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
KBSE2012-82(2013-3)

アーキテクチャに基づく検証ケースの提案

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　システムの妥当性を形式手法だけで検証することは実践的ではない．このため形式手法と，テストやレビュなどとの組み合わせを明確化するための手法が求められている．

本稿では，アーキテクチャに基づいて形式手法による検証範囲を明確化する検証ケースを提案する．

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，アーキテクチャ，検証ケース，形式手法

A proposal on architecture based verification case

Shuichiro Yamamoto

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

Abstract *Although formal method is attracted to verify system correctness, it is not practical to verify every property of system only by formal methods. Therefore the integration with formal methods and assurance case methods is expected. In this paper, architecture based quality case method is surveyed and then the verification case method is proposed to integrate architectural argument decomposition and formal evidences.*

Keyword Dependability Case, Assurance Case, Architecture, Verification Case, Formal Method

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，筆者らはユースケース図に基づくD-Case作成法を提案している．また，システムのアーキテクチャは非機能要求を実現する必要がある[15]から，システムのディペンダビリティをアーキテクチャが保証することを確認する必要がある．要求に対するアーキテクチャの妥当性を議論することは，アーキテクチャに影響を与える要求と，影響しない要求を区別することでもある．また，要求に基づいてアーキテクチャを評価する手法が提案されている．筆者らによるディペンダビリティケースの研究については既存の報告を参照されたい[8][9][10][11][13][14].

本稿では，アーキテクチャに基づいてシステムの妥当性を検証するために，D-Caseに基づく検証ケースを提案する．

以下では，まず第2節で本研究の背景となるアーキテクチャ有意要求ならびに品質ケースについて述べる．第3節では，システム安全検証について述べる．第4節で提案手法とその具体例について述べる．最後に第5節で，まとめと今後の課題を明らかにする．

2　アーキテクチャ有意要求の必要性

システム開発が失敗する理由の最初の2つが，要求とアーキテクチャの欠陥である．　アーキテクチャに影響する非機能要求のことをアーキテクチャ有意要求（Architecturally Significant Requirements, ASR）という．ASRを正しく定義するのは困難である．この理由は，機能

要求はアーキテクチャの要素としてのコンポーネントに割り当てることが容易にできるのに対して，システム全体の振舞いに影響を与える ASR は，特定のコンポーネントに対応しないためである．大規模システム開発では，アーキテクチャが決まらないと，プロジェクトの組織体制を決めることができない．したがって下流工程の設計，実装，統合，テスト，展開にも影響する．

典型的なアーキテクチャ有意要求には以下の要素が含まれる．

(1)指定された品質特性を満足すべき程度としての品質要求

(2)アーキテクチャ制約

(3)アーキテクチャが実現する機能要求

このようなアーキテクチャ有意要求の必要性を，品質モデルの観点から見れば，次のように考えることができる．

システム品質は，システムアーキテクチャ品質によって決定される．システムアーキテクチャ品質は，アーキテクチャ有意要求によって決定される． したがって，アーキテクチャがシステムにとって必要であると同時に，アーキテクチャ有意要求も必要になる．換言すれば，ASR とそれに基づくアーキテクチャ判断が，システム全体の品質，開発計画，開発維持経費，保守性，拡張性，受容性，使用性に影響することになる．したがって，ASR とアーキテクチャの妥当性とともに，それらのリスクを分析することが重要になる．要求リスクとアーキテクチャリスクを分析するためには，要求とアーキテクチャを可視化することが重要になる．これによって，発注者の要求に対する，要求とアーキテクチャの準拠性ならびに，アーキテクチャに対するシステムとコンポーネント要求の準拠性を確認することができる．

【定義】

必要とされる特性に対して，開発生産物が示す程度のことを**品質**という．

品質を測定するためには品質特性を定義するとともに，測定尺度を定義する必要がある．このため，次のような品質モデルが必要になる．

2.1 品質モデル

開発生産物の品質は，品質特性(**Quality Characteristics**)，品質属性(**Quality Attributes**)，品質計測単位(**Quality Measurement Scales**)からなる品質モデルで定義される．品質モデルの構成要素の例を表 1 に示す．

表 1　品質モデルの構成要素例

モデル項目	例
品質特性	可用性，各調整，相互接続性，保守性，性能，移植性信頼性，安全性，セキュリティ，使用性
品質属性	遅延，応答時間，スループット
品質計測単位	ミリ秒，トランザクション数／秒

品質特性には，開発時に測定される開発指向の品質特性と，システム運用時に測定される運用指向の品質特性がある．開発指向品質特性の代表的な例は，バグ数に基づくソフトウェア品質である．運用指向品質特性には，ディペンダビリティ，効率性，相互接続性，持続可能性，使用性，性能，容量，構成可能性などがある．ディペンダビリティは，健全性と防御性（defensibility）がある．防御性は，耐故障性，安全性，セキュリティ，回復性がある．健全性には，可用性，正当性，予測可能性，信頼性などがある．なお，このような品質特性の階層分類の仕方には多様なものがあるので，ここで示したのは QUASAR[16]に基づく分類例である．

2.2 品質要求の記述例

【記述形式】

条件 X の下で,システムが 指定された品質基準 Y に対する閾値 Z に一致するか越えている

この記述形式に従うと，「システムは高信頼で，耐故障性があり，安全でセキュアである必要がある」というような品質要求の記述は，定量的ではないので，よくないことが分かる．これに対して，以下のような記述はよい品質要求の例である．

「通常運用条件の下で，基幹系機能に対して，システムの平均故障時間間隔が少なくとも 5000 時間である必要がある．」

2.3 品質ケース

開発生産物が十分な品質を持つことを示すために，主張(*Claim*)，議論(*Argument*)，証拠(*Evidence*)を用いた一貫性のある説明を**品質ケース(Quality Case)**という．

ここで「ケース」というのは，もともと，法廷用語で，有罪か無罪かを立証するために法廷での弁論のやり取りを Toulmin[17]が定式化した論証手法に由来する．システム安全性やディペンダビリティを確認するための手法として，安全性ケース，ディペンダビリティケース,アシュアランスケースなどが考案されている．安全性ケースやディペンダビリティケースは，それぞれ，安全性やディペンダビリティについてのアシュアランスケースである．アシュアランスケースは保証ケースともいう．

Firesmith は，品質ケースによって品質の評価とシステムが要求品質を持つことの認証を容易化できるとしている．この意味では，アシュアランスケースを適用して品質ケースが開発されているのだから，アシュアランスケースを用いてアーキテクチャ要求品質を保証できるともいえる．

品質ケースのもとになったアシュアランスケースを記述する手法では，直接的に主張を支持する証拠が見つかるまで，確認すべき主張を階層的に展開していく． アシュアランスケースについては文献を参照されたい．筆者らは，現在，ディペンダビリティケースについて研究を進めており，ディペンダビリティケースを作成するための手法などを開発している[11].

2.4 品質ケースの構成要素

保証ケースは，表 2 に示したように，上述したアシュアランスケースと同様に，主張，議論，証拠から構成されている．

◆品質ケースとアシュアランスケースの違い

アシュアランスケースに対する品質ケースの主な拡張点と制約点は次のとおりである[1].

【拡張点】

品質特性と品質属性を直接記述できる.

【制約点】

要求とアーキテクチャに限定している .

図形要素を矩形に限定している.

階層を，主張層，議論層，証拠層からなる3階層に限定している.

以下では品質ケースを記述するための QUASAR 図式 QUality Assessment of System Architectures and their Requirements (QUASAR)では，2 種類の品質ケースとして，要求品質ケースと，アーキテクチャ品質ケースを定義している.

表 2　品質ケースの構成要素

要素	記述	例
主張	開発生産物が十分な品質を保証できることを記述	品質特性と品質属性に基づいて，プロジェクトの公式品質モデルの中で品質を定義する．
議論	主張に対して，査定者の信念を正当化するための，明白で説得力のある適切な論証	判断，考案内容,トレードオフ,分析，シミュレーション結果,前提条件,関連する理由など
証拠	議論を裏付けるための適切で信用できる証拠	公式のプロジェクト図式，モデル，要求仕様，アーキテクチャ文書，要求リポジトリ，分析報告，シミュレーション結果報告，査定人が証明したデモ内容など

3.5 要求品質ケース

要求品質ケースでは，主張として品質特性を記述し，議論としてアーキテクチャ判断やトレードオフ分析，前提などを記述する．証拠には，図式，モデル，文書などを記述する．

要求品質ケースの記述例を図 1 に示した．このように，要求品質ケースは，主張層，議論層，証拠層からなる 3 階層で記述する．また，主張，議論，証拠をすべて矩形で記述している点がアシュアランスケースと異なる点である．これは，後述するアーキテクチャ品質ケースでも同様である． この理由は，UML クラス図の表記法を用いて，品質ケースを書く上での構成要素の表記法を単純化することにしたためであろう．

主張層では，UML のクラス図の集約関係（図 2 のひし形）を用いて，品質特性を構成要素に分解している．主張と議論の関係，議論と証拠の関係は上位の要素が成立することを支援することを示している．

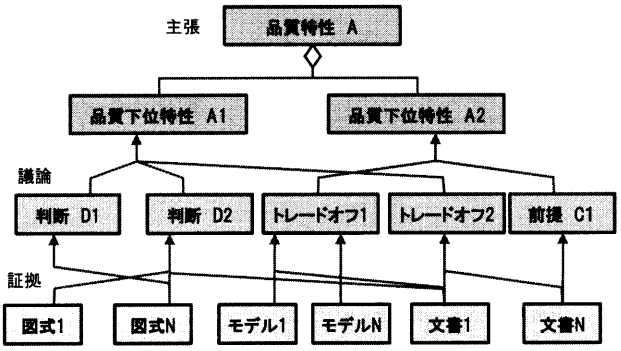


図 1　アーキテクチャ品質ケースの例

なお，図から分かるように，アシュアランスケースと矢印の方向が逆になっていることに注意しよう．また主張内の関係には矢印がない．

3.6 アーキテクチャ品質ケース

アーキテクチャ品質ケースでは，主張層，議論層，証拠層からなる 3 階層で記述する．アーキテクチャ品質ケースの例を図 2 に示した．

最上位の主張として，アーキテクチャが品質特性を満たすことを記述する．たとえば，「アーキテクチャが相互運用性をサポートする」というように，真偽を判定できる命題文で記述する．この際上位にの主張の下位には，そのために成立することが求められる下位の品質特性名を記述する．相互接続性であれば，「プロトコル相互運用性」，「構文相互運用性」，「意味相互運用性」など，システムの相互運用性を構成する要素対象によって，「相互運用性」を修飾した複合名詞を主張名とする．アーキテクチャ品質ケースの議論層では，具体的に採用したアーキテクチャの方式を記述している．たとえば，階層アーキテクチャや SOA などである．これらのアーキテクチャ方式によって，上位の品質要求が正当化できることを関係によって説明している．

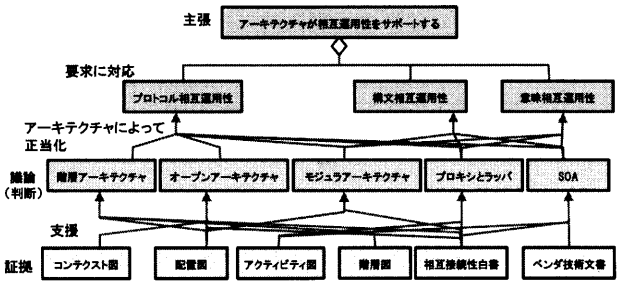


図 2　アーキテクチャ品質ケースの例

アーキテクチャ品質ケースの証拠層では，コンテクスト図や配置図，階層図，技術文書などを証拠として明記する．これらの証拠によって議論で記述した，アーキテクチャ方式が支援されることを関係によって示している．

3 アーキテクチャ指向検証ケース

3.1 システム安全検証とは

システム障害や運用トラブルなどを予防するために，システム構造や運用手順などに基づいて，システムリスクを客観的に掘り起こし，体系的なリスク対策ができていることを確認する必要がある．

システムの安全性を検証するためには，次の 3 条件に基づいて客観的にシステムの安全性を確認する必要がある．

① システムが安全に動作できる前提条件

前提条件が満たされている場合，システムは正常運用できる

この「前提条件」が明確に定義されているかどうかを確認する必要がある

② 前提条件からの逸脱が発生する例外条件

例外条件が発生した場合，適切な例外処理の提供により，システム運用を回復できる．

この「例外条件」が明確に定義されていて，対応する「例外処理」が容易されているかどうかを確認する必要がある．

③ 前提条件とも例外条件とも異なる想定外条件

想定外事象が発生した場合，適切な運用手順の用意により，システムトラブルを回避する必要がある．

経済性や技術の限界から，すべての例外に対応することはできない．したがってシステムには必ず想定外の事象が発生する可能性が残るので「残余リスク」がある．この残余リスクを識別しておくことで，トラブル対応を適切にマネジメントする必要がある．

この 3 条件は，システムを構成するすべての要素に対して適用できますから，システム全体に対して体系的に安全性を検証することができるようになる．また，調達コンポーネントについても同様に検証する必要がある．

形式手法を用いる場合，事前条件と事後条件を形式的に定義する必要がある．したがって，形式手法では上述した条件①の場合に対して，システムの安全性を形式的に検証できる．しかし，想定できない条件については，安全性の正当性を形式手法で検証することはできない．例外条件に対しては，例外対策の事前条件と事後条件が形式的に定義できる場合には例外対策を対象として安全性を形式的に検証できる場合がある．残余リスクに対しては，対応するシステムの機能がないことから形式的に検証することはできない．

3.2 アーキテクチャ指向検証ケース

アーキテクチャ指向検証ケースでは，システムのアーキテクチャに基づいて，形式手法を用いて構成要素ごとにシステムの安全性を検証する手法である．システムのディペンダビリティや安全性を，アーキテクチャに基づいて分解して確認する手法については，筆者らも提案している．形式手法だけでシステムの安全性を完全に検証することは，効率や性能などの理由から現実的には難しい．このため，テストやレビュなどを形式手法と組み合わせる必要がある．そこで，本稿では，D-Case を用いて形式手法で検証する部分と既存手法で確認する部分を客観的な根拠に基づいて説明する手法をアーキテクチャ指向検証ケースと称する．

以下では，この手法を説明する．

まず，アーキテクチャに基づいて分解する．

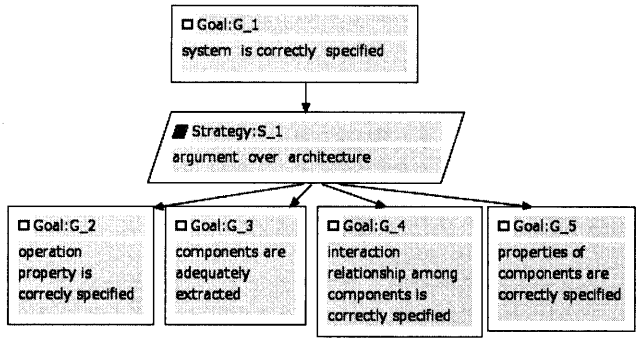


図 2 アーキテクチャ指向検証ケースの記述例

G_2 では，証明可能な性質であることを主張している．G_3 では，システム構成要素が適切に抽出されていることを主張している．G_4 では，コンポーネント間の相互作用が正しく仕様化されていることを主張している．G_5 では，構成要素が満たすべき性質が正しく仕様化されていることを主張している．

G_2,G_3 に対する証拠の例を図 3 に示す．

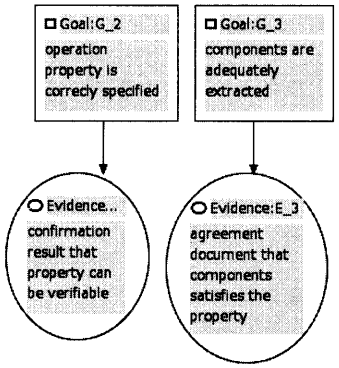


図 3　主張 G_2,G_3 に対する証拠の例

G_4,G_5 に対する議論分解の例を図 4 に示す．G_4 と G_5 がそれぞれ，戦略 S_2,S_3 によって，G_7 と G_6 に分解されている．証拠 E_2,E_1 は，形式手法による証明結果である．

この例では，システム構成要素とその相互作用の安全性が形式手法によって証明できる例である．

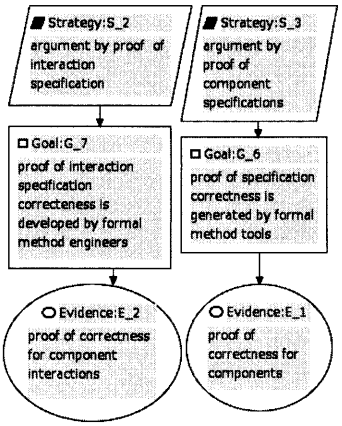


図 4 G_4,G_5 に対する議論例

また，別の例として，付図 1 に，マイコンが止まらないことに対する検証ケースの例を示す．

5　考察

5.1 有効性

ディペンダビリティケースをアーキテクチャに基づいて形式手法と組合せることによって作成するための方法を具体的に例示した．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．今後，この検証ケースを用いた実験を進めることにより，具体的な形式手法による検証事例を明らかにする必要がある．

5.2　適用性

提案した検証ケースをシステム開発ライフサイクルと運用ライフサイクル全体に対して適用できることを確認する必要がある．このような検証ケースの構成例を図 3 に示す．

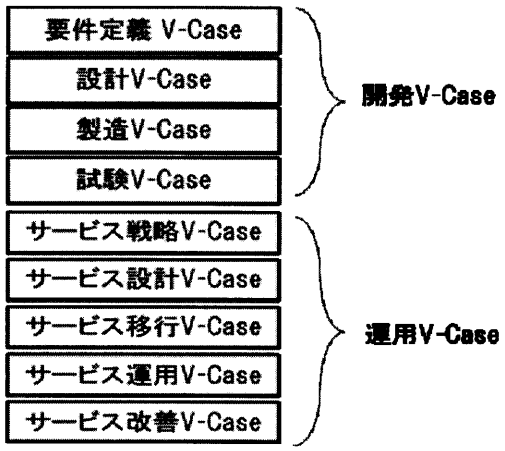


図 3　検証ケースの構成例

6 おわりに

本稿では，ディペンダビリティケースをアーキテクチャと形式手法に基づいて作成するための検証ケースを提案するとともに，具体的な適用例を紹介した．

今後は本稿で提案した検証ケースを用いた D-Case の記述実験を通じて手法として洗練していく予定である．

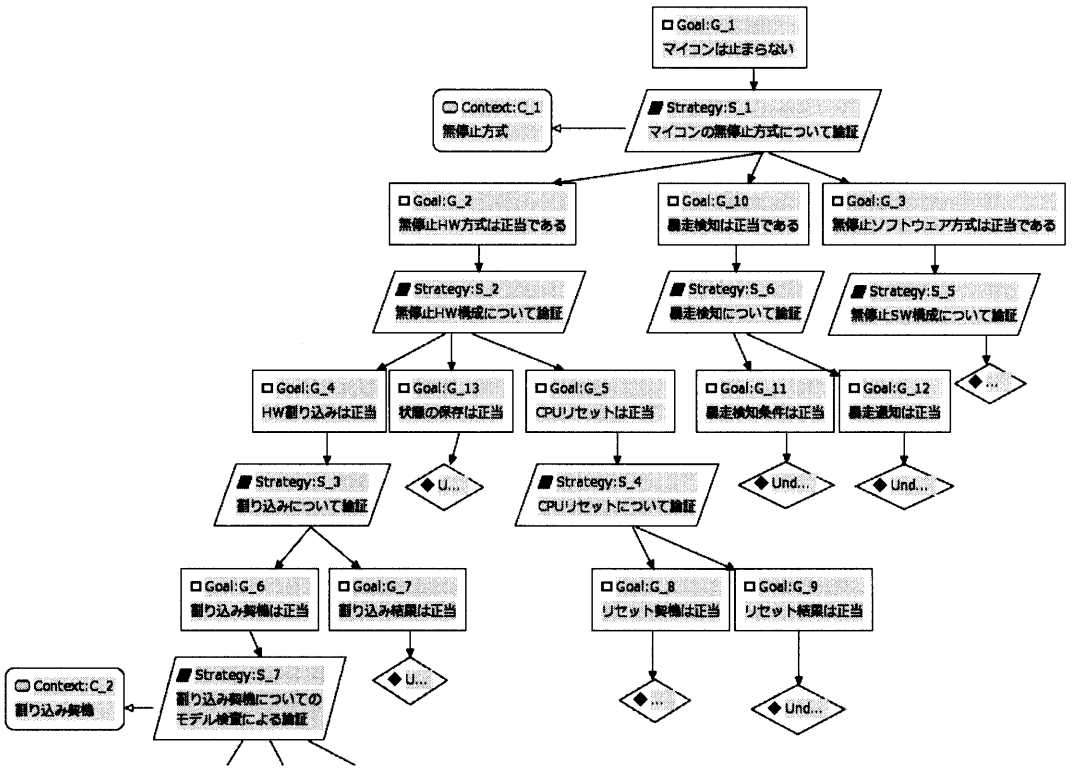
また，図 3 に示した検証ケース体系についても具体化していく必要があり，これらについても稿を改めて紹介するとともに，検証ケースケース作成ガイドラインとしてまとめる予定である．

謝辞

本研究は科研費基盤研究(S) 24220001「アーキテクチャ指向形式手法に基づく高品質ソフトウェア開発法の提案と実用化」の支援を受けたものである.

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] 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
[12] Robin Bloomfield and Peter Bishop, Safety and Assurance Cases: Past, Present and Possible Future – an Adelard Perspective, 2010
[13] 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
[14] D-Case エディタ, http://www.dependable-os.net/tech/D-CaseEditor/
[15] 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
[16] Donald Firesmith, QUality Assessment of System Architectures and their Requirements (QUASAR) Version 3.1, NDIA 12th Annual Systems Engineering Conference, 2009
[17] Steven Toulmin, Richard Rieke, Allan Janik , Introduction of reasoning, Macmillan publishing, 1984



付図１　マイコンは止まらないことに対する検証ケースの例