実践的保証ケース作成方式



山本 修一郎※1　森崎 修司※1　渥美 紀寿※2

システムの安全性や高信頼性を保証するために必要となる保証ケースの作成を実践的に支援できる保証ケース作成方式が必要である．このため，モデルに基づく保証ケースの統一的作成法，コードに対する保証ケース作成手法，客観的保証ケースレビュー法からなる実践的保証ケース作成方式を提案し，その有効性を実験的に確認する．

A Practical Approach to develop assurance case

Shuichiro Yamamoto※1, Shuji Morisaki※1, Noritoshi Atsumi※2

Practical assurance case development methods are necessary to validate safety and dependability of critical systems. In this paper, a model based unified assurance case creation method, code based assurance case development method and objective assurance case review method are proposed and experimentally evaluated.

1　はじめに

航空宇宙，自動車や医療機器など高い安全性が要求される複雑なシステムを実現するために，保証ケース（assurance case）の作成が必要とされるようになっている．例えば，自動車分野で導入が必要とされるISO26262機能安全規格などで，安全性に対する保証ケースである安全性ケースの作成が開発プロダクトだけでなく開発プロセスに対しても義務付けられている．このため，多様なモデルに対する統一的な保証ケースの作成法の研究，既存コードに対する保証ケースの作成法の研究，保証ケースの客観的なレビュー法の研究が必要である．これらの研究の必要性を以下に述べる．

(1) 多様なモデルに対する統一的な保証ケース作成法の必要性

システムの安全性を保証するためには，システムを構成するソフトウェアのモデルだけでなく，システムの利用モデルや構造モデルを明らかにすることにより，これらのモデルの安全性を保証ケースで確認する必要がある．これらのモデルを記述する言語には，BPMN（Business Process Modeling Notation），UML（Unified Modeling Language）やSysML（Systems Modeling Language）などがある．BPMNでは業務プロセスをモデル化できるのでシステムの利用・運用プロセスを定義できる．現代のソフトウェア開発で広く普及しているUMLで記述できるモデルには，ユースケース図やクラス図，シーケンス図，状態図，アクティビティ図などが含まれる．要求図などによりUMLを拡張したSysMLは組込みシステム向けに普及してきている．これらの多様なモデルに対して保証ケース作成法を用意できないと，産業界への保証ケースの適用は進展しない．

保証ケースの作成を容易化するために，アーキテクチャやプロセス構造に基づく保証ケースパターンが報告されている．しかし，多様なモデルに対する統一的な保証ケースの作成法については明らかになっていない．

これまで，著者らはネットワーク装置監視システムの保証ケース作成へのアーキテクチャパターン適用評価について研究を進めてきた[Yamamoto2013a]．しかし，一般的なアーキテクチャに対する保証ケース作成法については明らかになっていなかった．また，これまで，多様な工程生産物に対するモデルに基づく50以上の保証ケースパターンを具体化してきた[Yamamoto 2013b]．しかし，このような個別的な方法では新たなモデルが登場するごとに新たな保証ケース作成法が必要になるという問題がある．

このため，本論文では，任意のモデルがモデルを構成する概念要素とその関係に基づいていることに着目して，「統一的保証ケース作成法」を提案する．もし，統一的保証ケース作成法がなければ，モデルごとに保証ケースを作成する手法を開発する必要があり，冗長であるだけでなく，非効率であるという問題がある．このため，任意のモデルに共通する概念要素とその関係に基づいて保証ケースを階層的に分解することによって，「統一的保証ケース作成法」を提案する．ここで，任意のモデルから保証ケースを作成できるという点でモデルごとに個別的に保証

※1　名古屋大学　　※2　京都大学

ケースを作成する手法ではないということから「統一的」という用語を使用している．

(2) 既存コードに対する保証ケース作成法の必要性

ソフトウェアコードを再利用することで，ソフトウェア開発を効率化するだけでなく，開発されたソフトウェアの品質を向上できる可能性がある．この場合，再利用対象コードの安全性を保証する必要があるため，コードに対する保証ケースの作成法が必要である．

既存コードの安全性を保証するには，上述したようなモデルに対する保証ケースだけでなく，モデルが作成されていないような既存コードに対する保証ケースが必要である．しかし，現状では，コードに対する保証ケース作成法は確立されていない．

(3) 保証ケースの客観的なレビュー法の必要性

既に作成された保証ケースもあることから，保証ケースを適切にレビューする方法が必要である．保証ケースの具体的なレビュー法については，保証ケースが正しい構成規則を用いて記述されていることや成果物と保証ケースとの追跡性並びに網羅性を確認する手法が報告されている．また，主張，証拠などの不完全性や依存性についての定義がある．しかし，保証ケースを構成する主張を記述する用語の適切性や一貫性，主張を下位の主張に分解する際の網羅性，主張に対する証拠の十分性などの観点から，保証ケースの妥当性を内容に踏み込んで客観的に確認するためのレビュー法は確立されていない．

なお，広辞苑[Shinmura1991]によれば「実際に行動するさま」が「実践的」であると定義されていることから，本論文では，手法を提案するだけでなく，実際の適用例を示すことで，提案した手法が実践的であることを明らかにする．

2 関連研究

システムの安全性を確認するために，安全性ケース(Safety case)，保証ケース(Assurance case，アシュアランスケース)やディペンダビリティケース(Dependability case)が注目されている．例えば，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[Jackson2008]．このため，GSN (Goal Structuring Notation)[Kelly1997][Kelly1998][Yamamoto 2009]を用いてこれらを記述する方法が提案されている．

保証ケースでは，ゴール(主張)，戦略，前提(コンテクスト)，根拠(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．しかし，実際のシステム開発プロジェクトの担当者による保証ケースの作成を容易化するためには，表記法だけでなく，システム開発プロセスやシステム開発文書に即した具体的な保証ケース作成手順を提供する必要がある．

このため，保証ケースの作成を支援するために，開発手順，開発文書に基づく作成法，議論分解手法，ツールについて研究されている．

保証ケースの開発手順の研究には，Kellyによる6段階開発手法[Kelly1997][Kelly1998]とEuropean Organisation for the Safety of Air Navigationが制定している安全性ケース開発マニュアル[European Organisation2006]がある．

6段階手法では，保証ケースを作成するために，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[Kelly1997][Kelly1998]．安全性ケース開発マニュアルでは，安全性ケースをレビューするためのチェックリストを提案している．しかし，これらの開発手順の研究では，具体的な開発文書を対象としていないという問題がある．

開発文書に基づく保証ケース作成法の研究には，複数のシステムから構成されるシステム(System of Systems, SoS)に対する手法[Despotou2007]と，複合的な開発運用文書に基づく手法[Denney2012]が提案されている．前者の手法では，SoS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，矛盾点の解消に基づいて保証ケースを作成できる．後者の方法では，概念文書，設計書，運用手順書，ハザードリストに基づいて安全性ケースを作成できる．しかし，これらの開発文書に基づく保証ケースの作成法では，特定の開発運用文書を対象としており，任意のモデルに対して適用できないという問題がある．

保証ケースを作成する際に必要となる議論分解の観点として，システム構成や機能構成，属性構成などが整理されている[Bloomfield 2010][Yamamoto2013a]．しかし，議論分解の代表的な例が提示されているだけであり，具体的な開発運用工程生産物に基づく保証ケースの作成法は提示されてはいないという問題があった．

DEOSプロジェクト[DEOS2013]では保証ケースを編集できるエディタを開発している．しかし，開発運用工程生産物に対して保証ケースを手作業で編集して作成する必要があるという問題があった．

このように，従来研究では，対象となる任意のシステム構造に基づいて，保証ケースを作成する具体的な手法が明確ではなかった．

3 保証ケース作成支援方式

以下では，モデルに基づく保証ケースの統一的作成法，コードに対する保証ケース作成法，保証ケースの客観的なレビュー法からなる実践的保証ケース作成方式(図1)について，目的，課題，手法の概要について述べる．

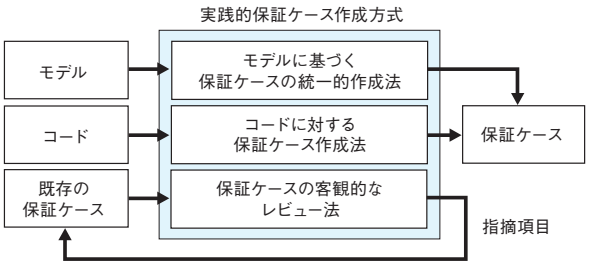

図1　実践的保証ケース作成方式の概要

3.1 モデルに基づく保証ケースの統一的作成法

【目的】

多様なモデルに対する保証ケースの作成を容易化するために，任意のモデルに対して適用できる保証ケースの統一的な作成法を確立する．

【課題】

これまで，モデルごとに保証ケースの分解パターンを用意していた．しかし，多様なモデルに対して個別に分解パターンを用意するのは限界があった．

【手法】

どのようなモデルも，必ずモデル要素と要素関係によって定義されている．したがって，モデルを保証する場合，モデル要素と要素関係に基づいて，保証ケースを一般的に分解することができる（モデル構成に基づく分解であることからこの分解をアーキテクチャ分解と呼ぶ）．また，モデル要素とその関係が期待される品質特性を持つことについて，品質特性の下位特性ごとに分解して保証することができる（品質特性の構成に基づく分解であることから，この分解を品質特性分解と呼ぶ）．更に，モデル要素とその関係が必要な品質特性を持つ上でのリスクとその対策が実施されていることに対して分解することができる（リスク対策についての分解であることから，この分解をリスク分解と呼ぶ）．このように，対象とするモデルが品質特性を持つことを，アーキテクチャ分解，品質特性分解，リスク対策分解に従って統一的に保証ケースを作成できる．

まず「システム」に着目して「システムが特性を満たす」という主張をアーキテクチャ分解パターンで構成要素とその関係に分解する．次に，「特性」に着目して，下位特性に分解できる場合は，特性分解パターンで下位特性に分解する．最後に，下位特性に着目して，下位特性のリスクに基づいて，リスク分解パターンで分解することにより，リスク対策の証拠を明らかにすることができる．

モデルは，構成要素とその関係によって定義される．例えば，ユースケース図の場合，アクターとユースケースがモデルの構成要素，アクターとユースケースの相互作用がモデルの関係である．同様にオープングループのアーキテクチャフレームワークであるTOGAF（The Open Group Architecture Framework）［TOG2011］のアーキテクチャ記述言語ArchiMate［Josely2013］［Yamamoto2016b］によるモデルも同様に，構成要素とその関係で定義できる．例えば，ビジネスアーキテクチャ（BA）モデルには，ビジネスイベント，ビジネスプロセス，ビジネス対象，ビジネス情報形式などの構成要素と，トリガ関係，利用関係，実現関係などがある．

したがって，図2に示すように，モデルに対して統一的なモデル分解パターンを構成できる．まず，モデルの構成に基づいて，モデルの構成要素とその関係で主張を分解する．次に，構成要素の種類並びに，構成要素関係の種類で主張を分解する．更に，構成要素と構成要素関係の実体に基づいて主張を分解する．

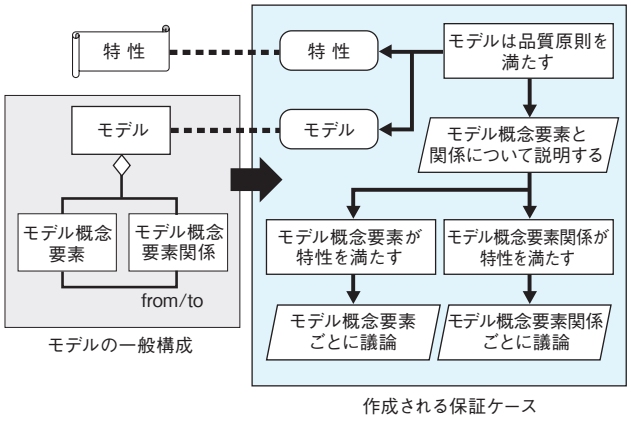

図2　モデル構造に基づく保証ケース作成の考え方

上述した着想に基づいて集合論の記法を用いることにより，定式化した保証ケース生成アルゴリズムを以下に示す［Yamamoto2015b］［Yamamoto2015c］．

【定義】保証ケース生成アルゴリズム

モデルをA＝<ConceptSet, RelationshipSet>とする．

ここで，ConceptSet＝{<Nc, Cc>｜Ncは概念名，Ccは概念種別}，RelationshipSet＝{<Nr, Cr>｜Nrは関係名，Crは関係種別}である．

Aについて，以下の集合を定義する．ここで，xはAの概念の具体例，rはAの関係の具体例である．

Aに出現する概念の集合

ConceptCategory（A）＝{ Cc｜<x, Cc>はAのConceptSetの要素}

Aに出現する関係の集合

RelationshipCategory（A）＝{ Cr｜<r, Cr>はAのRelationshipSetの要素}

Aに出現する概念のインスタンス集合

ConceptInstance（A）＝{ x｜<x, C>はAのConceptSetの要素}

Aに出現する関係のインスタンス集合

RelationshipInstance（A）＝{ r｜<r, Cr>はAの RelationshipSetの要素}

これの集合に基づいて，保証ケースを以下の手順で作成する．

最上位の主張を，「モデルAが特性を満足する」

第2レベルの主張に分解するための説明を「モデルAの概念と関係について説明する」として

第2レベルの主張を「モデルAの概念が特性を満足する」と「モデルAの関係が特性を満足する」とする

第3レベルの主張に分解するための説明ノードのラベルを「概念種別ごとに説明する」と「関係種別ごとに説明する」として

第3レベルの主張をConceptCategory（A）とRelationshipCategory（A）の要素ごとに以下のようにして作成する．ここで，CはAに出現する概念種別，CrはAに出現する関係種別である．

「モデルAの概念種別Cが特性を満足する」

「モデルAの関係種別Crが特性を満足する」

第4レベルの主張に分解するための説明ノードのラベルを「概念ごとに説明する」と「関係ごとに説明する」として第4レベルの主張をConceptInstance（A）とRelationshipInstance（A）の要素ごとに作成する

「モデルAの概念種別Cのインスタンスxが特性を満足する」

「モデルAの関係種別Crのインスタンスrが特性を満足する」

第5レベルの主張を，第4レベルの主張に対するリスクに次のようにして作成する

「モデルAの概念種別Cのインスタンスxが特性を満足する上でのリスクに対応できる」

「モデルAの関係種別Crのインスタンスrが特性を満足する上でのリスクに対応できる」

（アルゴリズム終わり）

このアルゴリズムに基づいて，図3に示す簡単なユースケース図から保証ケースが作成できる例を以下に示す．図3のユースケース図には，アクターaとb，ユースケースwがある．アクターとユースケースの関係として，r1とr2がある．ここで概念要素がa，b，w，概念間の関係がr1とr2である．このアルゴリズムに基づいて作成された保証ケースを図4に示す．ただし，紙幅の関係から，図4では，リスク対応については省略している．

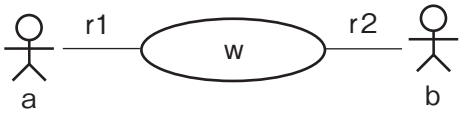

図3　ユースケース図の例

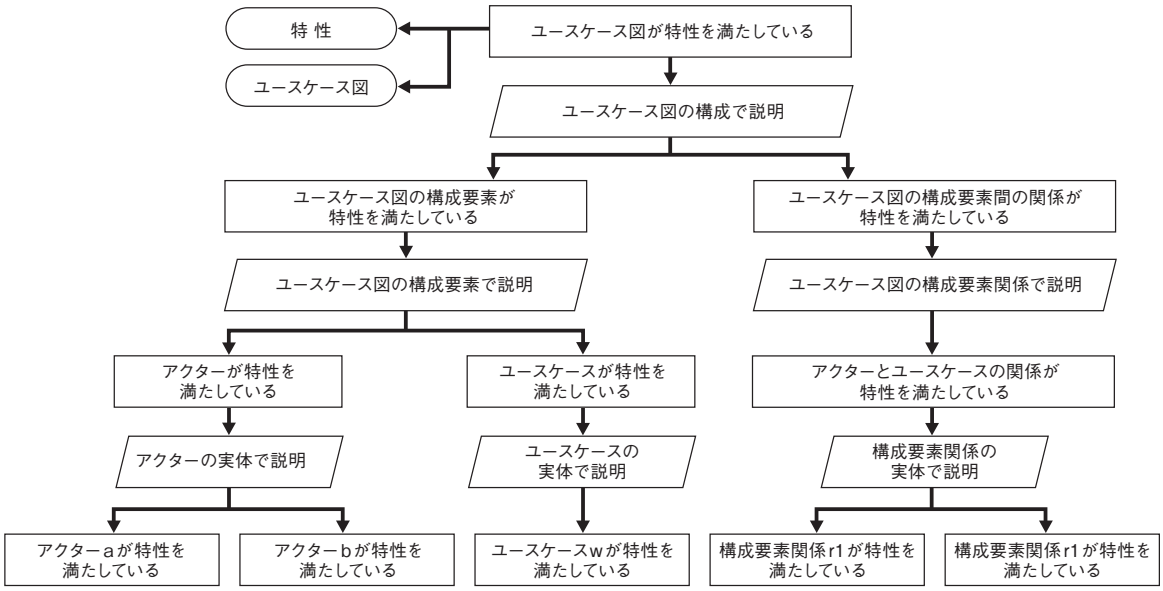

図４　ユースケース図に対応する保証ケース

また，このアルゴリズムに基づいて，ArchiMateのBAに対して保証ケースを作成できることについては文献[Yamamoto 2015b][Yamamoto2015c]を参照されたい．このように，上述したアルゴリズムを用いることにより，任意のモデルに対して統一的に保証ケースを作成できる．

3.2　コードに対する保証ケース作成法

【目的】

モデルが記述されていない既存コードに対する保証ケース作成法を具体化する．

【課題】

既存システムを保証する場合，モデルが定義されていないことが多いため，コードに対する保証ケースの作成法がないという問題があった．

【前提】

既存コードに対して，仕様はあるが，モデルはないことを前提としている．オープンソースソフトウェアなどの場合には，仕様とコードはあるがモデルは存在しない場合が多いことから，このような前提を置くこととした．また，仕様からモデルを作成して統一的保証ケース作成法によって保証ケースを作成する場合，保証すべき特性を「安全性」．概念要素を「関数の引数」，概念要素の関係を「引数関係」として，仕様の安全性を確認する保証ケースを作成することになる．したがって，モデルに基づく保証ケース作成法では，証拠の作成までは具体化していない点，コードに対する保証ケースではない点，コードの安全性という具体的な特性を考慮していない点が異なる．したがって，コードとその仕様がある場合には，モデルを作成するのは冗長であり，本節で述べるような直接的な保証ケース作成法が必要である．

【手法】

コードに対する保証ケースの作成では，対象となるコードへの入力と出力に着目する．入出力仕様に対して，対応するコードでは指定された入力に対する処理によって出力を作成する必要がある．

この点に着目して，コードに対する保証ケースでは，図５に示すように，「引数を説明」と「引数関係を説明」からなる入出力分解層で必要な入出力に対する主張を分解することにより，対応するコードについて具体化すべき証拠層を用意する．次いで，コードを探索して対応するコードがあれば，証拠として明記する．もし対応するコードがなければ，証拠がないことになり，仕様を実現するコードが抜けていることを検出できる．この考え方を，証拠に基づく欠陥摘出原理(Defect Detection by Evidence, DDBE)と呼ぶ．

保証ケースに基づくコード保証手順は以下のようになる．

【手順１】入出力仕様に基づき入出力制約を作成

【手順２】入出力制約に基づき，証拠が未定義になっている保証ケースを作成

【手順３】対応するコード断片を証拠に用いて，保証ケースを説明

【手順４】保証ケースの主張を説明するコード断片がない場合，コードの欠陥として指摘

（手順終わり）

保証ケースの証拠を上述した手順に従って説明できない場合，コードには欠陥があることになる．つまり，この手順はコードの欠陥を摘出する具体的な方法を与えている．この基本的な考え方を図示すると，図５のようになる．

この保証ケースでは，「コードが安全である」という主張を最上位ゴールとする保証ケースを入出力引数の関係仕様を前提として分解している．ここで，入出力引数や入出力引数関係に対する制約が引数仕様並びに引数関係仕様である．引数関係仕様には，入力引数間の関係，入力引数と出力引数の関係，出力引数間の関係がある．このような引数仕様並びに引数関係仕様が成立することを確認するコードがあるとき，「引数や引数関係が安全である」と言う．

最下位の証拠が，入力引数，出力引数，入出力引数の関係についての安全性を確認するために必要なコードを確認した結果に対応している．

3.3　保証ケースの客観的なレビュー法

【目的】

保証ケースレビュー観点の分類，観点に応じたレビュープロセスを定式化する．

【課題】

既存の保証ケースをレビューする場合，客観的なレビュー法

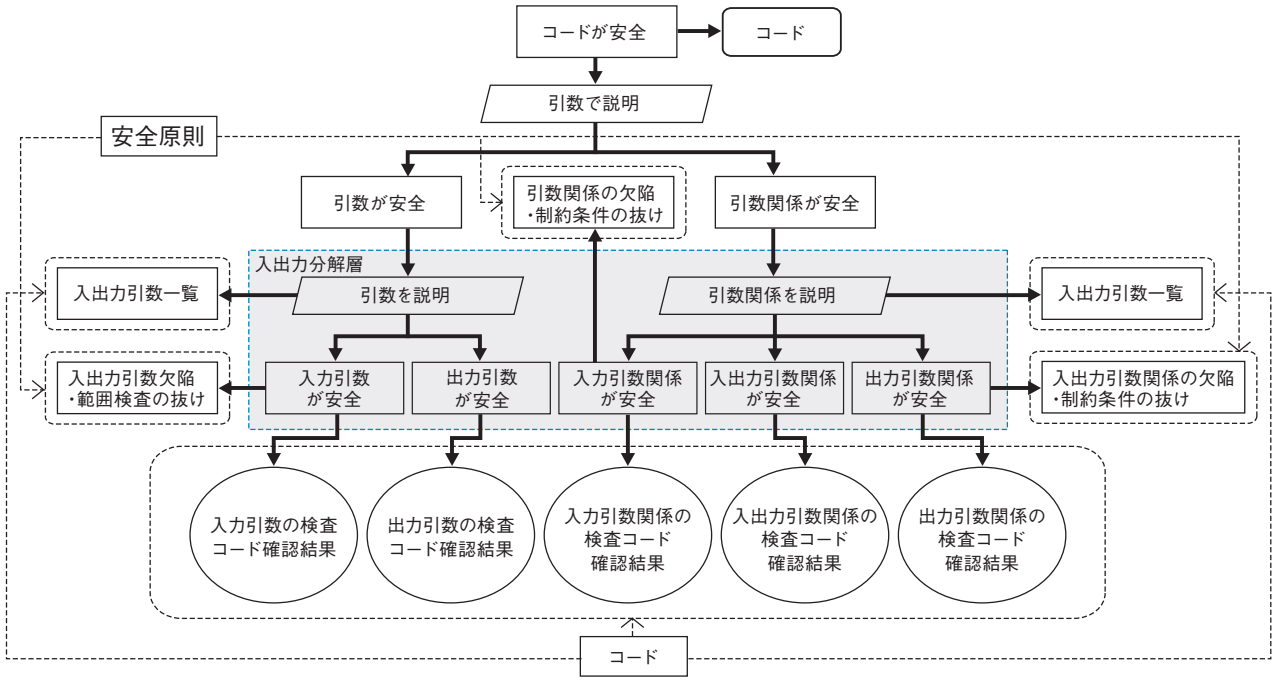

図5　コードに対する保証ケース作成法

が明確ではないため，属人的なレビューになりやすいという問題があった．とくに，保証ケースの主張では，「システムが安全である」などのように，日本語文を用いるため，用語関係があいまいになりやすいという問題があった．

【手法】

保証ケースのレビュープロセスを，理解，問題識別，原因分析，欠陥修正から構成し，理解段階で，保証ケースからシステミグラム［Boardman2008］を作成することにより，システミグラム上で問題識別，原因分析を客観的に実施できるようにする．また，この結果に基づいて，保証ケースの欠陥修正を容易化できる．

ここで，システミグラムは，名詞をノードとし，動詞をノード関係で表現する単純な図式である．システミグラムを用いて，保証ケースを構成する主張を理解することができる．

保証ケースの妥当性を保証対象の内容に踏み込んで客観的に確認するためのレビュー法を確立するため，システミグラムを用いて保証ケースが対象とする構成情報を図式化することにより，保証ケースの内容に基づくレビュー法について述べる［Yamamoto2015d］．

システミグラムのノードは，人工物，エージェント，重要属性を表す．ノードリンクは，ノード間の関係を表す．ノードの包含関係では，ノードの内部ノードで，下位ノードやノード属性を表すことができる．

システミグラムを用いた保証ケースのレビュー法では，レビューする保証ケースの主張を，以下の2種類に分類する．

［特性主張］＜対象＞が＜特性＞を満足している
［対策主張］＜対策＞によって＜リスク＞に対応している

特性主張は対象Sと特性Pの構成要素に従って，下位の特性主張に分解される．また，特性主張と対策主張には，上位の特性主張を下位の対策主張が説明するという関係がある．このとき，分解で指定される前提はリスクRの構成要素である．

対策Mについての主張は証拠Eによって説明される．すなわち，このような対策主張は最下位の主張である．特性主張を構成する対象と特性に対して，対象の内部状態として，「特性を満足している」を持つシステミグラムを対応付ける．対策主張の場合，「対象が特性を満足する」という上位の主張を実現するために，「対策によってリスクに対応している」という主張が必要になる．

上述した関係をまとめると，図6に示すようなシステミグラムになる．

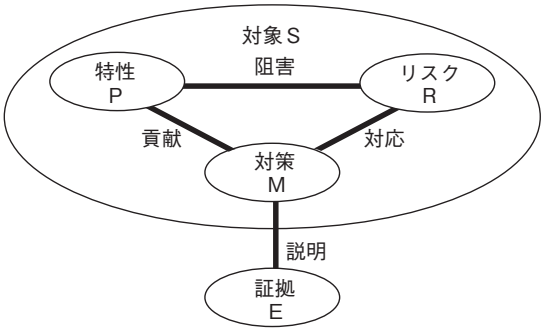

図6　対象，特性，リスク，対策，証拠とシステミグラム

システミグラムを用いて，保証ケースのレビュー手順を，①内容理解，②問題識別，③原因究明，④修正から，次のように構成できる．

［内容理解］保証ケースに対するシステミグラムを上述の方法で作成する
［問題識別］レビュー観点（表1）に基づき，用語関係，特性関係を分析して，問題を識別する
［原因究明］対応項目の欠落・誤りを検出することによって，問題原因を特定する
［修　　正］原因究明された問題について，欠落・誤り項目を補完・訂正する

表1　レビューの観点

観 点	定 義	指 標
完全性	必要な項目が含まれていること	不足項目数
明確性	あいまいさがないこと	不明項目数
適切性	不必要な項目が含まれていないこと	孤立項目数
追跡性	根拠が明確であること	追跡不能項目数

4　適用実験

4.1　統一的保証ケース作成実験

定義で示したアルゴリズムに従って，保証ケースを作成する支援ツールUC2CT(Unified Context to Claim Tool)を試作した(図7)．このツールでは，XMLで記述したモデル定義情報を入力として，保証ケース情報を図8に示すような表形式画面に生成して編集できる[Yamamoto2015a]．

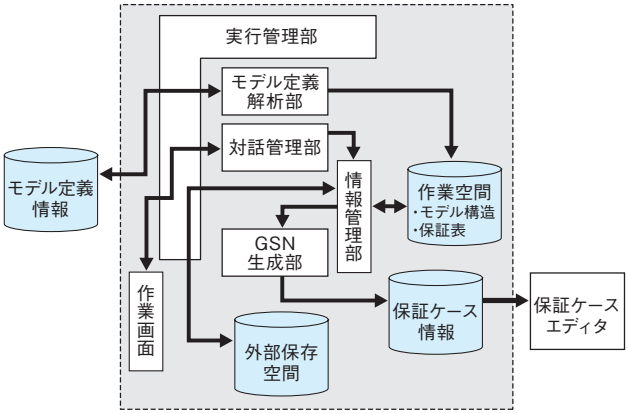

図7　保証ケース作成支援ツールの構成

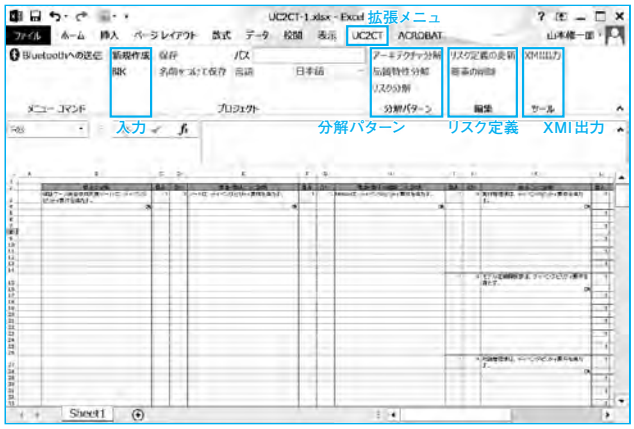

図8　UC2CT画面例

出力された保証ケース情報を保証ケースエディタに入力することにより，モデルに基づく保証ケースを確認できる(図9)．このツールによって任意のモデルに対して保証ケースを半自動的に作成できる．

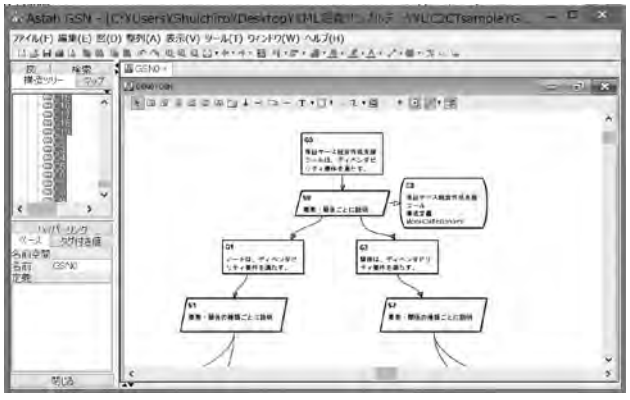

図9　生成された保証ケースの編集画面例

試作ツールを用いた保証ケース作成実験の結果を図10に示す．

実験対象としたモデルは，保証ケース作成支援ツールUC2CTのシステム構成を示す図7である．このシステム構成図に対する保証ケースの規模は，主張218個，戦略53個，前提47個，証拠165個となった．ツールを使わずに保証ケースを作成した時間は275分であった．これに対してツールを使用して保証ケースを作成した時間は47分であった．この結果，ツールの利用により，保証ケース作成時間を約5.6倍(約82%削減)に向上できることが判明した．

また，後者の作成時間にはモデル定義情報，品質特性情報，リスク対策情報を記述するためのXML定義時間28分を含む．もし，モデルからXML定義を自動抽出できれば，保証ケース作成時間が約14.5倍になることも判明した．なお，この19分の内訳は，ツールへのXML入力，アーキテクチャ分解，品質特性分解，リスク対策分解，保証ケースエディタ変換を含むことを注意しておく．

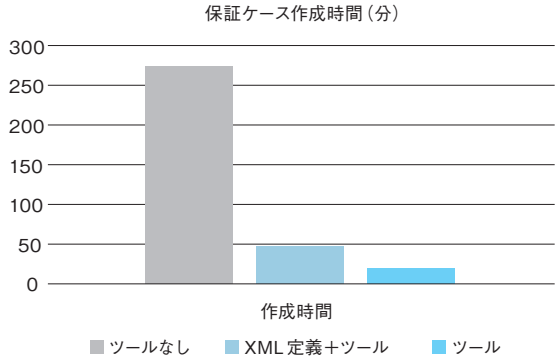

図10　保証ケース作成時間の比較

4.2　コードに基づく保証ケース作成実験

3.2節で述べたコードに対する保証ケース作成法の有効性を確認するために，具体的なコードを対象として保証ケースを作成する実験を実施した[Miyabayashi2015]．実験対象の概要は，次のようである．

対象とする入力仕様は，SSL/TLS Protocol V 1.0で，3584行からなる文章である．対象とするコードは，オープンソースのOpenSSL 1.0.1j s3_clnt.cである．コード行数は，3469であった．

被験者は，名古屋大学の学部4年生が1名である．

まず，入力仕様分析によって，11個のTBE(To Be Explained，証拠としてのコードの断片に対応する説明項目)を摘出した．対応するコードの断片を発見できれば，それを証拠としてTBEを置き換える．この入力仕様分析に要した時間は10時間．入力仕様に基づく保証ケース作成の時間は5時間だった．

次いで，Open SSLのコードを探索して10件のTBEに対する証拠として対応するコードの断片を具体化したが，残り1件のTBEについては，対応するコードがなかった．この1件の具体化できなかったTBEに対応する保証ケースの部分を図11に示す．

この図では，「セッションIDが空であるか，クライアントが送信した値と一致しない場合は安全である」という主張を説明しようとしている．この場合，対応するコードで鍵の入力条件を確認して，「暗号スィートの方式が，入力鍵を有効としない場合の処理は安全である」ことを説明する必要がある．このため，鍵の入力条件が不適切であるから，「エラー処理を中断していることを説明」する必要がある．この理由は，「不適切な入力処理の続行を禁止」する必要があるからである．したがって，主張「暗号スィートの方式が，入力鍵を有効としない場合に処理を中断している」に対応する証拠となるコードがなくてはならない．

ところが，このTBEに対応するコードが確認できなかった．このTBEが具体化できなかった部分が実際にOpen SSLの脆弱性に対応していた．このコードに基づく保証ケースの説明時間は8時間であった．

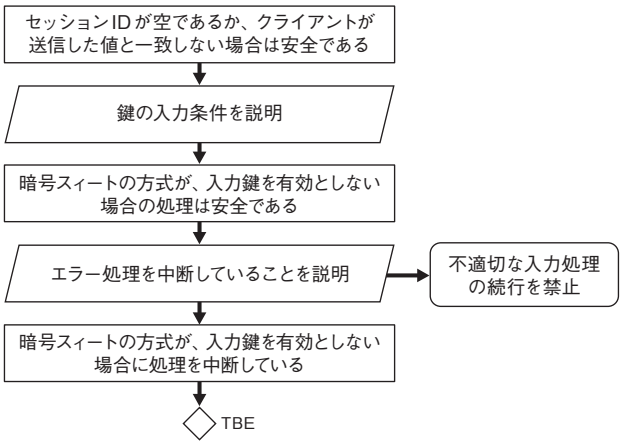

図11　TBEに対する保証ケース

4.3　保証ケースレビュー実験

同一の保証ケース事例を対象として，保証ケースを直接レビューする場合(直接法と呼ぶ)と，システミグラムを用いてレビューする場合(間接法と呼ぶ)とを比較する実験を実施することにより，間接法による保証ケースレビュー手順の有効性を確認する[Yamamoto2016a]．

実験対象システムは，電気ポットの加熱制御システムである．このシステムに対して，「加熱が安全である」という主張に対する保証ケースを14名の被験者(大学院学生)が作成した．

作成された14件の保証ケースを，保証ケースについて経験を持つ2名の大学院学生が次のようにしてレビューした．まず，直接法によって保証ケースをレビューした．次いで，システミグラムを用いた間接法で保証ケースをレビューした．

ここで，システミグラムを用いた間接法による保証ケースレビュー手順は以下のようになる．

a)保証ケースからシステミグラムを作成する手順を定義しておき，

b)この手順に従って被験者の作成した保証ケースからシステミグラムを作成して，

c)作成されたシステミグラムに対して，レビュー基準を用いて指摘項目を作成した．

図12に被験者による保証ケースから，レビュー実施者が作成したシステミグラムの例を示す．このように，保証ケースをシステミグラムに書き直すことでノード間の接続関係を明確化できることが分かる．

直接法と間接法による保証ケースのレビュー結果を表2に示す．ここで，数値の単位は件数である．また，2名のレビュー実施者が共に指摘した場合，共通欄の件数，そうでない場合差異欄の件数として計上した．

表2　レビュー実験結果

項目	直接法		間接法	
	共通	差異	共通	差異
完全性	0	0	15	10
明確性	0	63	53	6
適切性	0	0	4	53
追跡性	10	5	4	53
合計	10	68	76	122

間接法における適切性と追跡性の件数が一致しているのは，孤立ノードについては，上位ノードとの関係が追跡できなくなるためである．またシステミグラムを用いたレビューでは，対象とした保証ケースで多くのリスクが挙げられているものの，リスクに対する対策が一つも挙げられていない事例が多いことが判明した．また，対策が挙げられていないため，保証ケースの証拠ノードで示された内容が，不必要な証拠として指摘された．この結果，指摘件数が多くなったと思われる．

表2の結果から，直接法と間接法の平均指摘件数と共通指摘率を図示すると図13のようになった．ここで，平均指摘件数と共通指摘率は表2の合計欄の数値に基づいて下式で求めた．また14は被験者数である．なお，平均差異件数を求めるために，レビュア数2で除した．

平均指摘件数＝(共通合計件数＋(差異合計件数／2))／14
共通指摘率＝共通指摘件数／合計指摘件数

間接法の平均指摘件数は，約9.79，共通指摘率は約55.7%である．これに対して直接法の平均指摘件数は約3.14，共通指摘率は約22.7%となった．

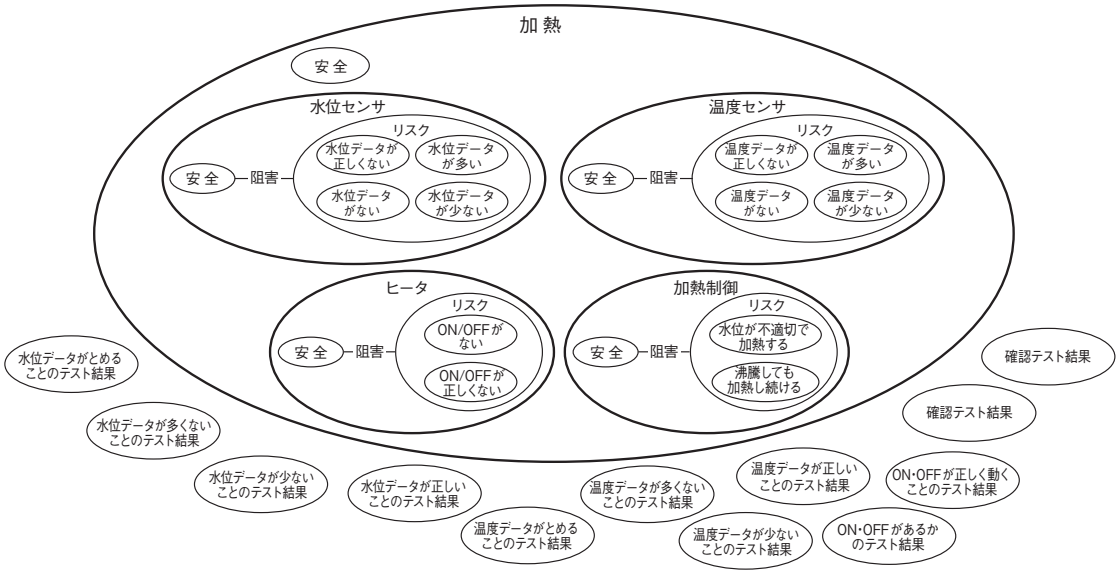

図12　保証ケースから作成したシステミグラムの例

この結果から，間接法が直接法よりも指摘件数で約3.1倍，指摘内容の共通率で約2.4倍になることが判明した（図13）．

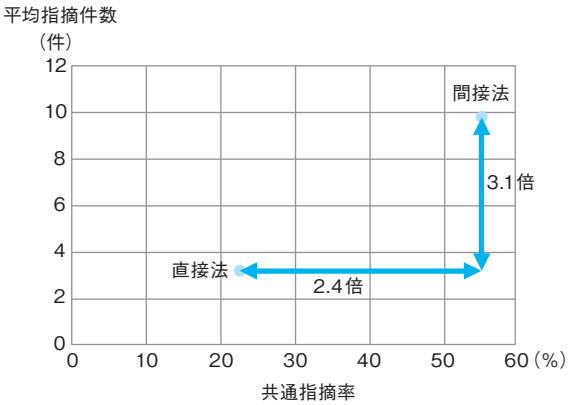

図13　保証ケースレビュー法の比較結果

5　まとめと今後の課題

本論文では，システムの安全性や高信頼性を保証するために必要となる保証ケースの作成を実践的に支援できる保証ケース作成方式を提案し，その有効性を実験的に確認した．提案した実践的保証ケース作成方式は，モデルに基づく保証ケースの統一的作成法，コードに対する保証ケース作成法，客観的保証ケースレビュー法からなる．

モデルに基づく保証ケースの統一的作成法では，要素と要素関係を持つ任意のモデルから統一的に保証ケースを作成できる保証ケース作成アルゴリズムを定式化した．また，このアルゴリズムに基づき，保証ケース作成支援ツールUC2CTを試作した．UC2CTでは，XMLにより，モデル，品質特性，リスク対策を定義しておくことにより，保証ケース情報を半自動的に生成できる．生成された保証ケース情報を保証ケースエディタに入力することで保証ケースを作成できる．

このUC2CTを用いて保証ケースを作成する実験で評価した結果，手動でモデルから保証ケースを作成する場合に比べて，UC2CTを用いることにより，作成時間を約5．6倍に向上できることを明らかにした．

コードに対する保証ケース作成法DDBEをオープンSSLに適用した結果，保証ケースで12個の具体化すべき証拠を識別した．この証拠に対して，コードを分析した結果11個の証拠にはコードが対応したが，対応すべきコードがない証拠を1件検出した．このコードが欠落した証拠がオープンSSLの脆弱性の欠陥原因であることを確認した．したがって，もし，オープンSSLのコードレビューにDDBEを適用していれば，オープンSSLの脆弱性を未然に防止できた可能性が高いことから，DDBEの有効性を確認できた．

客観的保証ケースレビュー法では，システミグラムに基づく保証ケースレビュー法を具体化した．また，保証ケースのレビュー指標として，完全性，明確性，適切性，追跡性を定義することにより，システミグラムを用いた間接法による保証ケースのレビューと，直接，保証ケースをレビューする直接法を比較するレビュー実験を実施した．この結果，間接法が直接法よりも，レビュアによる指摘内容の共通性が約2.4倍高いこと，また指摘件数が約3.1倍高いことを明らかにした．

本研究の制約として，保証ケース作成実験で対象としたモデルとコードがそれぞれ1件であること，また保証ケースレビュー実験の課題も1件であることなどの限界がある．ただし，これらの制約を考慮したとしても，各手法の有効性が明確になっており，ほかの事例に対しても，本手法の有効性を実証できると考えている．

今後の課題として，上述した制約の解消と，産業界における適用事例の拡大がある．また，本論文で提案した手法の教育と共に，3手法を統合した保証ケース作成支援環境の実現に向けた取り組みが必要である．

謝辞

本研究は，独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター（SEC：Software Reliability Enhancement Center）が実施した「2015年度ソフトウェア工学分野の先導的研究支援事業」の支援を受けたものである．

【参考文献】

[Shinmura1991] 新村出編，広辞苑，岩波書店，第四版，1991.
[Kelly1997] T. Kelly, A Six-Step Method for the Development of Goal Structures, York Software Engineering, 1997.
[Kelly1998] T. Kelly, Arguing Safety, a Systematic Approach to Managing Safety Cases, PhD Thesis, Department of Computer Science, University of York, 1998.
[Yamamoto2009] 山本修一郎，要求工学第58回アシュアランスケースとGSN，ビジネスコミュニケーション vol.46, No.8, pp.68-71, 2009.
[Jackson2008] D. Jackson, et al, Software for dependable systems– sufficient evidence?, NATIONAL RESEARCH COUNCIL, 2008.
[European Organisation2006] European Organisation for the Safety of Air Navigation, Safety Case Development Manual, 2nd ed., EUROCONTROL, 2006.
[Despotou2007] G. Despotou, T. Kelly, Design and Development of Dependability Case Architecture during System Development, proceedings of the 25th International System Safety Conference (ISSC), System Safety Society, 2007.
[Bloomfield 2010] R. Bloomfield and P. Bishop, Safety and assurance cases: Past, present and possible future – an Adelard perspective, procedings of the 18th Safety-Critical Systems Symposium, pp.51-67, 2010.
[DEOS2013] DEOSプロジェクト，http://www.jst.go.jp/crest/crest-os/
[Denney2012] E. Denney and G. Pai, I. Habli, Perspectives on Software Safety Case Development for Unmanned Aircraft, proc. 42nd Annual IEEE/IFIP Intl. Conf. Dependable Systems and Networks(DSN2012), pp.1-8, 2012.
[TOG2011] The Open Groupe, TOGAF Version 9.1, Van Haren Publishing, 2011.
[Josely2013] A. Josely, A., et al., ArchiMate® 2.0, A Pocket Guide, The Open Group, Van Haren Publishing, 2013.
[Boardman2008] J. Boardman, and B. Sauser, Systems Thinking: Coping with 21St Century Problems, CRC Press, 2008.
[Yamamoto2013a] S. Yamamoto, Y. Matsuno, An Evaluation of Argument Patterns to Reduce Pitfalls of Applying Assurance Case, 1st International Workshop on Assurance Cases for Software-intensive Systems, Assure 2013, pp.12-17, 2013.
[Yamamoto2013b] 山本 修一郎，主張と証拠，ダイテックオンデマンド出版，2013，978-4-86293-095-8.
[Yamamoto2015a] S. Yamamoto, Assuring Security through Attribute GSN, ICITCS 2015, pp.1-5, 2015.
[Yamamoto2015b] S. Yamamoto, An approach to assure Dependability through ArchiMate, Assure 2015, pp.50-61
[Yamamoto2015c] 山本修一郎，森崎修司，渥美紀寿，正田稔，モデルに基づく統一的保証ケース作成手法の提案，AI学会，KSN研究会，2015. 10.
[Yamamoto2015d] S. Yamamoto, An assurance case review method using Systemigram, AAA2015, 2015.
[Miyabayashi2015] 宮林 凌太，渥美 紀寿，森崎 修司，山本 修一郎，入力分析に基づくコード保証方法の提案，KBSE研究会，Vol.115, No.281, KBSE2015-39, pp.17-22, 2015.
[Yamamoto2016a] 山本修一郎，森崎修司，渥美紀寿，近藤純平，大林英晶，GSNレビュ実験と評価について，AI学会KSN研究会，2016, 3.
[Yamamoto2016b] 山本修一郎，現代エンタープライズ・アーキテクチャ概論 - ArchiMate入門 - , p.132, デザインエッグ社, 2016.7, ISBN-10:4865436804