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
SS2012-34 KBSE2012-36 (2012-7)

エンタープライズ・アーキテクチャに対するディペンダビリティケース作成法の提案

徳野　達也† 　松野　　裕†† 　山本修一郎††

† 名古屋大学 大学院情報科学研究科 〒464-8601 愛知県名古屋市千種区不老町
†† 名古屋大学 情報連携統括本部 〒464-8601 愛知県名古屋市千種区不老町
E-mail: †tokuno@nagoya-u.jp, ††{matsu,yamamotosui}@icts.nagoya-u.ac.jp

あらまし　複雑化した情報システムのディペンダビリティを保証することは難しく，またそのディペンダビリティを保証する方法について明確になっていない．組織内の複数の情報システムを最適化するための手法としてエンタープライズ・アーキテクチャが注目されている．そこでエンタープライズ・アーキテクチャのディペンダビリティを保証することは有用である．本稿では，代表的なTOGAFを対象とし，エンタープライズ・アーキテクチャのディペンダビリティを確認するために，TOGAFアーキテクチャ開発プロセスを構造化した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　エンタープライズ・アーキテクチャ，TOGAF，ディペンダビリティ，ディペンダビリティケース

A proposal on a method to describe enterprise architecture development process

Tatsuya TOKUNO†, Yutaka MATSUNO††, and Shuichiro YAMAMOTO††

† Graduate School of Information Science Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Aichi, 464-8601 Japan
†† Strategy Office, Information and Communications Headquarters Nagoya University
Furo-cho, Chikusa-ku, Nagoya, Aichi, 464-8601 Japan
E-mail: †tokuno@nagoya-u.jp, ††{matsu,yamamotosui}@icts.nagoya-u.ac.jp

Abstract　It is difficult to assure the dependability of today's large and complex systems, and a method of dependability assurance for such systems has not been well developed. Recently, enterprise architectures have been drawing attention to develop and optimize enterprise systems, which involves many large and complex information systems. In this paper, a method to describe dependability case to confirm the dependability of enterprise architecture is proposed. The proposed method is based on the architecture development method of TOGAF.

Key words　Enterprise Architecture, TOGAF, Dependability, Dependability Case

1. はじめに

近年，システムの安全性を確認する手法としてアシュアランスケースやディペンダビリティケースが注目されている．ディペンダビリティケースによってディペンダビリティを保証する対象は数多く存在するが，ほとんどが特定のシステムを対象として行われている．しかし企業などの組織では特定のシステムではなく複雑化した複合的なシステム群を対象とすることが多い．

そのため複数の文書と複数のディペンダビリティケースが集合レベルで対応することになることが多い．個々のシステムのディペンダビリティが検討されていても，それら全体を連携させた場合のディペンダビリティは検討が行われていないことがある．またDEOSプロジェクト[1]の一環としてD-Case Editor[2]と呼ばれるディペンダビリティケース作成を支援するツールが開発されているが，このツールを用いて複雑なシステム全体のディペンダビリティケースを記述するのは容易ではない．

そこで組織内の複数のシステムを最適化する手法であるエンタープライズ・アーキテクチャのディペンダビリティを確認することで，複雑化した組織のシステムのディペンダビリティを保証する．本稿では，エンタープライズ・アーキテクチャとし

て代表的なTOGAF[6]を対象とし，TOGAFアーキテクチャ開発プロセスを構造化したディペンダビリティケースを作成する手法を提案する．これによりディペンダビリティケースの作成を支援し，作成を容易にすることが期待できる．

以下に本論文の構成を述べる．2章では，ディペンダビリティケースとその記述方法について説明する．3章ではTOGAFと，本研究のディペンダビリティケース作成の対象となるADMについて説明し，その記述項目について説明する．4章では実際にその記述項目から最初のゴール分解する方法について検討する．5章では4章で行った分解をより詳細化するための関係分析を行う．6章では実際に作成したディペンダビリティケースについて考察する．最後に今後の課題について述べる．

2. ディペンダビリティケースについて

アシュアランスケース[3][4]は，主に欧米で普及しているシステムの安全性などを確認するために用いられる方法である．また主にディペンダビリティについて議論する場合は，ディペンダビリティケースとも呼ばれる．そこで本稿ではこれら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．

ディペンダビリティケースの記法の一つにGoal Structuring Notation(GSN)[5]がある．これはTim Kellyらによって提唱された要求を木構造に分解し，議論を容易にする表記方法である．議論すべきゴールをトップゴールとし，ゴール分解の理由などが記述されるストラテジに基づいてトップゴールを複数のサブゴールに分解する．またコンテキストとしてゴールを議論する際の条件などの情報が記述される，そして最下層のゴールにエビデンスを与えることでそのゴールを保証している．このように抽象的なトップゴールを分解し，各サブゴールを保証することで，トップゴールを保証することが可能である．

本稿ではディペンダビリティケースをGSNで記述する．

3. TOGAFとは

TOGAF(The Open Group Architecture Framework)[6]とは，エンタープライズ・アーキテクチャの導入，作成，利用，維持を支援するための手法と支援ツールを提供するエンタープライズ・アーキテクチャを開発するためのフレームワークである．

TOGAFは，ベスト・プラクティスおよび既存アーキテクチャ資産の再利用可能なセットによって支えられた，反復型プロセス・モデルに基づいている．またTOGAFの重要な要素は手法であるということである．これはつまりアーキテクチャ開発手法(ADM)によって経営ニーズに合致したエンタープライズ・アーキテクチャを策定することができる．

3.1 アーキテクチャ開発手法

TOGAFの中核には前述したアーキテクチャ開発手法(ADM)が存在する．ADMはアーキテクチャ開発のための，検証済みの反復可能なプロセスを提供する．ADMは，アーキテクチャ・フレームワークの確立，アーキテクチャ・コンテンツの開発，トランジションの実行，アーキテクチャの実現のガバナンスを含んでいる．これらのアクティビティは全て，継続的なアーキテクチャの定義と実現の反復サイクルの中で実行される．これにより，ビジネス・ゴールと機会に応じて，コントロールされた方法で，組織がエンタープライズを変革することが可能となる．

ADMには以下の複数のフェーズが存在する．

- 初期フェーズ
- フェーズA：アーキテクチャ・ビジョン
- フェーズB：ビジネス・アーキテクチャ
- フェーズC：情報システム・アーキテクチャ
- フェーズD：テクノロジ・アーキテクチャ
- フェーズE：機会とソリューション
- フェーズF：移行プランニング
- フェーズG：実践ガバナンス
- フェーズH：アーキテクチャ変更管理
- 要件管理

3.2 アーキテクチャ開発手法の記述項目とその分析

ADMはプロセス全体にわたって，上記のフェーズの間，あるいはフェーズの中で反復される．ADMの各フェーズはそれぞれ目的，入力，ステップ，出力で構成される．そこでADMの各フェーズにおける目的，入力，ステップ，出力に注目することで，各フェーズに対してディペンダビリティケースを作成する．本稿では，初期フェーズとフェーズA：アーキテクチャ・ビジョンを対象としてディペンダビリティケースの作成を行う．

4. ゴール分解方法の検討

ディペンダビリティケースを作成する上で，まずフェーズのトップゴールを設定する．初期フェーズを例に考えるとまずフェーズのトップゴールは「初期フェーズがディペンダブルである」である．設定したゴールからサブゴールへ分解する方法として，以下の4つがある．

(1) デリバラブルごとにサブゴールに分解する
(2) アクティビティごとにサブゴールに分解する
(3) 目的ごとにサブゴールに分解する
(4) ステップごとにサブゴールに分解する

実際にそれぞれの場合でサブゴールに分解を行ってみる．

(1)の場合，デリバラブルとは作業の成果物であり，成果物を作成する作業ごとにゴールを分解するという手法が考えられる．この場合，成果物ごとに作業を分類することができるためサブゴールへの分解として妥当である．しかし成果物はフェーズの出力であるためデリバラブルはエビデンスとすべきである．

(2)の場合，TOGAFには各フェーズのアクティビティが存在し，そのフェーズで行うことの概要が記述されている．従って，アクティビティごとに分解することで，そのフェーズで行わなければならないことの概要ごとにゴールを分解することができる．

(3)の場合，そのまま最初のゴールを目的ごとに分解する．しかしこの目的は，フェーズの直接の目的であるとは言いにくく，アクティビティやステップを行う目的や理由であると考えられる．そのため最初のゴールからいきなり目的で分解するには，文脈が通らず分かりにくい構造となってしまう．

(4)の場合，フェーズのステップであるため，ステップごとにディペンダビリティを議論するのは適切であると言える．し

かし (3) と同様に直接ステップに分解すると，分解数が多くなるため構造が分かりにくくなると思われる．

そこで本稿では，(2) のアクティビティごとに分解することとする．

5. ディペンダビリティケースの作成と関係分析

5.1 ゴール分解の詳細化のための関係分析

実際に TOGAF 文書の記述からディペンダビリティケースを作成する方法を検討する．前述のように ADM の各ステップには目的，入力，ステップ，出力が存在する．またディペンダビリティケースには，ゴール，サブゴール，コンテクスト，エビデンスといった要素が存在する．

そこでそれらの要素を図 1 のように対応させる．まず前章で述べたようにアクティビティでトップゴールを分解する．次にアクティビティに対応する目的でゴール分解し，目的に対応するステップでゴール分解を行う．そして実際に図 1 のようにディペンダビリティケースを作成する．今回はステップで分解する前に目的で分解を行うが，目的の対応関係や位置については議論の余地があるため，今後他のフェーズを分解する際の課題である．

表 1　TOGAF ADM とディペンダビリティケースの対応関係

活動要素	ディペンダビリティケース要素
目的	上位ゴール
ステップ	下位ゴール
入力	コンテクスト
出力	エビデンス

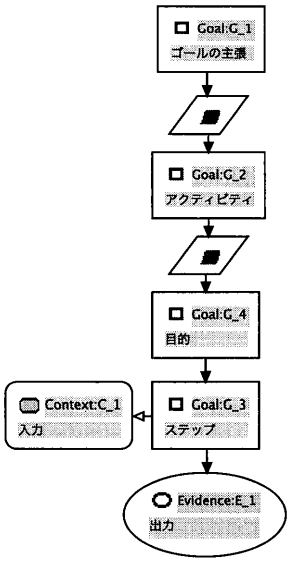

図 1　ADM 要素を利用したディペンダビリティケースの作成例

5.2 実際のフェーズへの適用

実際に以下の規則に従って，フェーズを分解する．

(1)　最上位のゴール「対象フェーズがディペンダブルである」をアクティビティごとに分解する

(2)　アクティビティを目的ごとにゴール分解する

(3)　目的をステップでゴール分解する

(4)　サブゴールである各ステップに対応する出力をエビデンスとして付加する

(5)　コンテキストとなる入力を対応するステップやストラテジに付加する

初期フェーズでは図 2 のようにフェーズのトップゴール「初期フェーズがディペンダブルである」をアクティビティごとに分解する．その後，アクティビティを目的で分解し，次に目的をステップで分解する．そして各ステップにそれぞれエビデンスとなる出力，コンテキストとなる入力を適宜ステップまたは上位のストラテジに付加する．実際に規則に従って初期フェーズのアクティビティの一つを分解したものは図 3 のようになる．

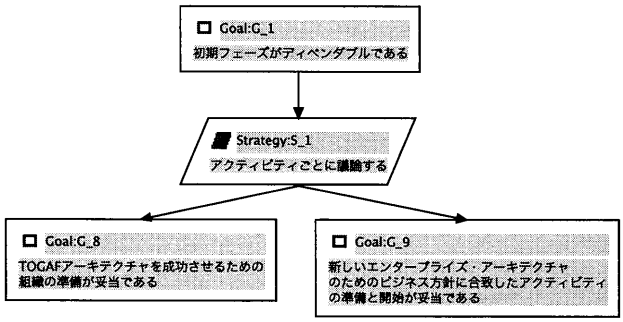

図 2　初期フェーズのゴールをアクティビティ要素で分解した例

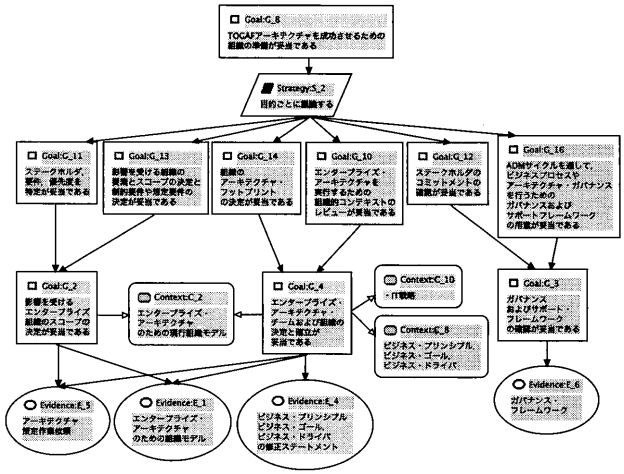

図 3　初期フェーズのアクティビティの一つを分解した例

ステップと出力などの要素間の関係は TOGAF 文書の記述から判断している．例えば，図 3 を参照すると，目的の一つと対応する G16 の妥当性を確認するためには，ステップの一つである G3 の妥当性が必要であるということを示している．また出力に対応する E1 と E5 のデリバラブルが G2 の妥当性を示している文書（成果物）として対応していることを示している．

このように実際に初期フェーズの分解を行ったものが図 4 であり，フェーズ A：アーキテクチャビジョンの分解を行ったものが図 5 である．また図 4 のディペンダビリティケースのノードの要素間の関係を表にしたものが付録の表である．表の読み

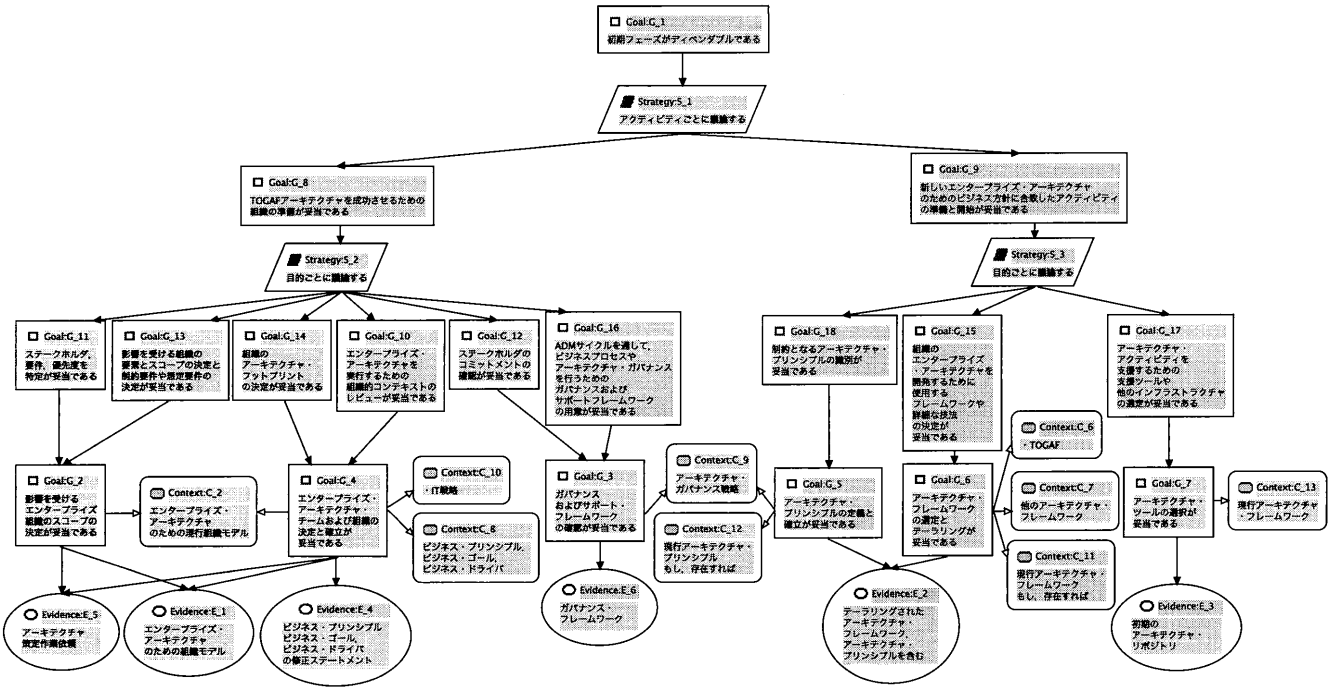

図 4　初期フェーズのディペンダビリティケース例

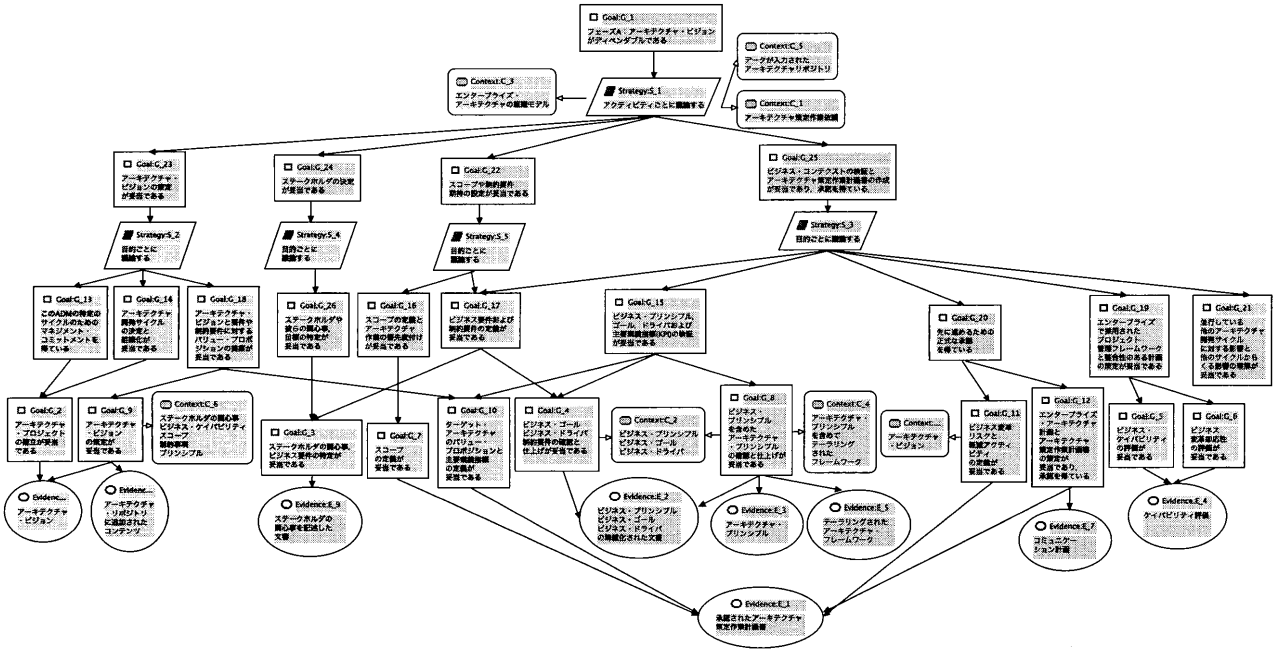

図 5　フェーズ A：アーキテクチャ・ビジョンのディペンダビリティケース例

方は，アクティビティ・目的要素間関係を例にすると，表の列の数字は目的の行の数字と対応している．アクティビティ1の「TOGAF アーキテクチャを成功させるための組織を準備」は目的 1，2，3，4，5，7 と関係していることを指している．表に記載されている “*” はディペンダビリティケースの関係の矢印に対応している．

6. 考　　察

提案手法を TOGAF 文書に適用することで，文書構造からディペンダビリティケースを作成することができることが分かった．しかし以下のような課題がある．

6.1 要素間の関係付け

階層構造を作成する上で，要素間の関係に着目して各要素が対応することを判断する必要がある．例えば初期フェーズの目的の一つである「制約となるアーキテクチャ・プリンシプルを識別する」という記述に対して，ステップの一つである「アーキテクチャ・プリンシプルを定義し，確立する」のように共通の単語が存在する場合は，何らかの関係があることが推測できる．

しかしそうでない場合は TOGAF 文書の記述の意味を理解して，要素間の関係を抽出する必要がある．例えば目的として

書かれている「ステークホルダのコミットメントを確認する」に関して，ステップの「ガバナンスおよび，サポートフレームワークを確認する」の段階で「ステークホルダと相談する必要があり，完了時点で起こりえる影響は関連するステークホルダに理解され，合意されている必要がある」と TOGAF 文書には記述されている．記述の詳細を読むことで目的とステップが対応していることが分かるが，要素の概要だけでは要素間関係が分からず，ステップの内容を TOGAF 文書から理解しなければいけない．

6.2 各ステップと入出力の関係

同様に各フェーズに対しては，入力と出力が定められているが，各ステップに関しては内容の記述があるだけで，何を利用し，そのステップの結果何が生成されるかが明確になっていない．また後述するフェーズ内部の入出力の存在などを考えると，各ステップに対応する入出力を明確にする必要がある．

6.3 ステップ内の入出力

前述したようにフェーズ A：アーキテクチャ・ビジョンで作成したディペンダビリティケースでは，フェーズ内部の入出力が存在した．具体的には最終的なフェーズの出力を生成するのではなく，後のステップである「アーキテクチャ・ビジョンの策定」での入力に使用する「ステークホルダの関心事」を生成するステップが存在した．そのためこのステップにはフェーズの出力としての出力が存在しないため，このステップにはエビデンスが存在しない．本稿ではこのようなステップの妥当性を確認するために，フェーズの出力としては記述されていないが，「ステークホルダの関心事を記述した文書」をエビデンスとしてステップに付加した．また同様に「アーキテクチャ・ビジョンの作成」ステップで利用する「ステークホルダの関心事」などの入力もフェーズの入力として記述されていないが，コンテキストとしてステップに付加した．このようにフェーズ内部だけの入出力は文書の入出力に記述がないため適宜判断して付加する必要がある．

6.4 ステップの妥当性確認

ディペンダビリティケースを作成するにあたって，アクティビティや目的やステップの内容をゴールに当てはめることができるように命題の形に変更している．そのためステップの活動が妥当であるかどうかをエビデンスによって判断している．エビデンスとして利用しているデリバラブルはフェーズの出力として生成されるが，このデリバラブルはステップが行われることで出力される．しかし，その行われた活動が本当に妥当な活動であったかまでは判断できない可能性がある．

そのため，各ステップに対してさらにゴール分解することで，ステップの妥当性を確認する必要がある．そこで各ステップがフェーズのように入出力と詳細なステップが記述されるべきである．もしくは，本稿ではディペンダビリティケースに加えていないが，エビデンスとしてデリバラブルとともに各ステップの活動のレビューを行った文書を付加させることでステップの妥当性を保証できる．

6.5 目的とステップの関係

図 5 のフェーズ A：アーキテクチャ・ビジョンのディペンダビリティケースを見てみると他のアクティビティから分解された目的同士が特定の同じステップを利用している部分が存在した．このことからステップに分解する際に，混ざってしまい，上位でカテゴリ分けした意味がなくなってしまっている．

またフェーズ A：アーキテクチャ・ビジョンの目的として「並行している他のアーキテクチャ開発サイクルに対する影響と他のサイクルからくる影響を理解する」が存在するが，これは特定のステップによって満たされる目的ではないと判断し，明確に対応するステップが発見することができなかった．コンテキストの場合は，さらに上位のストラテジに付加することが可能であることを考えると，目的はコンテキストとして記述することも考えられる．

7. おわりに

本稿では実際にディペンダビリティケース作成法を提案し，初期フェーズとフェーズ A：アーキテクチャ・ビジョンに対して適用した．また適用結果に基づき，以下の課題を明らかにした．

- 要素間の関係付け
- 各ステップと入出力の関係
- 入出力関係の曖昧さ
- ステップ内の入出力
- ステップの妥当性確認
- 目的とステップの関係

また今後 ADM の残りの 8 フェーズにも同様に本手法を適用することにより，有効性を評価する予定である．

謝　辞

本研究は CREST「実用化を目指した組み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．

文　献

[1] DEOS プロジェクト http://www.crest-os.jst.go.jp
[2] D-Case Editor http://www.dependable-os.net/tech/D-CaseEditor/
[3] Peter Bishop, Robin Bloomfield, Sofia Guerra, The future of goal-based assurance cases, DSN, 2004
[4] T. Scott Ankrum, Alfred H. Kromholtz, Structured Assurance Cases: Three Common Standards, IEEE International Symposium on High — Assurance Systems Engineering, 2005
[5] Tim Kelly and Rob Weaver. The goal structuring notation - a safety argument notation. In Proc. of the Dependable Systems and Networks 2004, Workshop on Assurance Cases, 2004.
[6] TOGAF Version 9 日本語訳版 http://www.opengroup.or.jp/togaf.html
[7] 山本修一郎，要求工学基礎知識，名古屋大学情報連携統括本部情報戦略室，2012

付　　録

表 A·1　初期フェーズのアクティビティ・目的要素間関係

	アクティビティ	1	2	3	4	5	6	7	8	9
1	TOGAF アーキテクチャを成功させるための組織を準備	*	*	*	*	*		*		
2	新しいエンタープライズ・アーキテクチャのためのビジネス方針に合致したアクティビティの準備と開始						*		*	*
	目的	1	2	3	4	5	6	7	8	9
1	エンタープライズ・アーキテクチャを実行するための組織的コンテキストをレビューする	*								
2	ステークホルダ，要件，優先度を特定する		*							
3	ステークホルダのコミットメントを確認する			*						
4	影響を受ける組織の要素とスコープの決定と制約要件や想定要件を決定する				*					
5	組織のアーキテクチャ・フットプリントを決定する					*				
6	組織のエンタープライズ・アーキテクチャを開発するために使用するフレームワークや詳細な技法を決定する						*			
7	ADM サイクルを通して，ビジネスプロセスやアーキテクチャ・ガバナンスを行うためのガバナンスおよびサポートフレームワークを用意する							*		
8	アーキテクチャ・アクティビティを支援するための支援ツールや他のインフラストラクチャを選定する								*	
9	制約となるアーキテクチャ・プリンシプルを識別する									*

表 A·2　初期フェーズの入力ステップ要素間関係

	入力	1	2	3	4	5	6
1	TOGAF					*	
2	他のアーキテクチャ・フレームワーク					*	
3	ビジネス・プリンシプル，ビジネス・ゴール，ビジネス・ドライバ			*			
4	アーキテクチャ・ガバナンス戦略		*		*		
5	IT 戦略			*			
6	エンタープライズ・アーキテクチャのための現行組織モデル	*		*			
7	現行アーキテクチャ・フレームワーク					*	
8	現行アーキテクチャ・プリンシプル				*		
9	現行アーキテクチャ・リポジトリ						*
	ステップ	1	2	3	4	5	6
1	影響を受けるエンタープライズ組織のスコープを決定する	*					
2	ガバナンスおよびサポート・フレームワークを確認する		*				
3	エンタープライズ・アーキテクチャ・チームおよび組織を決定し，確立する			*			
4	アーキテクチャ・プリンシプルを定義し，確立する				*		
5	アーキテクチャ・フレームワークを選定し，テーラリングする					*	
6	アーキテクチャ・ツールを選択する						*

表 A·3　初期フェーズの目的ステップ要素間関係

	目的	1	2	3	4	5	6
1	エンタープライズ・アーキテクチャを実行するための組織的コンテキストをレビューする			*			
2	ステークホルダ，要件，優先度を特定する	*					
3	ステークホルダのコミットメントを確認する		*				
4	影響を受ける組織の要素とスコープの決定と制約要件や想定要件を決定する	*					
5	組織のアーキテクチャ・フットプリントを決定する			*			
6	組織のエンタープライズ・アーキテクチャを開発するために使用するフレームワークや詳細な技法を決定する					*	
7	ADM サイクルを通して，ビジネスプロセスやアーキテクチャ・ガバナンスを行うためのガバナンスおよびサポートフレームワークを用意する		*				
8	アーキテクチャ・アクティビティを支援するための支援ツールや他のインフラストラクチャを選定する						*
9	制約となるアーキテクチャ・プリンシプルを識別する				*		
	ステップ	1	2	3	4	5	6
1	影響を受けるエンタープライズ組織のスコープを決定する	*					
2	ガバナンスおよびサポート・フレームワークを確認する		*				
3	エンタープライズ・アーキテクチャ・チームおよび組織を決定し，確立する			*			
4	アーキテクチャ・プリンシプルを定義し，確立する				*		
5	アーキテクチャ・フレームワークを選定し，テーラリングする					*	
6	アーキテクチャ・ツールを選択する						*

表 A·4　初期フェーズの出力ステップ要素間関係

	出力	1	2	3	4	5	6
1	エンタープライズ・アーキテクチャのための組織モデル	*		*			
2	テーラリングされたアーキテクチャ・フレームワークアーキテクチャ・プリンシプルを含む				*	*	
3	初期のアーキテクチャ・リポジトリ						*
4	ビジネスプリンシプル，ビジネス・ゴール，ビジネス・ドライバの修正ステートメントまたはその参照			*			
5	アーキテクチャ策定作業依頼	*		*			
6	ガバナンス・フレームワーク		*				
	ステップ	1	2	3	4	5	6
1	影響を受けるエンタープライズ組織のスコープを決定する	*					
2	ガバナンスおよびサポート・フレームワークを確認する		*				
3	エンタープライズ・アーキテクチャ・チームおよび組織を決定し，確立する			*			
4	アーキテクチャ・プリンシプルを定義し，確立する				*		
5	アーキテクチャ・フレームワークを選定し，テーラリングする					*	
6	アーキテクチャ・ツールを選択する						*