことを言うのだから、「SLA 文書」に書かれている要件ごとに場合分けするのも自然である。以下にそれに沿った解答例を示す。



G_2 から G_4 のゴールの展開は、それぞれの要件を脅かすリスクごとに場合分けして、それぞれに対処できることによって行われうる。しかしこの議論構造だと、運

用ワークフローの流れが見えなくなる。またリスク分析がその運用ワークフローにそって行われているので、G_2 から G_4 の議論の展開のためにリスクを SLA 要件ごとにまとめ直す必要がある。どちらの議論構造がよいかは、ウエブサーバの専門家とより議論したいが、運用における障害が深刻になっている現況も考えると、4.1 で示した、運用ワークフローでまず展開した解答例がよいと考える。

【問題２】ロボットレース
まず解答を示す。



4.1 節のステップ３から始める。
ステップ３　トップゴールを置く。レースであるから、勿論優勝することが目標である。しかしそれだけを目標にすると、大変である。ここでは、まず競技規約を守り、レースを完走することをディペンダビリティ要件とした。よってトップゴールを「ロボットは競技規約を守り、レースを完走することができる」とした。

ステップ４　ディペンダビリティ要求、環境情報、語彙定義を前提としてトップゴールにおく。競技規約を守りながら走るので、「ロボットレース競技規約」を前提ノードにおく。環境情報として、「レースコース仕様書」を置く。システムのスコープはロボットであるので、「ロボット仕様書」も前提ノードとしておく。

ステップ５　大まかな D-Case の構造を考える。コースはベーシックコースと障害コースがあり、走行するための条件が異なる。そのため参加者はベーシックコースと障害コースを分けて対策を行う(また、実際の ET ロボットコンテストでは、ベーシックコースと障害コースでは、前者が走行時間、後者が障害をクリアできるかを主に採点される)ので、まずベーシックコース、障害コースに分けるのが自然である。次の分け方は、それぞれのコースごとにリスクごとに場合分けして議論するとよい。

ステップ６　必要なドキュメントを前提として置く。リスクごとに場合分けして議論するための、それぞれのコースのリスク分析結果を G_2、 G_3 におく。

ステップ７　ドキュメントから D-Case のサブツリーを作る。G_2、 G_3 はそれぞれの前提ノードに付けられたリスク分析結果のリストに従って、サブゴールに展開できる。

ステップ８　サブツリーができていない部分を典型的な議論構造を使って作る。リスクに対処できていることを、それぞれのリスクごとに議論する必要がある。ここでは、設計、実装、テストのライフサイクルのそれぞれのフェーズで得られる結果を証拠とするような議論構造を考えた。これはライフライクルに沿った議論の一種であり、典型的な議論構造の一種である。

ステップ９　上記を必要なだけ繰り返す。この解答例でいいかというと、いろいろ問題がある。例えば、ベーシックコースと障害コースしか考えていないが、障害が発生しやすいのは、スタート直後と、ベーシックコースと障害コースが切り替わるポイントであるので、その部分の考慮が足りない(これは実際に ET ロボコンを経験された方がこの問題を解いてもらったときに頂いた意見である)。さらに、環境情報として、「レースコース仕様書」が置かれているが、実際のレースコースのコンディションが前提にない。2012 年度の ET ロボットコンテストでは、途中リタイアが続出した。去年よりレースコースの照明の設定が違っていて、うまくライントレースできなかったそうである。実際のレースコースのコンディションを前提ノードに置く必要がある。

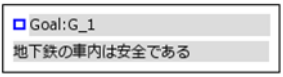
2012 年度の ET ロボットコンテストでは、富士ゼロックスのチームが D-Case を使ってモデル部門の最優秀賞(エクセレントモデル)を受賞した。書かれた D-Case

は公開される予定なので、ぜひ参考にしてほしい。

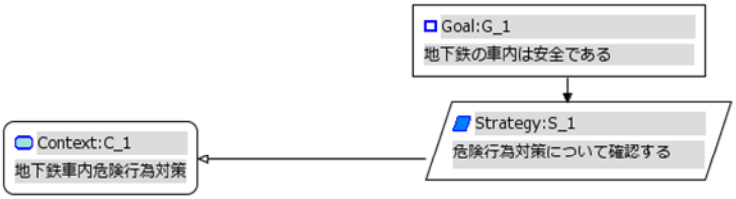
【問題３】地下鉄の車内安全性

(解説)

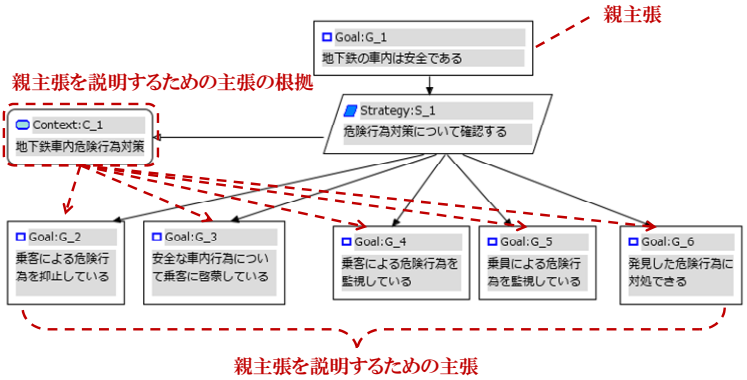
まず、最上位の主張を考える



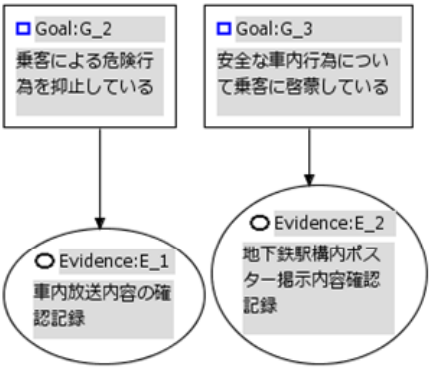
主張を展開する理由を考える



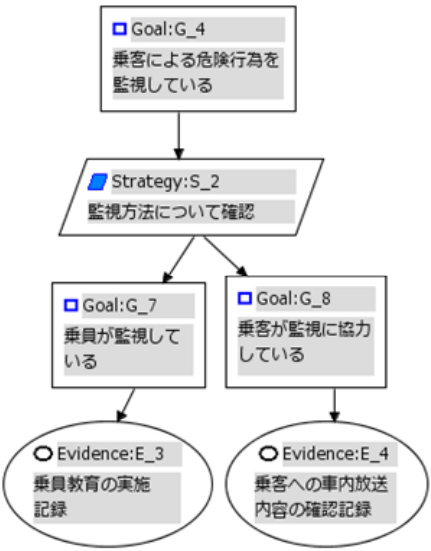
展開した主張によって親主張を説明できるか？



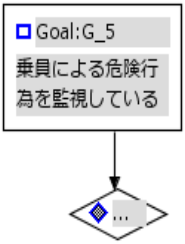
下位の主張に対する証跡を考える



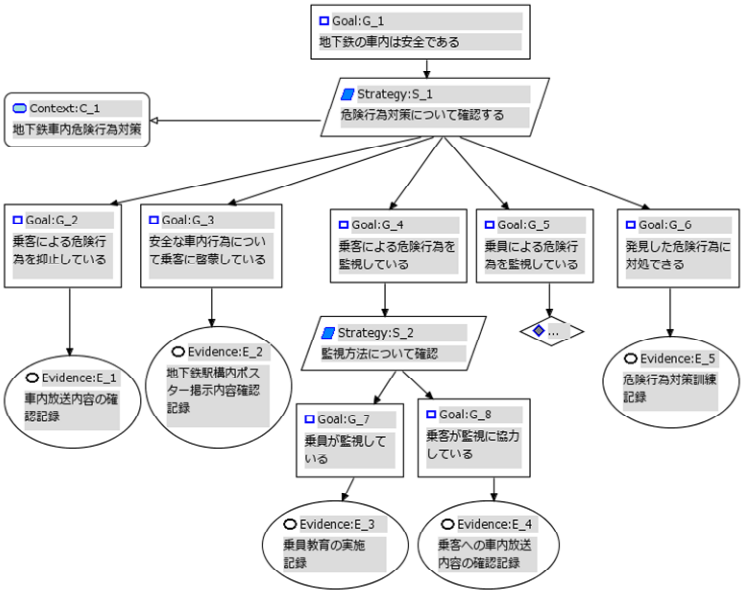
もし、下位の主張に対する直接的な証跡が対応付けられない場合、下位の主張を分解して説明することを考える。



主張の展開を保留する



結果をまとめる



地下鉄車内の安全性確認例

【問題２】電気ケトル

(解説)

まず電気ケトルの操作を考えると、以下のようになる。

1. 取っ手を持つ
2. 蓋をあける

3. タンクに水を入れる
4. 電源供給部とコンセントを繋ぐ
5. ポットを電源供給部に置く
6. スイッチを入れる
7. ランプが点灯する
8. スイッチを切る
9. ランプが消灯する
10. お湯が沸いたことを確認する
11. 取っ手をもってお湯を注ぐ

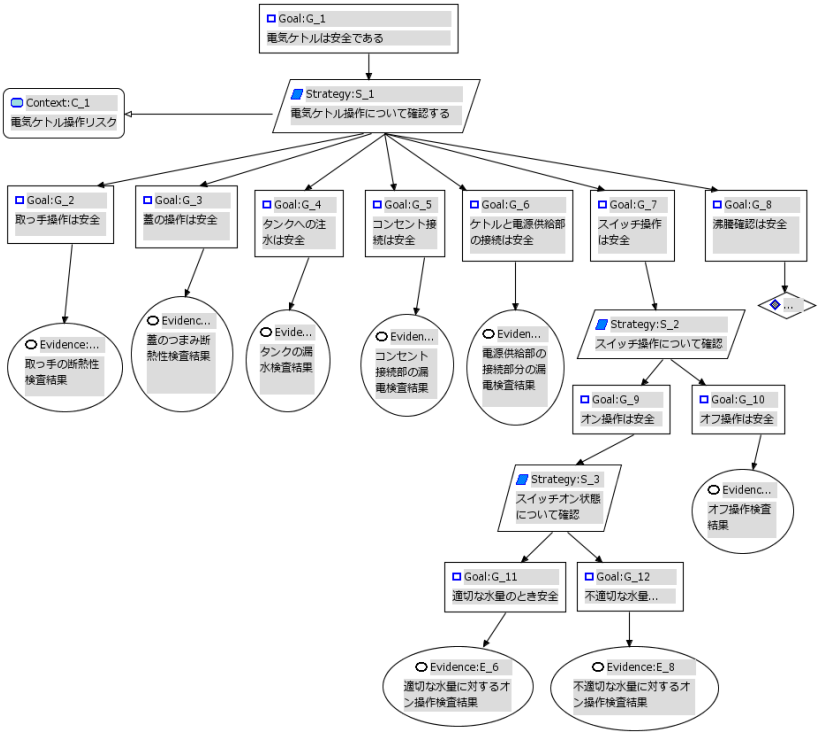
次に、これらの操作に対するリスクを下表のようにして列挙する。

	操作	リスク	影響	確認範囲
1	取っ手を持つ	やけどする	大	○
2	蓋をあける	やけどする	大	○
3	タンクに水を入れる	タンクから漏水する	大	○
4	電源供給部とコンセントを繋ぐ	出火する	大	○
5	ポットを電源供給部に設置する	設置できない	大	○
		電源を供給できない	大	○
6	スイッチを入れる	スイッチが入らない	中	
		水を入れ忘れて空焚きする	大	○
7	ランプが点灯する	点灯しない	中	
8	スイッチを切る	スイッチが切れない	大	○
9	ランプが消灯する	ランプが消灯しない	中	

操作リスクに対する安全性の主張を列挙する

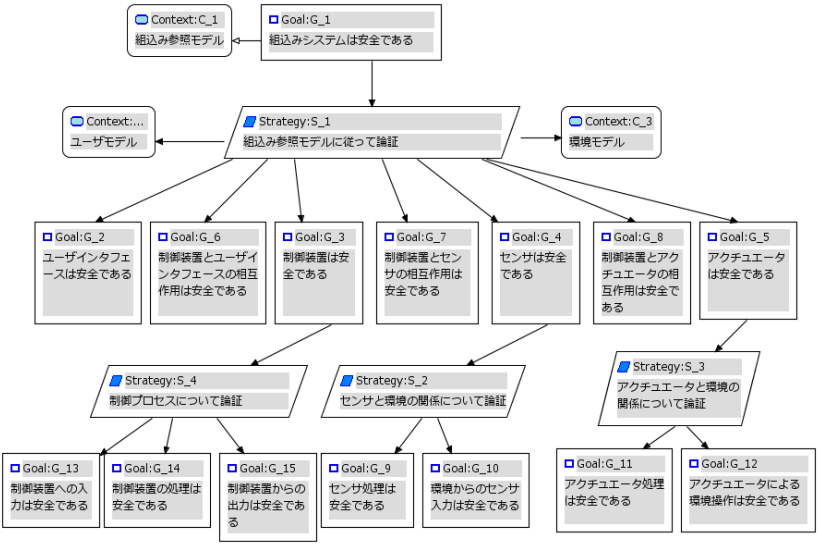
操作	安全性の主張
取っ手を持つ	やけどしない
蓋をあける	やけどしない
タンクに水を入れる	タンクから漏水しない
電源供給部とコンセントを繋ぐ	出火しない
ポットを電源供給部に設置する	設置できる
	電源を供給できる
スイッチを入れる	空焚きしない
スイッチを切る	スイッチが切れる

操作リスクに基づいて D-Case を作成した例を下図に示す。


Goal:G_1
電気ケトルは安全である
Context:C_1
電気ケトル操作リスク
Strategy:S_1
電気ケトル操作について確認する
Goal:G_2
取っ手操作は安全
Goal:G_3
蓋の操作は安全
Goal:G_4
タンクへの注水は安全
Goal:G_5
コンセント接続は安全
Goal:G_6
ケトルと電源供給部の接続は安全
Goal:G_7
スイッチ操作は安全
Goal:G_8
沸騰確認は安全
Evidence:...
取っ手の断熱性検査結果
Evidenc...
蓋のつまみ断熱性検査結果
Evide...
タンクの漏水検査結果
Eviden...
コンセント接続部の漏電検査結果
Eviden...
電源供給部の接続部分の漏電検査結果
Strategy:S_2
スイッチ操作について確認
Goal:G_9
オン操作は安全
Goal:G_10
オフ操作は安全
Strategy:S_3
スイッチオン状態について確認
Evidenc...
オフ操作検査結果
Goal:G_11
適切な水量のとき安全
Goal:G_12
不適切な水量...
Evidence:E_6
適切な水量に対するオン操作検査結果
Evidence:E_8
不適切な水量に対するオン操作検査結果

【問題３】組込みシステムの論理参照モデル

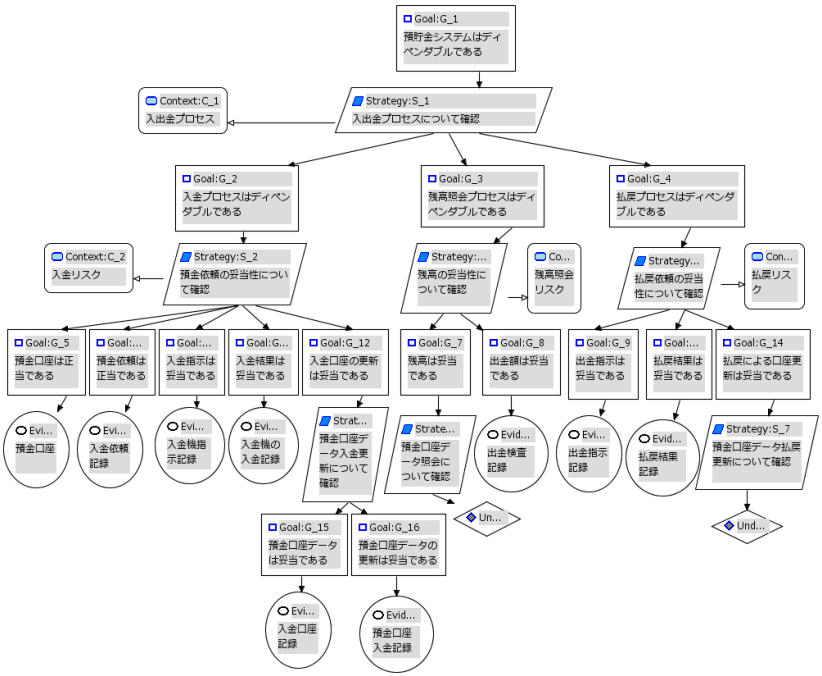
(作成例)注意：最下位の主張に対する保留関係は省略している



【問題４】預貯金システム

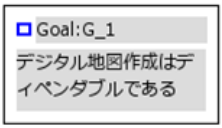
(作成のポイント)

預貯金システムに対するデータフロー図のプロセスごとに分解することにより D-Case を作成できる。データストア「預金口座データ」に対する照会、更新についても、ディペンダビリティについての主張を分解・定義することにより、確認している。

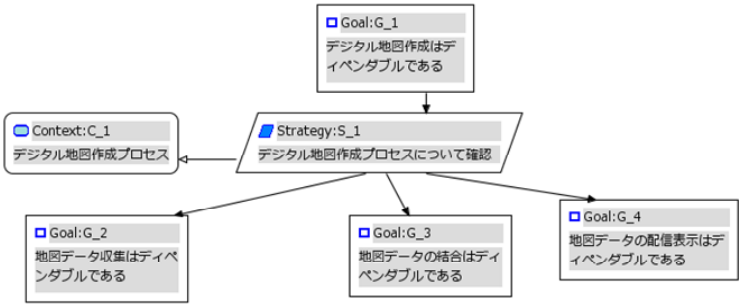


【問題５】デジタル地図作成システム

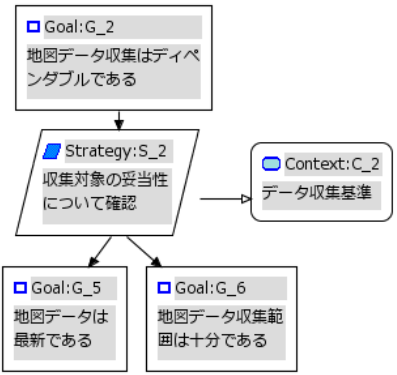
(解説)まず結論となる主張を書く。



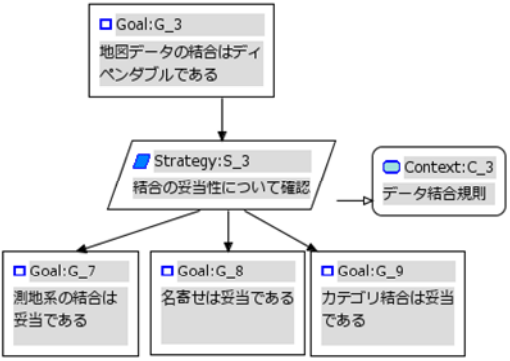
主張をプロセス分解する



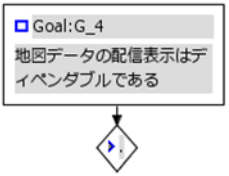
地図データ収集プロセスの主張を分解



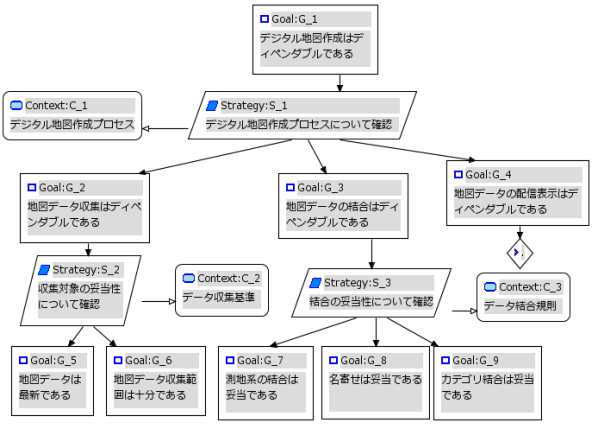
地図データ結合プロセスの主張を分解する



保留する主張を決定する。



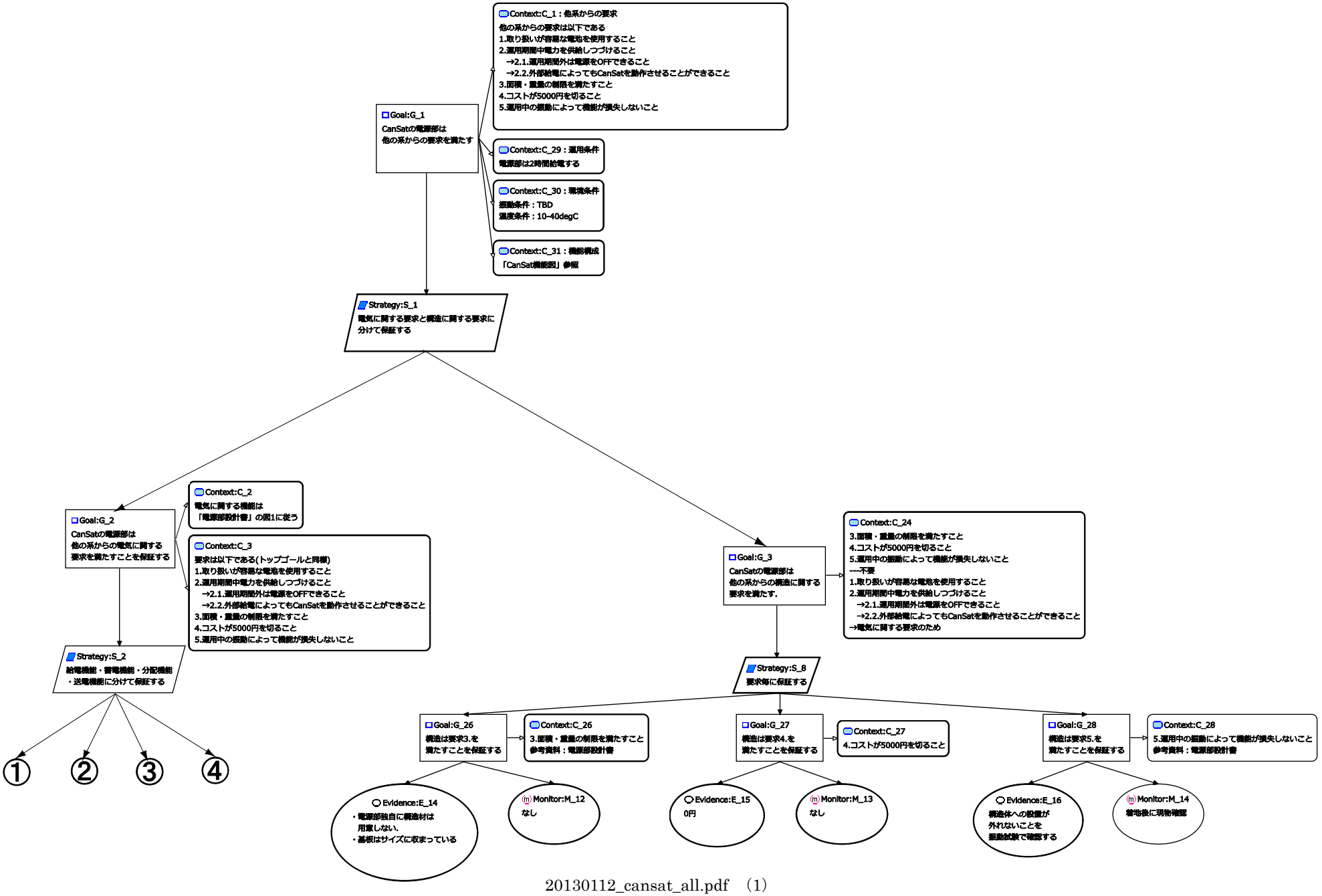
以上の検討をまとめて作成した D-Case を示すと以下のようになる。

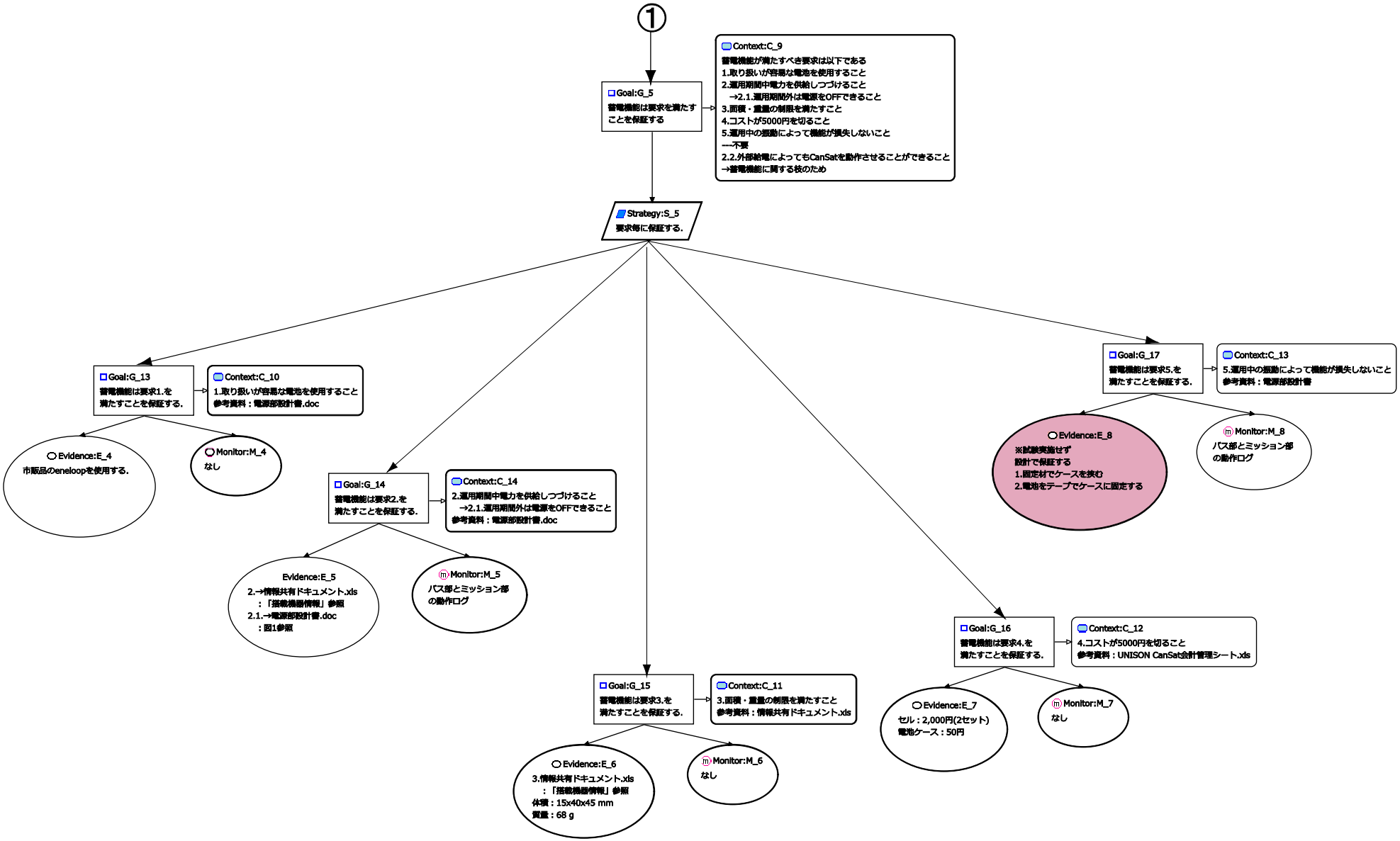


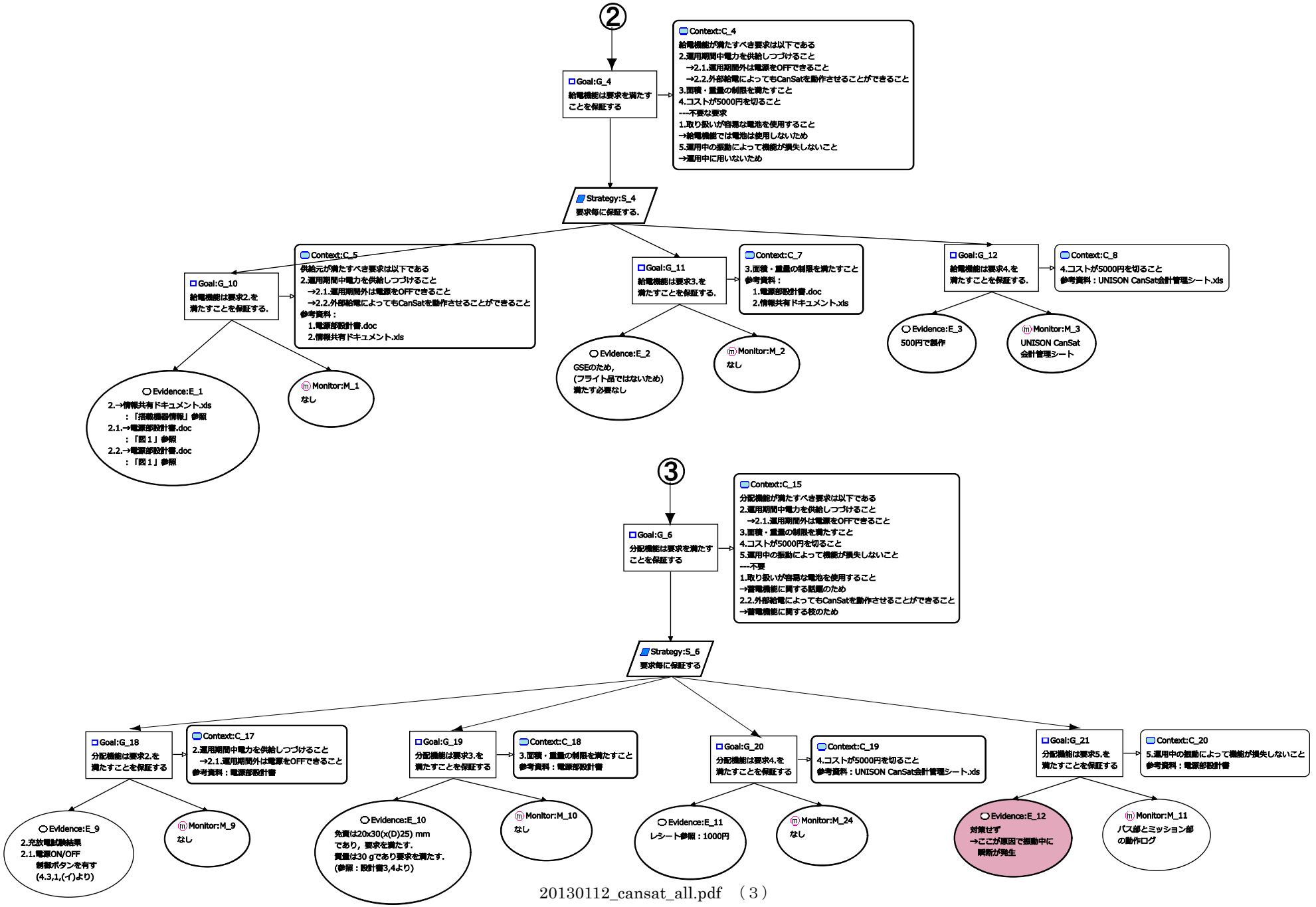
【問題６】

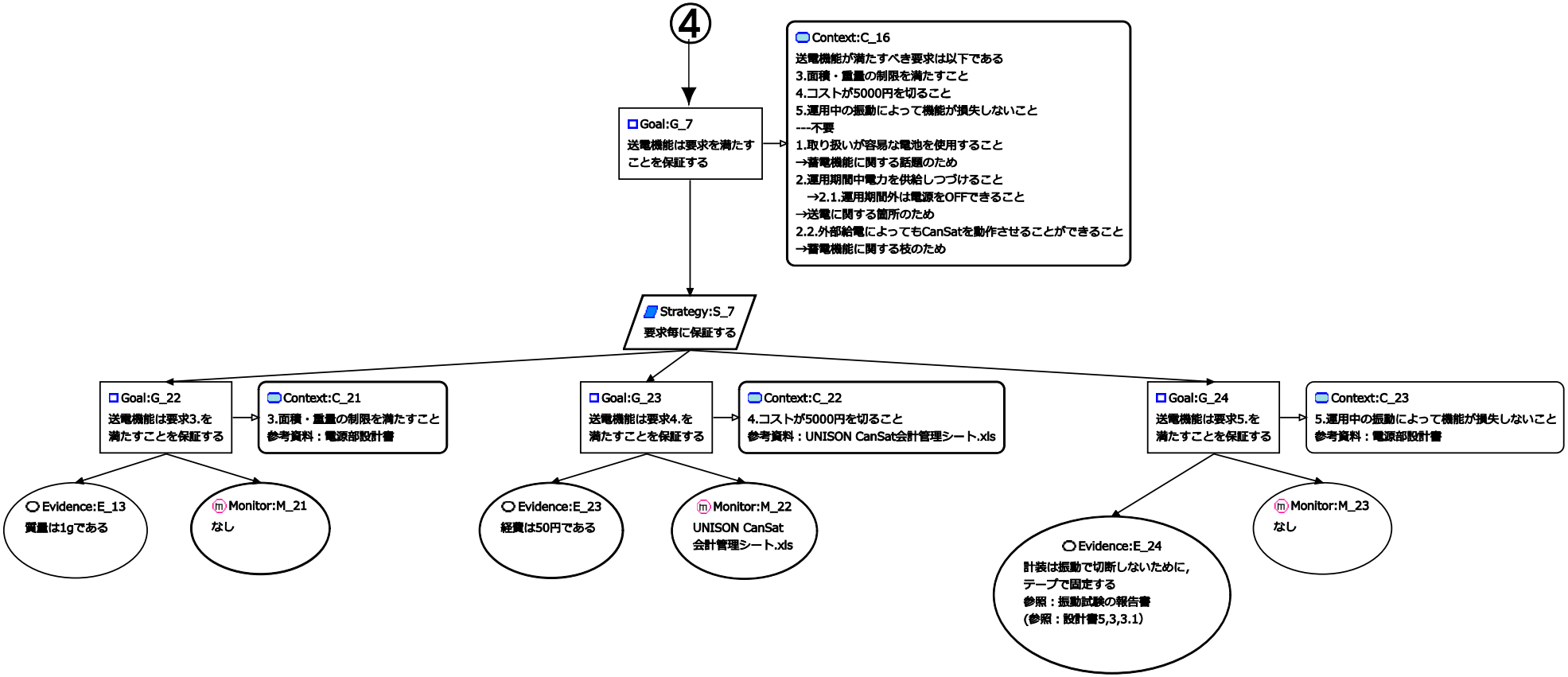
CanSat の電源部に関して D-Case を記述した結果を「20130112_cansat_all.pdf」に示す。

アシュアランスケースのトップゴールを「CanSat の電源部は他の系からの要求を満たす」と設定した。これに対してまずは機能の観点から分解を進める。これは、各ゴールに関わるインターフェースを少なくすることで、ゴールが保証出来ていることを明確に確認しやすくするためである。


Context:C_1：他系からの要求
他の系からの要求は以下である
1.取り扱いが容易な電池を使用すること
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
→2.2.外部給電によってもCanSatを動作させることができること
3.面積・重量の制限を満たすこと
4.コストが5000円を切ること
5.運用中の振動によって機能が損失しないこと
Goal:G_1
CanSatの電源部は
他の系からの要求を満たす
Context:C_29：運用条件
電源部は2時間給電する
Context:C_30：環境条件
振動条件：TBD
温度条件：10-40degC
Context:C_31：機能構成
「CanSat機能図」参照
Strategy:S_1
電気に関する要求と構造に関する要求に
分けて保証する
Goal:G_2
CanSatの電源部は
他の系からの電気に関する
要求を満たすことを保証する
Context:C_2
電気に関する機能は
「電源部設計書」の図1に従う
Context:C_3
要求は以下である(トップゴールと同様)
1.取り扱いが容易な電池を使用すること
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
→2.2.外部給電によってもCanSatを動作させることができること
3.面積・重量の制限を満たすこと
4.コストが5000円を切ること
5.運用中の振動によって機能が損失しないこと
Strategy:S_2
給電機能・蓄電機能・分配機能
・送電機能に分けて保証する
①
②
③
④
Goal:G_3
CanSatの電源部は
他の系からの構造に関する
要求を満たす.
Context:C_24
3.面積・重量の制限を満たすこと
4.コストが5000円を切ること
5.運用中の振動によって機能が損失しないこと
---不要
1.取り扱いが容易な電池を使用すること
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
→2.2.外部給電によってもCanSatを動作させることができること
→電気に関する要求のため
Strategy:S_8
要求毎に保証する
Goal:G_26
構造は要求3.を
満たすことを保証する
Context:C_26
3.面積・重量の制限を満たすこと
参考資料：電源部設計書
Goal:G_27
構造は要求4.を
満たすことを保証する
Context:C_27
4.コストが5000円を切ること
Goal:G_28
構造は要求5.を
満たすことを保証する
Context:C_28
5.運用中の振動によって機能が損失しないこと
参考資料：電源部設計書
Evidence:E_14
・電源部独自に構造材は
用意しない.
・基板はサイズに収まっている
Monitor:M_12
なし
Evidence:E_15
0円
Monitor:M_13
なし
Evidence:E_16
構造体への設置が
外れないことを
振動試験で確認する
Monitor:M_14
着地後に現物確認


①
Goal:G_5
蓄電機能は要求を満たすことを保証する
Context:C_9
蓄電機能が満たすべき要求は以下である
1.取り扱いが容易な電池を使用すること
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
3.面積・重量の制限を満たすこと
4.コストが5000円を切ること
5.運用中の振動によって機能が損失しないこと
---不要
2.2.外部給電によってもCanSatを動作させることができること
→蓄電機能に関する枝のため
Strategy:S_5
要求毎に保証する.
Goal:G_13
蓄電機能は要求1.を満たすことを保証する.
Context:C_10
1.取り扱いが容易な電池を使用すること
参考資料：電源部設計書.doc
Evidence:E_4
市販品のeneloopを使用する.
Monitor:M_4
なし
Goal:G_14
蓄電機能は要求2.を満たすことを保証する.
Context:C_14
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
参考資料：電源部設計書.doc
Evidence:E_5
2.→情報共有ドキュメント.xls
：「搭載機器情報」参照
2.1.→電源部設計書.doc
：図1参照
Monitor:M_5
バス部とミッション部の動作ログ
Goal:G_15
蓄電機能は要求3.を満たすことを保証する.
Context:C_11
3.面積・重量の制限を満たすこと
参考資料：情報共有ドキュメント.xls
Evidence:E_6
3.情報共有ドキュメント.xls
：「搭載機器情報」参照
体積：15x40x45 mm
質量：68 g
Monitor:M_6
なし
Goal:G_16
蓄電機能は要求4.を満たすことを保証する.
Context:C_12
4.コストが5000円を切ること
参考資料：UNISON CanSat会計管理シート.xls
Evidence:E_7
セル：2,000円(2セット)
電池ケース：50円
Monitor:M_7
なし
Goal:G_17
蓄電機能は要求5.を満たすことを保証する.
Context:C_13
5.運用中の振動によって機能が損失しないこと
参考資料：電源部設計書
Evidence:E_8
※試験実施せず
設計で保証する
1.固定材でケースを挟む
2.電池をテープでケースに固定する
Monitor:M_8
バス部とミッション部の動作ログ


②
Goal:G_4
給電機能は要求を満たすことを保証する
Context:C_4
給電機能が満たすべき要求は以下である
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
→2.2.外部給電によってもCanSatを動作させることができること
3.面積・重量の制限を満たすこと
4.コストが5000円を切ること
---不要な要求
1.取り扱いが容易な電池を使用すること
→給電機能では電池は使用しないため
5.運用中の振動によって機能が損失しないこと
→運用中に用いないため
Strategy:S_4
要求毎に保証する.
Goal:G_10
給電機能は要求2.を満たすことを保証する.
Context:C_5
供給元が満たすべき要求は以下である
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
→2.2.外部給電によってもCanSatを動作させることができること
参考資料：
1.電源部設計書.doc
2.情報共有ドキュメント.xls
Goal:G_11
給電機能は要求3.を満たすことを保証する.
Context:C_7
3.面積・重量の制限を満たすこと
参考資料：
1.電源部設計書.doc
2.情報共有ドキュメント.xls
Goal:G_12
給電機能は要求4.を満たすことを保証する.
Context:C_8
4.コストが5000円を切ること
参考資料：UNISON CanSat会計管理シート.xls
Evidence:E_1
2.→情報共有ドキュメント.xls
：「搭載機器情報」参照
2.1.→電源部設計書.doc
：「図１」参照
2.2.→電源部設計書.doc
：「図１」参照
Monitor:M_1
なし
Evidence:E_2
GSEのため,
(フライト品ではないため)
満たす必要なし
Monitor:M_2
なし
Evidence:E_3
500円で製作
Monitor:M_3
UNISON CanSat
会計管理シート
③
Goal:G_6
分配機能は要求を満たすことを保証する
Context:C_15
分配機能が満たすべき要求は以下である
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
3.面積・重量の制限を満たすこと
4.コストが5000円を切ること
5.運用中の振動によって機能が損失しないこと
---不要
1.取り扱いが容易な電池を使用すること
→蓄電機能に関する話題のため
2.2.外部給電によってもCanSatを動作させることができること
→蓄電機能に関する枚のため
Strategy:S_6
要求毎に保証する
Goal:G_18
分配機能は要求2.を満たすことを保証する
Context:C_17
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
参考資料：電源部設計書
Goal:G_19
分配機能は要求3.を満たすことを保証する
Context:C_18
3.面積・重量の制限を満たすこと
参考資料：電源部設計書
Goal:G_20
分配機能は要求4.を満たすことを保証する
Context:C_19
4.コストが5000円を切ること
参考資料：UNISON CanSat会計管理シート.xls
Goal:G_21
分配機能は要求5.を満たすことを保証する
Context:C_20
5.運用中の振動によって機能が損失しないこと
参考資料：電源部設計書
Evidence:E_9
2.充放電試験結果
2.1.電源ON/OFF
制御ボタンを有す
(4.3,1,(イ)より)
Monitor:M_9
なし
Evidence:E_10
免責は20x30(x(D)25) mm
であり，要求を満たす.
質量は30 gであり要求を満たす.
(参照：設計書3,4より)
Monitor:M_10
なし
Evidence:E_11
レシート参照：1000円
Monitor:M_24
なし
Evidence:E_12
対策せず
→ここが原因で振動中に
瞬断が発生
Monitor:M_11
パス部とミッション部
の動作ログ


④
Context:C_16
送電機能が満たすべき要求は以下である
3.面積・重量の制限を満たすこと
4.コストが5000円を切ること
5.運用中の振動によって機能が損失しないこと
---不要
1.取り扱いが容易な電池を使用すること
→蓄電機能に関する話題のため
2.運用期間中電力を供給しつづけること
→2.1.運用期間外は電源をOFFできること
→送電に関する箇所のため
2.2.外部給電によってもCanSatを動作させることができること
→蓄電機能に関する枝のため
Goal:G_7
送電機能は要求を満たすことを保証する
Strategy:S_7
要求毎に保証する
Goal:G_22
送電機能は要求3.を満たすことを保証する
Context:C_21
3.面積・重量の制限を満たすこと
参考資料：電源部設計書
Goal:G_23
送電機能は要求4.を満たすことを保証する
Context:C_22
4.コストが5000円を切ること
参考資料：UNISON CanSat会計管理シート.xls
Goal:G_24
送電機能は要求5.を満たすことを保証する
Context:C_23
5.運用中の振動によって機能が損失しないこと
参考資料：電源部設計書
Evidence:E_13
質量は1gである
Monitor:M_21
なし
Evidence:E_23
経費は50円である
Monitor:M_22
UNISON CanSat
会計管理シート.xls
Evidence:E_24
計装は振動で切断しないために，
テープで固定する
参照：振動試験の報告書
(参照：設計書5,3,3.1)
Monitor:M_23
なし

機能ごとに要求を満たしていることを確認する。今回は簡易化のために、各機能に対して 1 階層分のみしか記述していないが、必要があればさらにゴールの分解を行う。

今回はトップゴールで「他の系からの要求を満たすこと」と設定しているために、要求を満たしていることを確認するために、最下層で要求ごとにゴールを分解し、エビデンスで要求を満たしていることを保証した。

発展課題：添付の仕様書通り製造したところ、実験中に不具合が生じた。書いた D-Case を使って、指摘せよ。

※間違い箇所[抜粋]

3. 振動試験結果

3.1.振動によって部品が外れないことを確認した

試験環境：常温下、電源は OFF 状態

試験条件：給電はしない状態で振動試験を実施した

解答：

電源は OFF 状態としていたが、電源は ON 状態とし、各機器が動作しているかどうかをモニタすべきであった。

――

今回は要求から以下のように求められている。[抜粋]

2. 電源部は運用期間中に搭載機器へ電力を供給し続けること

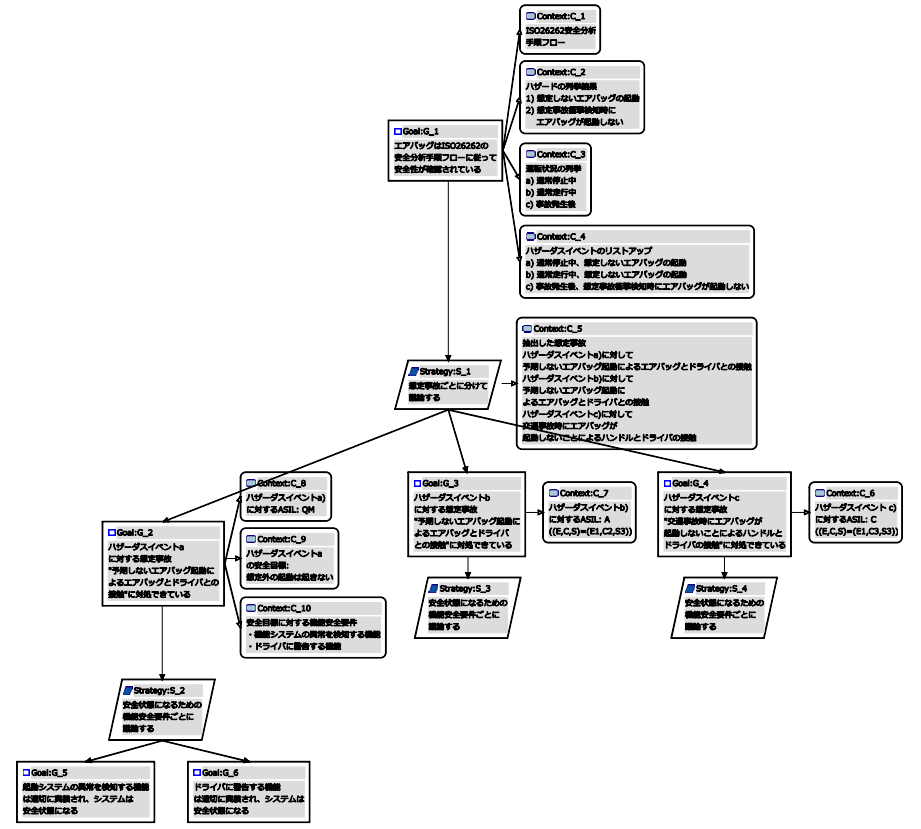
(ア)運用期間中とはロケットに搭載してから打上げ、CanSat を放出してから回収するまでの期間(90 分)を示す。常時加速度を計測する。

今回振動試験は電源を OFF 状態で部品が外れないことのみを確認したが、今回は要求から、“*電源部は運用期間中に搭載機器へ電力を供給し続けること*”が求められているため、振動試験中も電源を投入させた状態で給電が瞬断しないことを確認すべきであった。設計者間で意思疎通を取っていれば防ぐことができた初歩的な不具合であったが、1.いつもは電源が OFF 状態であるという思い込み　2.設計者と試

験実施者が異なる人物であり試験実施条件についてよくよく話し合うことをしなかった、等の理由から不具合を含んだ状態で実験をするに至った。

【問題７】

基本的な構造のみ示す。実際の規格を対象にした D-Case は今後の重要な研究課題である。



上図では、ハザーダスイベント a にのみ、システムが設定された安全状態になるための機能安全要件が適切に満たされているかまで D-Case を作っている。

索引

た

は

ま

ら

実践 D-Case
ディペンダビリティケースを活用しよう！

2013 年　3 月　25 日　発行

著　者　　松野裕、山本修一郎

発　行　所株式会社アセットマネジメント

印刷・製本株式会社アセットマネジメント
〒461-0018　名古屋市東区主税町 4-85
TEL　052－856－6645　FAX　052－856－6646
URL　http://odp.daitec.co.jp/
On Demand Printing & Publishing

ISBN 978-4-86293-091-0