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
KBSE2013-12 (2013-05)

議論分解パターンに基づくディペンダビリティケースの作成実験

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．このため，ディペンダビリティケースでの議論分解を効果的に扱うために議論分解パターンが提案されている．しかし，具体的な有効性については明確ではなかった．本報告では，議論分解パターンを用いたディペンダビリティケースの作成実験を実施した結果について報告する．

キーワード ディペンダビリティケース，アシュアランスケース，議論分解パターン，実験評価

An Experiment on Dependability Case Development based on Argumentation Patterns

Shuichiro Yamamoto

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

Abstract *Dependability case is attracted to assure system safety and accountability. Patterns of argumentation to decompose dependability cases are proposed, but the details of patterns were not sufficiently clear. In this paper, a result of experimental evaluation of developing dependability cases based on argumentation patterns is described.*

Keyword Dependability case, Assurance Case, Argumentation decomposition pattern, Experimental evaluation

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，D-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].

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 D-Case 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，D-Case 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，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した[8]．このように，ディペンダビリティを確認する対象の構造に基づいて D-Case を作成する場合，典型的な議論分解パターンが共通に出現する．また，筆者らが実践している D-Case の研修会では，D-Case による議論分解についての参加者からの質問には共通事項があることが分かってきた．このため，本稿では，既存の議論分解パターンに加えて，筆者らが識別した議論分解パターンを具体化するとともに有効性について考察する．

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 節で本研究の背景とディペンダビリティケースの議論分解パターンの例を説明する．第 3 節では，実験対象システムの概要について述べる．

第4節で実験結果を明らかにする．第5節で実験結果に基づいて議論分解パターンの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について述べる．

2 研究の背景

ディペンダビリティケースの研究動向については既存の報告を参照されたい[8][9][10][11][12][13][14][15][16][17][18][19][20].

ディペンダビリティケースの議論分解については Bloomfield の分類[21]がある．この内容を表 1 に示した．しかし，簡単な説明があるだけで具体的な内容については明確にされていない．このため，筆者らによる研修教材では，議論分解パターンを D－Case エディタ[22]により簡単に例示した[23]．しかし，D-case の記述例を示しただけで説明していなかった．

表 1　議論分解パターンの例

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

なお Bloomfield[21]は，ディペンダビリティケースの動向をサーベイしているので参考になる．しかし，ディペンダビリティケースの議論分解パターンが具体化されていなかったため，筆者らは分解パターンを説明するための記述項目をソフトウェア・アーキテクチャを記述するためのパターン体系[22]に基づいて，体系的に整理する方法を提案することにより議論分解パターンを説明した[23][24][25].

アーキテクチャ分解パターンの記述例を以下に示す．

【名称】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保証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 1 に示すように，分解することができる．

図 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．

【特性】

（利点）システムのアーキテクチャに従って，保証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保証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．

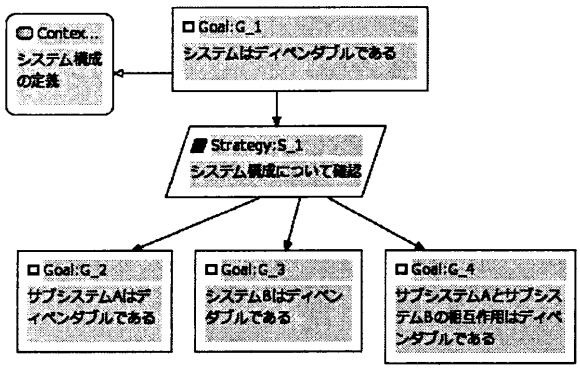

図 1　アーキテクチャ分解の例

また，筆者らは議論分解パターンの有効性を明らかにするための記述実験が必要であることを指摘した[24]．本稿では，議論分解パターンの選択と適用上の課題を明らかにするために実施した記述実験の結果について述べる．

3　記述実験

本節では，ディペンダビリティケースの議論分解パターンの有効性を確認するために実施した記述実験について述べる．

3.1 実験対象

実験対象システムは組込みシステムである．被験者は組込みシステム開発で 20 年以上の経験を持つ技術者である．この被験者は 4 時間のディペンダビリティ

ケースの集合研修に参加した．研修教材の内容は以下のとおりである．

(1)ディペンダビリティケースの概要　10 頁
(2)ディペンダビリティケース開発法　26 頁
(3)ディペンダビリティケース演習　15 頁
(4)議論分解パターン　15 頁

被験者はこの研修を受講した後に，以下に示す組込みシステムに対してディペンダビリティケースを作成した．

3.2 実験対象システム

実験対象は LAN 機器管理システム(LAN Device Management System, LDMS)である．LDMS の構成要素にはマネージャ，センサー,LAN 機器がある．LAN 機器には正規の機器と不正な機器がある．これらの構成要素は図 2 に示すようにネットワークと LAN によって接続されている．LDMS の目的は正規 LAN 機器の情報を管理することにより，不正 LAN 機器が LAN に接続されていれば検出することである．LAN 機器はセンサーによって監視される．センサーは LDMS マネージャによって制御されている．LDMS マネージャは約 2000 個のセンサーを制御する．各センサーは約 1000 個の LAN 機器を監視できる．センサーは国内外の拠点に設置されネットワークで接続されている．

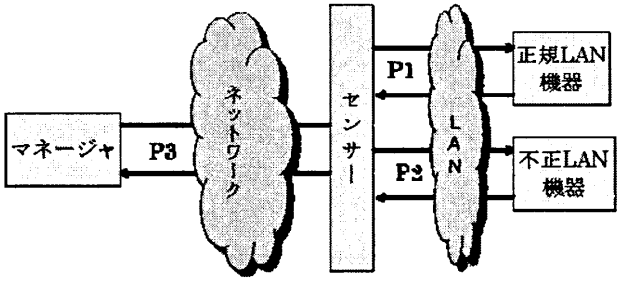

図 2 LDMS の概要

図 2 の P1, P2, P3 はネットワークの接続点を示している．接続点 P1 と P2 は LAN 機器をセンサーによって監視するために使用する．P3 はセンサーを監視するために LDMS マネージャによって用いられている．

LDMS 構成要素の主要機能を説明すると以下のようになる．

（1）マネージャ

・センサーから収集した情報を編集して表示する
・センサーとの通信情報を監視する

（2）センサー

・不正 LAN 機器を検出すると接続を遮断する
・予め設定された時間間隔で LAN 機器の CPU 使用率，メモリ使用率などを二次記憶装置に記録する
・CPU のクロックタイミングを切り替える
・例外時にソフトウェアにより自動的に再開する
・マネージャとの通信状態を監視する

（3）マネージャとセンサーの相互作用

・マネージャからセンサーに対して更新された遮断テーブルを配布する
・センサーからマネージャに対して検出した未知の LAN 機器を通知する
・マネージャによってセンサーの状況を監視する

4 実験結果

4.1 作成工数

被験者が LDMS に対する D-Case を 1.5 人月で作成した．表 2 に作業の内訳を示した．同表で示したように 5 個の作業項目があった．作業項目は，仕様理解,議論分解パターンの選択,アーキテクチャ分解，リスク分析，D-Case 作成であった．D-Case エディタ[27]で D-Case を作成した．

表 2 作業工数の内訳

作業分類	人時	比率
仕様理解	5	2%
議論分解パターン選択	30	14%
アーキテクチャ分解	10	5%
リスク分析	62	28%
D-Case 作成	110	51%
合計	217	100%

仕様理解,議論分解パターンの選択,アーキテクチャ分解，リスク分析, D-Case 作成の比率は，それぞれ 2%, 14%, 5%, 28%, 51%であった.全工数の 19%(=14%+5%)がアーキテクチャ分解パターンの適用にかかったことが分かる．議論分解パターンの理解工数が適用工数の 3 倍となった．この理由は被験者がディペンダビリティケースならびに議論分解パターンの知識がなかったためである．議論分解パターンの理解工数は被験者の持つディペンダビリティケースと議論分解パターンの知識に依存する．

仕様理解工数の割合が 2%と少ない理由は，被験者がこの分野の経験者であったためである．仕様理解工数は被験者が持つ分野知識に依存する．

4.2 D-case の記述例

図 3 に LDMS に対して議論分解パターンを適用して作成したディペンダビリティケース（D-Case）の例を

示す．同図では最上位のディペンダビリティケースを示した．また表3では作成したディペンダビリティケースのノード数の内訳をLDMSの構成要素とその相互作用ごとに示した．全体で1098個のノードがあった．主張と証拠の割合はそれぞれ 49.1%(=539/1098) と 33.5%(=368/1098)であった．前提ノード数は 10 個しかなかったが,前提ノードに対するリスクの個数は 144 であった．

表3 ディペンダビリティケースのノード数の内訳

構成要素		前提	主張	戦略	証拠
Sensor	Power unit	1(16)	83	30	71
	Main board	1(17)	60	21	42
	HW case	1(6)	20	7	13
	HW interaction	1(16)	54	18	43
	Software	1(25)	124	41	60
	HW- SW Interaction	1(11)	35	11	27
Manager	HW	1(4)	13	4	10
	SW	1(18)	56	18	38
	HW- SW Interaction	1(8)	24	8	16
Interaction between sensors and manager		1(23)	70	23	48
Total		10(144)	539	181	368

(注) 括弧内の数字はリスク数を示す．

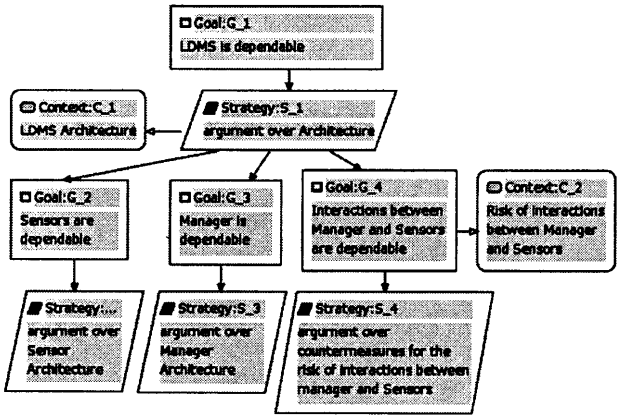

図3 LDMSへのアーキテクチャ分解パターンの適用例

4.3 被験者の意見

LDMSにディペンダビリティケースを適用した結果に対して被験者から以下の意見が得られた．

過去に摘出したすべての同様の欠陥を網羅できたことから，アーキテクチャ分解パターンの適用が適切であった．

アーキテクチャ分解に基づいて系統的にディペンダビリティケースを作成することにより，製品とシステム構成要素の欠陥を分析できた．これによって開発者による欠陥原因の分析を支援できる．

アーキテクチャ分解パターンによってリスク分析とその対策の関係を明確化できる．アーキテクチャ分解パターンの適用によって，系統的なリスク分析を遂行できるから，従来手法よりもリスク抽出漏れを削減できる．

また被験者によれば，対象システムに対する十分な知識がアーキテクチャ分解パターンを適用する上でも必要である．

5 考察

5.1 議論分解パターンの有効性

被験者の意見からアーキテクチャ分解パターンがリスク分析を遂行する上で有効であったことが明確になった．一方で，議論分解パターンを理解して，その中からアーキテクチャ分解パターンを選択するまでに全体の約14%の工数が必要となった．しかし，アーキテクチャ分解パターンを選択した後は，アーキテクチャ分解，リスク分析，ディペンダビリティケースを系統的に作成できた．

もし，アーキテクチャ分解パターンを知らなければ，被験者は 1098 ノードのディペンダビリティケースを1.5人月では作成できなかったと考えられる．

5.2 知識と工数の関係

システムのディペンダビリティを保証するためには，対象システムの知識とディペンダビリティケース適用知識が必要である．したがって，これらの知識の有無によって対象システムに対するディペンダビリティケース作成工数は大きく左右される．

今回の適用実験によって，ディペンダビリティケースの未経験者が対象システムの知識があれば，議論分解パターンを学習することによって系統的にディペンダビリティケースを作成できることが明らかになった．

対象システムに対する深い知識がない場合に，議論分解パターンを用いてディペンダビリティを漏れなく確認できるかどうかについては今後明らかにしていく必要がある．

5.3 D-Case エディタ

実験では，1098ノードという大規模なディペンダビリティケースをD-Caseエディタで作成できた．一方で大規模なD-Caseを編集する場合,画面上に表示できるノード数の限界から，編集効率が低下するという問題があった．また印刷する際に，適切な範囲を選択して部分的な印刷を工夫する必要があった．

これらの課題を解決するために，D-Caseエディタを

拡張して,モジュールを扱えるようにする予定である.実用規模のシステムに対するディペンダビリティケースを作成する上では,編集作業を効率化する上でモジュール化機能が必要である.

今回の実験では D-Case 作成に全体の約半分の工数がかかっているが,大規模なディペンダビリティケースを効率よく編集できれば,この工数を削減できる可能性がある.

6 おわりに

本稿では,ディペンダビリティケースを作成するための議論分解パターンの有効性を確認するために実施した記述実験結果を報告した.

実験では LAN 機器管理システムを対象としてアーキテクチャ分解パターンを適用することによって 1098 ノードの D-Case を約 1.5 人月で作成し,144 個のリスクに対して十分な対策ができている証拠を確認した.

今後は本稿で対象としたアーキテクチャ分解パターンだけではなく他の議論分解パターンを用いた D-Case の記述実験を通じて有効性を確認していく予定である.

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[29][30][31].

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] 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

[9] 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

[10] 松野裕, 山本修一郎,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, KBSE研究会, IEICE-112, vol. IEICE-KBSE-419, KBSE2012-61, pp.19-24

[11] 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

[12] 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

[13] 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, 信学技報, vol. 112, no. 314, KBSE2012-42, pp. 25-30, 2012

[14] 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

[15] 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, 信学技報, vol. 112, no. 314, KBSE2012-55, pp. 103-108, 2012

[16] 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, 信学技報, vol. 112, no. 314, KBSE2012-52, pp. 85-90, 2012

[17] 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

[18] 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
[19] 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, 信学技報 vol.112, No.165, pp.151-156, 2012
[20] Shuichiro Yamamoto, Yutaka Matsuno, d* framework: Inter-Dependency Model for Dependability, DSN 2012
[21] Robin Bloomfield and Peter Bishop, Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective, 2010
[22] 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
[23] 山本修一郎, 松野裕, ディペンダビリティケース分解パターンについての考察, KBSE 研究会, 信学技報, vol. 112, no. 496, KBSE2012-80, pp. 67-72, 2013
[24] 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
[25] 松野裕, 高井利憲, 山本修一郎, 実践 D-Case,～ディペンダビリティ・ケースを活用しよう！～, ダイテックホールディング, 2013, ISBN 978-4-86293-091-0
[26] D-Case 実証評価研究会, http://dcase.jp/
[27] D-Case エディタ, http://www.dependable-os.net/tech/D-CaseEditor/
[28] 松野裕, 山本修一郎, アシュアランスケースツールへのプログラミング言語技術の適用, KBSE 研究会, 信学技報, vol. 112, no. 496, KBSE2012-81, pp. 73-78, 2013
[29] DEOSプロジェクト, http://www.crest-os.jst.go.jp
[30]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*
[31] Mario Tokoro eds., Open Systems Dependability, Dependability Engineering for Ever-Changing Systems, CRC Press, 201