一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
KBSE2012-61 (2013-01)

ユースケース分析に基づくディペンダビリティケース作成法の提案

松野 裕 山本 修一郎

名古屋大学 情報連携統括本部 情報戦略室
〒464-8601 名古屋市千種区不老町

E-mail: syamamoto@ acm.org

あらまし ディペンダビリティケースがシステムの安全性や説明責任を保証する方法として注目されている．しかし，これまでのディペンダビリティケースではユースケース分析を効果的に扱う方法が明確ではなかった．このため，本報告では，ユースケースを構成する記述項目とディペンダビリティケースの構成要素との対応関係を明確化することにより，ディペンダビリティケースを作成する手法を提案する．

キーワード ディペンダビリティケース，ユースケース，UML，アシュアランスケース，リスク分析，機能要求，非機能要求，システム開発文書

A Consideration on Introducing Responsibility Attributes to Dependability Case

Shuichiro Yamamoto and Yutaka Matsuno

Nagoya University, Strategy Office, Information and Communications Headquarters
Furo-cho. Chikusa-ku, Nagoya 464-8601 Japan
E-mail: syamamoto@ acm.org

Abstract *Although dependability case is attracted to assure system safety and availability, methods and guidelines how to describe dependability cases based on use cases are not sufficiently clear. In this paper, problems and issues to describe use case based dependability cases are clarified. Then a method to describe dependability cases based on use cases is proposed and explained with an example.*

Keyword Dependability case, Use case, UML, Assurance Case, Risk analysis, Functional Requirements, Non Functional Requirements, System development document

1 はじめに

システムの安全性を確認するために，安全性ケース(Safety case)，アシュアランスケース（Assurance case, 保証ケース）やディペンダビリティケース（Dependability case）が注目されている[1][2][3][4][5][6][7]．このため GSN(Goal Structuring Notation)を用いてこれらを記述する方法が提案されている[1][2].

筆者らが参加しているDEOSプロジェクト[8][9][10]の一環としてディペンダビリティケースの作成を支援するために D-Case エディタが開発されている[11]. D-Caseエディタでは，GSNに基づいてディペンダビリティケースを記述できる．

DEOS プロジェクトでは，システムのディペンダビリティを D-Case に基づいて確認することが目標の一つになっている．システムのディペンダビリティを D-Case に基づいて確認するためには，システム開発・運用の構造を反映したディペンダビリティケースが必要となる．たとえば，UML を用いたシステム開発文書に対して，ディペンダビリティケースを作成する必要がある．しかし，これまでのディペンダビリティケースでは，UML を前提としたディペンダビリティケースの作成手法が明確ではないという問題があった．

このため，本稿では，UML によるシステム開発文書のうち，とくにユースケース図に着目して，具体的にディペンダビリティケースを作成する方法を提案する

とともに，適用上の課題について考察する．

なお本稿では，安全性ケースやアシュアランスケース，ディペンダビリティケースを総称してディペンダビリティケースという用語を用いる．

以下では，まず第2節で本研究の背景を示す．第3節でユースケースに基づいて，システムのディペンダビリティケースを作成する上での課題を明らかにするとともに対処策を検討する．第4節では，ユースケースに基づくディペンダビリティケースを作成するための基本的な考え方を提案する．第5節で，提案手法に対する有効性と適用性について考察する．最後に第6節で，まとめと今後の課題を明らかにする．

2 研究の背景

2.1 研究動向

重要システムの実行中に優先順位の高い要求を満足することを確認するために，ディペンダビリティケースが必要とされている[7].

ディペンダビリティケースでは，主張（Claim），説明（Strategy, 戦略），前提（Context, コンテクスト），証拠（Evidence, 証跡，ソリューション）によって，システムのディペンダビリティに関する議論を構造化して確認することができる．なお，本稿で「戦略（Strategy）」に対する訳語として「説明」を用いたのは，説明責任を遂行するために，主張の階層関係の理由をStrategyが「説明」しているからである．

ディペンダビリティケースの関連研究として，安全性ケースやアシュアランスケースについて以下のような手法が研究開発されている．

GSNを作成するために，①ゴールを識別する，②ゴールを記述するための基礎を定義する，③ゴールを満足させるための戦略を識別する，④戦略を記述するための基礎を定義する，⑤戦略を吟味する，⑥基本的な証跡を識別するという6段階の手法をKellyが提案している[1][2].

システム障害に至る可能性のある潜在的なソフトウェアの故障モードを識別し，故障モードについての証跡を提示するという証跡に基づくソフトウェア安全性プロセス（evidence based software safety process）の必要性が指摘されている [3].

安全性ケースを再利用するための安全性ケースパターンや，安全性ケースをモジュール化できるモジュラー安全性ケースが提案されている[4][5].

従来のガイドワードを拡張した逸脱分析を用いてディペンダビリティケースを作成する手法[6][12]，シナリオを用いてソフトウェア設計時にディペンダビリティ要求を満たすような代替案を選択する手法[13][14]が提案されている．

European Organisation for the Safety of Air Navigation制定している安全性ケース開発マニュアル[15]では，安全性ケースのコンテクストを定義することが重要であると指摘している．また安全性ケースをレビュするためのチェックリストを提案している．

複数のシステムから構成されるシステム（System of Systems）の開発過程で，システム分析，ゴール要求抽出，代替設計案の識別，矛盾点の解消からなるディペンダビリティケースを構造化して作成する手法が提案されている[16].

UMLのユースケース図では，アクタ（主体）ごとにユースケースを用いることにより，アクタとシステムとの相互作用を定義することができる．また，コントロールケース[17]を用いて，システムアーキテクチャの非機能要求を分析する手法が提案されている．コントロールケースでは，運用条件，運用リスク，リスク対策，非機能要求分類からなるコントロールケーステンプレートに基づいて，システムアーキテクチャに影響を与える非機能要求を定義することができる．

しかし，コントロールケースは，非機能要求を明確に定義するための手法であり，アクタとシステムとの相互作用についてのディペンダビリティを確認する手法ではない．

筆者らはDEOSプロジェクトの一環として，D-Case作成手法[18][19][20][21]，スーパーコンピュータの運用手順[22][23]ならびに，エンタープライズアーキテクチャ開発手法（TOGAF ADM）[24][25]に対するディペンダビリティケース作成実験を進めている．

またディペンダビリティケースへの責任属性を導入する手法を提案している[26]．さらに，ディペンダビリティケース作成知識を体系化するための調査を進めている[27][28][29].

筆者らは，主体間でディペンダビリティゴールが依存する関係を表現するために，d*フレームワークを提案している[10][30].

また概念文書，設計書，運用手順書，準備ハザードリストに基づいて安全性ケースを作成する手法が提案されている[31]．またアシュアランスケースの研修コースも提供されている[32][33][34].これらの手法では，多様な適用分野や開発工程を対象としてディペンダビリティケースの作成法が個別的に提案されている．

しかし，ユースケースで分析されたシステムがディペンダブルであることをディペンダビリティケースに基づいて確認するためには，ユースケースの記述要素をディペンダビリティケースに対応づける具体的な手順を明確にする必要がある．

この点で，現状のディペンダビリティケース作成手法は，ユースケースを適用したシステムに対して，手

順が明確になっていないという問題がある．

2.2 研究の位置付け

上述したことから，本研究では，ユースケース図を用いて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用いて，ディペンダビリティを確認する手法を具体化することとした．この理由は、上述したように，従来のディペンダビリティケース手法がユースケースを対象として具体化されていないためである．

システム開発・運用のディペンダビリティを確認するためには，図1に示すように，システム開発・運用の全工程に対して，ディペンダビリティケースを作成する必要がある．本稿が対象とするユースケース図に対するディペンダビリティケース作成手法は開発D-Caseにおける要件定義 D-Caseの作成手法の一つである．

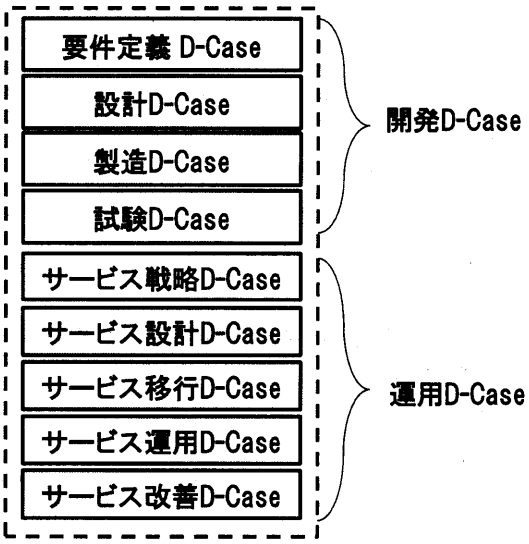

図1　システム開発・運用 D-Caseの構成

3　ディペンダビリティケース作成上の課題

本節では，ユースケースに基づいて，システムのディペンダビリティケースを作成する上での課題を明らかにする．

ディペンダビリティケースでユースケースのディペンダビリティを確認する場合，ユースケースの記述要素について，次の基本的な課題を解決する必要がある．

① アクタの扱い
② アクタとシステム間のメッセージの扱い
③ 契機の扱い
④ 事前条件，事後条件の扱い
⑤ メッセージ系列の扱い
⑥ 基本経路，代替経路，例外経路の扱い
⑦ パッケージの扱い

以下ではこれらの課題について検討する．

3.1　アクタの扱い

ディペンダビリティケースではアクタに対するノードがない．このため，アクタを直接，ディパンダビリティケースで表現できないという問題がある．しかし，前提，主張，説明，証拠に対するノードでアクタ名を参照することができる．

したがって，ユースケースのアクタをディペンダビリティケースのノードでどのように参照するかを決める必要がある．たとえば，前提では，アクタに対して定義された役割を参照できる．主張では，アクタがシステムとの相互作用について責任を果たすことを明記できる．説明では，システムが対象とするアクタの範囲に着目して主張を分解できる．証拠では，アクタの活動についての計画や結果報告を参照できる．

このようにして，ディペンダビリティケースでアクタに関するディペンダビリティを確認できる．

3.2 メッセージの扱い

ユースケースでは，次のような形式でメッセージを記述する．

[形式] メッセージ
<アクタ>が<対象>を<動作>する

ディペンダビリティケースでは，このようなメッセージ記述に対して，システムがディペンダブルであることを説明する必要がある．

このためには，アクタがディペンダブルであること，対象がディペンダブルであること，システム動作がディペンダブルであることを説明する必要がある．この場合，以下のようにしてディペンダビリティケースの断片を構成できる．

主張ノード「<アクタ>が<対象>を<動作>することがディペンダブルである」

説明ノード「メッセージの構成要素について説明する」

下位の主張ノード
「<アクタ>がディペンダブルである」
「<対象>がディペンダブルである」
「<動作>がディペンダブルである」

これらの下位主張を説明するためには，<アクタ>，<対象>，<動作>ごとに，どのようなリスクがあって，リスク緩和のために，どのような対策が，証拠として用意されているかを明らかにする必要がある．このとき，リスク分析内容は，たとえば以下のようにして前提ノードで提示できる．

リスク分析に対する前提ノード
「<アクタ>リスク一覧」

「<対象>リスク一覧」
「<動作>リスク一覧」

3.3 契機の扱い

ディペンダビリティケースでは，契機を記述するためのノードはない．しかし，ユースケースの契機が，ディペンダブルであることを説明することはできる．たとえば，契機がディペンダブルでないリスクを前提ノードで分析しておき，証拠ノードで契機リスク対策を明記することができる．これによって，ユースケースの契機がディペンダブルであることを説明できる．

3.4 事前条件と事後条件の扱い

ユースケースの事前条件をディペンダビリティケースでは，前提ノードで明記できる．事後条件は，ユースケースが遂行されたときに成立すべきことであるから，主張ノードで述べるとともに，証拠ノードで主張内容としての事後条件が成立している証拠を提示する必要がある．

3.5 メッセージ系列の扱い

ユースケースには，複数のメッセージ系列が含まれる可能性がある．この場合，ディペンダビリティケースでは，説明ノードによって，すべてのメッセージ系列がディペンダブルであることを説明できる．

たとえば，次のような説明ノードを作成できる．

「すべてのメッセージ系列について説明する」

この説明ノードの下位にはメッセージ系列ごとに，以下のような主張を作成して説明することができる．

「第 n 番目のメッセージ系列はディペンダブルである」

3.6 基本経路，代替経路，例外経路の扱い

ユースケースの経路についても，メッセージ系列と同様にして，説明ノードを用いて，ディペンダビリティケースを分解できる．

すなわち，次のような説明ノードを作成できる．

「すべての経路について説明する」

この説明ノードの下位にはメッセージ系列ごとに，以下のような３種の主張を作成して説明できる．

「基本経路はディペンダブルである」
「代替経路はディペンダブルである」
「例外経路はディペンダブルである」

この場合，例外経路では，基本経路の条件例外を扱うことから，基本経路の説明の中で，リスク緩和対策が例外経路に対応する可能性もある．したがって，例外経路の条件を分析することにより，例外経路の動作に対する主張を基本経路に対するディペンダビリティケースの内容に統合する必要があると思われる．

なお，テキストによっては，基本経路を主成功シナリオ，代替経路と例外経路をまとめて，拡張と呼んでいることもあるので注意しておく．しかし，ユースケースの内容としてはいずれもメッセージ系列であることから，これらの呼び方についての差異は本質的ではない．

3.7 パッケージの扱い

複数のユースケースをパッケージとしてまとめることができる．また複数のパッケージによってシステムの機能を定義できる．

したがって，以下のような説明ノードを用いることにより，パッケージに対してディペンダビリティケースを分解できる．

「すべてのパッケージに対して説明する」

このとき，システムに対して定義されたすべてのパッケージごとに，以下の説明ノードを用意する．

「第 k 番目のパッケージに含まれるユースケースについて説明する」

この説明ノードに対して，パッケージに含まれるすべてのユースケースについて，対応するディペンダビリティケースを作成することができる．

4. ユースケースに対するディペンダビリティケース作成法

4.1 基本的な考え方

システムに対するユースケースが，パッケージによってグループ化されている可能性があるので，個別ユースケースについてのディペンダビリティケースに分解する．この手順は次のようになる．

【手順】パッケージ分解
[段階 1]パッケージがあれば，パッケージに基づいて分解することにより，パッケージのディペンダビリティケースを作成する．
[段階 2]パッケージ内に複数のユースケースがあれば，ユースケースごとに分解することにより，ユースケースに対するディペンダビリティケースを作成する．
（手順終わり）

以下では，ユースケースに対するディペンダビリティケースの作成方法を示す．

【手順】ユースケース分解
[段階 1]ユースケースに対する主張を作成する
[段階 2]ユースケースの記述項目ごとに分解する
[段階 3]アクタに対するディペンダビリティケースを作成する
[段階 4]契機に対するディペンダビリティケースを作成する
[段階 5]事前条件に対するディペンダビリティケースを作成する
[段階 6]メッセージに対するディペンダビリティケースをメッセージ分解手順により作成する

[段階 7]事後条件に対するディペンダビリティケースを作成する

（手順終わり）

【手順】メッセージ分解

[段階 1]アクタに対するディペンダビリティケースを作成する

[段階 2]対象に対するディペンダビリティケースを作成する

[段階 3]動作に対するディペンダビリティケースを作成する

（手順終わり）

4.2 具体例

以下では文献[35]で例示されている「物を購入するユースケース」を参考にして作成した「商品を購入するユースケース」に対して,提案した手法を説明する.このユースケース記述の内容を付録に示した.このユースケースは1つであり,パッケージはない.また基本経路と代替経路がある.例外経路はない.

このユースケースに対して提案した手順を段階2まで適用して作成したディペンダビリティケースの例を図2に示す.

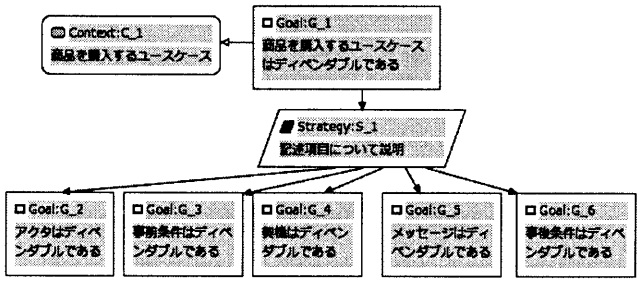

図2ユースケースに基づくディペンダビリティケース（段階2）

次に,アクタに対する主張を分解すると図3のようになる.このユースケースのアクタは「依頼者」「承認者」「決裁者」「購買担当者」「商品販売者」「受取人」である.図3では,依頼者について分解している.他のアクタについての分解は省略している.

依頼者についてのリスクを前提ノードで提示している.ここでは,商品購入を依頼する資格があることが,依頼者アクタがディペンダブルであるために必要であることを確認している.このように,ユースケースだけでは確認できないことをディペンダビリティケースによって確認できることが分かる.

また,メッセージ「依頼者が依頼を作成する」に対するディペンダビリティケースの分解例を図4に示した.ここでは,依頼者アクタ,依頼対象,作成動作について,それぞれがディペンダブルであることを確認するために説明ノードで分解している.なお,G_16がG_7と同じ「依頼者がディペンダブルである」となっていることを注意しておく.このようにして分解された下位の主張に対してディペンダビリティについての確認活動を継続していくことができる.

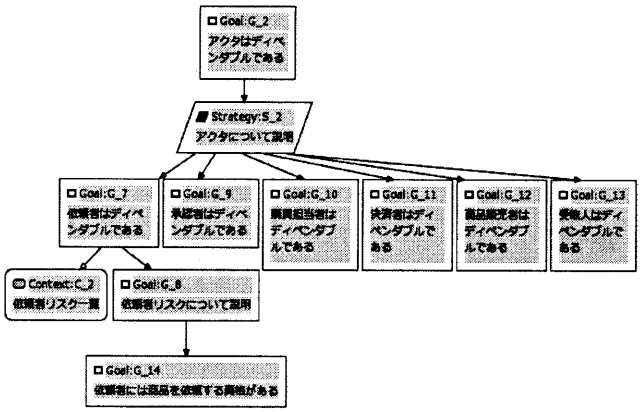

図3アクタ分解の例

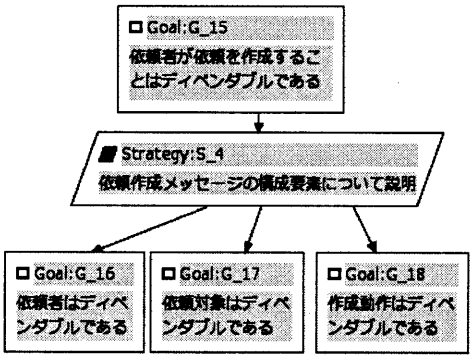

図4メッセージ分解の例

5 考察

5.1 有効性

ディペンダビリティケースを作成する方法を提案して具体例に適用することにより,有効性を確認できした.しかし,提案した手法の有効性を定量的に評価するまでには至っていない.

5.2 適用性

提案したユースケースに基づくディペンダビリティケースの作成方法を実際のシステム事例に適用することにより,適用性を評価していく必要がある.また,UMLのアクティビティ図やシーケンス図を用いたユースケース記述法についてもディペンダビリティケース作成方法を考案して評価する必要がある.

6 おわりに

本稿では,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上での課題を明らかにするとともに,具体的な作成手順を提案した.

今後は本稿で提案したディペンダビリティケース作成法の記述実験を通じて手法として洗練していく.また本稿では代替経路や例外経路の扱いについて詳しく触れる余裕がなかったのでこれらについても報告する予定である.

システムのディペンダビリティを確認・保証するためには，UMLなどの開発手法を前提にしたディペンダビリティケースの作成方法が必要になる．これらについて引き続き研究を進めるとともに，一連の研究成果を踏まえて，ディペンダビリティケース作成ガイドラインとしてまとめる予定である．

謝辞

本研究は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）の支援を受けたものである[8][9].

参考文献

[1] Kelly, T. P, A Six-Step Method for the Development of Goal Structures, York Software Engineering, 1997
[2] T. Kelly. “Arguing Safety, a Systematic Approach to Managing Safety Cases”. PhD Thesis, Department of Computer Science, University of York, 1998
[3] J. A. McDermid. Software safety: where's the evidence? In SCS '01: Proceedings of the Sixth Australian workshop on Safety critical systems and software, pages 1-6, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.
[4] I. Bate, T. Kelly, Architectural considerations in the certification of modular systems, Reliability Engineering and System Safety 81, pp.303–324,2003
[5] Tim Kelly and Rob Weaver, The Goal Structuring Notation – A Safety Argument Notation, Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, July 2004
[6] Despotou G., Kelly T., Extending the Concept of Safety Cases to Address Dependability. In proceedings of the 22nd International System Safety Conference (ISSC), Providence, RI USA, proceedings by System Safety Society 2004.
[7] Jackson, D. et al, Software for dependable systems–sufficient evidence?, NATIONAL RESEARCH COUNCIL, 2008
[8] DEOSプロジェクト, http://www.crest-os.jst.go.jp
[9]DEOSプロジェクト, 2011 科学技術振興機構 White Paper DEOS-FY2011-WP-03J, *www.dependable-os.net/ja/topics/file/**White_Paper**_V3.0J.pdf*
[10] Mario Tokoro eds., Open Systems Dependability, Dependability Engineering for Ever-Changing Systems, CRC Press, 2012
[11] D-Caseエディタ, http://www.dependable-os.net/tech/D-CaseEditor/
[12] G. Despotou, T. Kelly, EXTENDING SAFETY DEVIATION ANALYSIS TECHNIQUES TO ELICIT FLEXIBLE DEPENDABILITY REQUIREMENTS, In proceedings of the 1st IET (Former IEE) International Conference on System **Safety** Engineering (ICSS), London, UK, June 2006.
[13] G. Despotou, T. Kelly. “Using Scenarios to Identify and Trade-off Dependability Objectives in Design. Proceedings of the 23rd International System Safety Conference (ISSC), 2005
[14] T. Kelly. Using software architecture techniques to support the modular certification of safety-critical systems. In Proceedings of the 11th Australian Workshop on Safety-Related Programmable Systems, August 2005.
[15] European Organisation for the Safety of Air Navigation, Safety Case Development Manual, 2nd ed., EUROCONTROL, Oct. 2006.
[16] G. Despotou, T. Kelly, Design and Development of Dependability Case Architecture during System Development, In proceedings of the 25th International System Safety Conference (ISSC), Baltimore, MD USA. Proceedings by the System Safety Society, 2007.
[17] Joe Zou and Christopher J. Pavlovski: “Modeling Architectural Non Functional Requirements: From Use Case to Control Case”, Proceedings of IEEE International Conference on e - Business Engineering 2006(ICEBE’06) (2006).
[18]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！～，ダイテックホールディング, 2012, ISBN 978-4-86293-079-8
[19] 山本修一郎，松野裕，ディペンダビリティケース作成法に関する一考察，KBSE研究会，IEICE-112, vol. IEICE-SS-164, No. IEICE-KBSE-165, pp.61-66, 2012
[20] 松野 裕，高井利憲，ヴァイセ パテゥー，山本修一郎，アシュアランスケース構築法の提案，KBSE研究会，2012
[21] Vaise Patu, Yutaka Matsuno, Shuichiro Yamamoto, Application of D-Case to the usage flow diagram scenario of the Distributed E-Learning System called KISSEL in Asian Pacific Universities, KBSE研究会，2012
[22] 高間翔太，松野裕，山本修一郎，スーパーコンピュータ運用手順に対するディペンダビリティの確認手法の提案，信学技報，vol. 112, no. 165, KBSE2012-18, pp. 37-42 2012
[23] 高間翔太，松野裕，山本修一郎，ディペンダビリティ・コンテクストの推定手法の提案，KBSE研究会，2012
[24] 徳野達也，松野裕，山本修一郎，エンタープライズアーキテクチャ開発プロセスに対するディペンダビリティケース作成法の提案，信学技報，vol. 112, no. 165, KBSE2012-36, pp. 145-150 2012
[25] 徳野達也，松野裕，山本修一郎，TOGAF NEXTに対するADMプロセステンプレートの提案，KBSE研究会，2012
[26] 山本修一郎，松野裕，ディペンダビリティケースへの責任属性導入法の検討，KBSE研究会，2012
[27] 松野裕，ヴァイセ バトゥ，山本修一郎，アシュアランスケースへの構造化文書の適用に関する調査，信学技報，vol. 112, no. 165, KBSE2012-20, pp. 49-54, 2012
[28] Vaise Patu, Yutaka Matsuno, Shuichiro Yamamoto, Research framework for dependability science based on assurance cases, IEICE Tech. Rep., vol. 112, no. 165, KBSE2012-21, pp. 55-59, July 2012
[29] 猿渡卓也，松野裕，星野隆，山本修一郎，Modular GSNの定式化，KBSE研究会，IEICE-112, vol. IEICE-SS-164, No. IEICE-KBSE-165, pp. 151-156,2012
[30] Shuichiro Yamamoto, Yutaka Matsuno, d* framework: Inter-Dependency Model for Dependability, DSN 2012
[31] Ewen Denney and Ganesh Pai, Ibrahim Habli, Perspectives on Software Safety Case Development for Unmanned Aircraft, DSN2012
[32] Adelard,www.adelard.com/asce/index.html
[33] AAMI, Safety Assurance Cases for Medical Devices, http://www.aami.org/meetings/courses/safety.html
[34] Dependability computing, assurance case course, http://www.dependablecomputing.com/courses.html
[35] アリスター・コーバーン，ユースケース実践ガイド，効果的なユースケースの書き方，ウルシステムズ監訳，翔泳社,2001

付録．商品を購入するユースケース（[35]を参照して改訂）

[事前条件]なし
[契機]依頼者が商品購入を決定する
[事後条件]
承認者が依頼を承認している．決裁者が依頼を確認している．購買担当者が注文書を作成している．受取人が受け取りを登録している．商品販売者が受領書を受け取っている．依頼者が届いた商品を記録している．
[基本経路]
依頼者が依頼を作成して，承認者に送付する．
承認者が予算を確認し，商品価格を確認し，依頼を承認して購買担当者に提出する．
購買担当者が在庫を確認し，商品販売者を選定する．
決裁者が承認印を確認する．
購買担当者が注文書を作成し，商品販売者に送付する．
商品販売者が商品を受取人に引き渡し，受領書を受け取る．受取人が受け取りを登録し，商品を依頼者に送付する．依頼者が依頼した商品が届いたことを記録する．
[代替経路]
商品を受け取る前であればいつでも，依頼者が依頼内容の変更や依頼のキャンセルができる．キャンセルすると，その依頼は処理できなくなる．価格が下がった場合，依頼が承認者まで戻される．