アクターを導入したアシュアランスケースにおける責任関係パターンの検討

猿渡　卓也
名古屋大学大学院 情報科学研究科
名古屋市千種区不老町
saruwatari.takuya@e.mbox.nagoya-u.ac.jp

山本　修一郎
名古屋大学 情報連携統括本部 情報戦略室
名古屋市千種区不老町
yamamotosui@icts.nagoya-u.ac.jp

概要
近年，IT システムサービスの障害に対する対処が大きな課題となってきている．この課題に対処するための技術として，システムのディペンダビリティに関して構造的に議論を記述できるアシュアランスケースが注目されている．アシュアランスケースを使って構造化される議論には，一定のパターンが存在する．そこで，本稿ではアクターを導入したアシュアランスケースにおけるアクター間の責任関係パターンについて検討し，システムを使うサービスの責任関係パターンとして定義した．この責任関係パターンは，システムを使うサービスのアシュアランスケースを対象にしており，アシュアランスケースを使った議論をすすめる際，活用されることが想定されている．

Keywords
アシュアランスケース，d* framework，責任関係パターン

1. はじめに
近年，重要なシステムサービスの障害が社会的な問題になってきている．複数のシステムが複雑に相互連携している状況が，この問題への対応を益々難しいものにしている．このような状況下で，システムのディペンダビリティを保証する手段として，アシュアランスケースが注目を集めてきている．アシュアランスケースとは，システムのディペンダビリティに関する議論を構造的に記述した文書である．アシュアランスケースを使うことにより，システムのディペンダビリティに関する議論を整理し，ステークホルダ間で共有することが可能となる．アシュアランスケースを使って構造化される議論は，パターン[1]から構成されていると考えることができる．本稿では議論を構築する際に使用することを目的として，アシュアランスケースにおけるパターンを検討し定義した．定義したパターンは，システムを使うサービスを対象としたアシュアランスケースにおけるアクター間の責任関係に関するパターンである．

本稿の構成は次の通りである．2 章では，関連研究について述べる．3 章では、d* framework について説明する．d* framework は，アクターを導入したアシュアランスケースの記法であり，本研究で使用している．4 章では，定義した責任関係パターンについて述べる．5 章で考察を実施し，6 章で本稿のまとめを実施する．

2. 関連研究
近年，アシュアランスケースに関する研究が数多く実施されている．一般的にアシュアランスケースは，構造的な図として表現される．アシュアランスケースの代表的な記法として，Goal Structuring Notation（GSN）[1][3]が提案されている．GSN とは，ゴール，戦略，コンテキスト，証拠の 4 つの要素を使ってシステムのディペンダビリティに関する議論を記述することができるアシュアランスケースの記法である．ここでの戦略とは，ディペンダビリティを保証するための戦略ではない．ディペンダビリティに関する議論を実施するための戦略である．GSN で作成されたアシュアランスケースの例を Figure 1 に示す．尚，この例は，GSN で記述されたアシュアランスケースを説明するために示したものであり，現実のシステムに対応した例ではない．

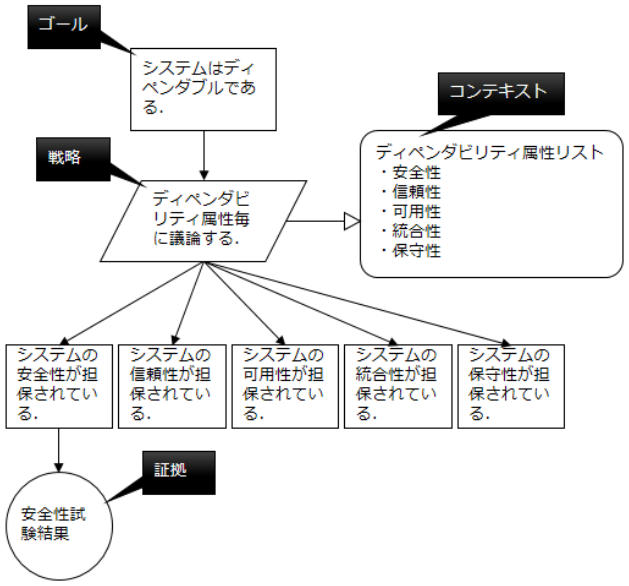

Figure 1　GSN で作成されたアシュアランスケースの例

アシュアランスケースのモジュール化に関する研究も複数実施されている．その中の一つとして，GSN にアクターの概念を導入した d* framework[4][5]が提案されている．

アシュアランスケースのパターンに関する研究として，アシュアランスケースに現れる議論のパターンに関する研究[3][6]，アシュアランスケース内の議論分解のパターン関する研究[7][8][9]等がある．これらは，ゴール，戦略，コンテキスト，証拠を使ってどのように議論を組み立てるかを示したパターンである．それゆえ，d* framework で新しく導入されたアクターに関係するパターンに関連する研究は存在しない．すなわち，アクター間の責任関係に関するパターンは存在しない．本稿では，アクター間の責任関係のパターンに関する検討を実施した．本稿において検討したアクター間の責任関係パターンは，アクター間の責任関係に特化したアシュアランスケースに現れる議論のパターンの一種であると考えることができる．

3. d* framework

d* framework[4][5]とは，GSN[1][3]を拡張したアシュアランスケースの記法である．d* framework により記述されたアシュアランスケースでは，GSN で定義されている要素に加えて，アクターの要素が使用される．アクターとはアシュアランスケースが対象としているサービスあるいはシステムを構成する要素である．システム，サブシステム，人，組織等がアクターとして定義される．アクターの導入により，アクター内のディペンダビリティと，アクター間のディペンダビリティを整理して議論することが可能となる．d* framework で記述されるアシュアランスケースには，下記の 2 種類が存在する．

1. アクターが他のアクターに対して責任を遂行することを表す全体のアシュアランスケース
2. アクター自体がディペンダブルであることを示すアシュアランスケース

本稿で検討し定義するシステムを利用するサービスの責任関係パターンは，d* framework を使って構築されるアシュアランスケースのうち，上記 1 で示される全体のアシュアランスケースを構築する際，利用することを想定している．d* framework で構築されたシステム全体のアシュアランスケースの例を Figure 2 に示す．尚，この例は，d* framework で記述されたアシュアランスケースを説明するために示したものであり，現実のシステムに対応した例ではない．また，上記 2 で示されるアクター自体のアシュアランスケースは，Figure 1 で示される GSN によるアシュアランスケースと同様のものとなる．

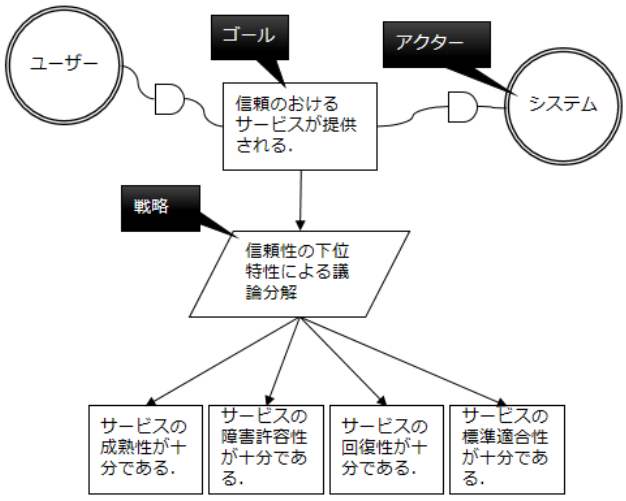

Figure 2　d* framework で作成された全体のアシュアランスケースの例

4. システムを利用するサービスの責任関係パターン

本稿では，システムを利用するサービスの責任関係パターンを検討し定義した．これは，d* framework を使った全体のアシュアランスケースにおいて，ユーザー，システム，サービス提供者，サービス開発者がアクターとして定義されている際の各アクター間の責任関係をパターン化したものである．

4.1 定義

本稿では，システムを利用するサービスの責任関係パターンを以下のように定義する．以下に，定義した責任関係パターンに関する，状況，問題，利点（問題の解決），欠点（限界）を示す．また，責任関係パターンとして定義したパターンそのものも示す．

- 状況

  ユーザー，システム，サービス提供者，システム開発者がアシュアランスケースのアクターとして定義されている．

- 問題

  システムを適切に開発運用して，効果的に利用する必要がある．そのため，アシュアランスケースに定義されている各アクター間における適切な責任関係について議論する必要がある．しかし，適切な責任関係の特定が難しいという問題がある．

- 利点（問題の解決）

  責任関係パターンの利用により，アクター間に適切な責任関係を特定できる．その結果，アクターの役割分担が明確になる．さらに，アクター間に特定された責任関係毎に下位の議論分解パターンの適用を容易化できる．

- 欠点（限界）

  アクター間の責任関係を明確にしようとするモチベーションが無く，責任関係を明確にできない場合，このパターンを適用することができない．

- パターン

  システムを利用するサービスの責任関係パターンは，階層的に定義されている．すなわち，Table 1 に示す 4 つの下位パターンにより構成される．下位パターンは，2 つのアクター（Depender，Dependee）と，アクター間の責任関係，及び責任関係の議論分解パターン[9]によって定義される．

  これらの下位パターンは，ユーザー，システム，サービス提供者，サービス開発者の 4 つのアクターを考慮してシステムを使ったサービスのアシュアランスケースを検討する際，考えられたものである．また，4 つのアクターが定義されている時，それらアクターの間に一般的に存在すると考えられる責任関係である．しかし，これらのパターンで，4 つのアクター間に存在する全ての責任関係を網羅しているという訳ではない．従って，不足している下位パターンについては随時追加する必要がある．

Table 1　下位パターン

ID	Depender	Dependee	責任関係	議論分解パターン
AP01	ユーザー	IT システム	サービス信頼性 に関する責任関係	信頼性の下位属性による属性分解
AP02	IT システム	サービス 提供者	運用責任 に関する責任関係	運用プロセスによるプロセス分解
AP03	IT システム	システム 開発者	開発責任 に関する責任関係	開発プロセスによるプロセス分解 開発プロダクトによるシステム分解
AP04	システム 開発者	サービス 提供者	発注責任 に関する責任関係	発注プロセスによるプロセス分解

4.2 パターンの利用方法

提案した責任関係パターンは，d* framework を使ったアシュアランスケースによる議論を構築する際，下位パターン毎に下記のように利用できる．

責任関係パターンを利用するための前提として，アクター（ユーザー，システム，サービス提供者，システム開発者）が定義されている必要がある．定義された 2 つのアクターの組が，Table 1 の下位パターン中の Depender，Dependee の組と合致する時，その下位パターンから，それらアクターの間に責任関係欄に示される責任関係が存在することがわかる．具体的な状況に応じて検討し従って，責任関係欄で示された責任関係が検討対象のサービスにおいて必要である場合は，これを用いたアシュアランスケース内での議論が可能となる．さらに，Table 1 より，どの議論分解パターンを使って議論を進めることができるのかがわかる．例えば，ユーザーと IT システムがアクターとして定義されている場合，AP01 の下位パターンからアクター間にサービス信頼性に関する責任関係が存在する可能性があることがわかる．さらに，そのサービス信頼性に関する責任関係は，信頼性の下位属性による属性分解により議論を進めることができるということがわかる．

4.3 パターンの利用例

ここでは，システムを利用するサービスの責任関係パターンを利用したアシュアランスケースの構築例を示す．尚，この例では，アシュアランスケースの記述に D-Case Editor[10]を利用している．そのため，アクターと，アクター間の責任関係を示す図表表現が異なる．

この例では前提として，ユーザー，システム，システム開発者，サービス提供者が定義されている状況を想定している．想定している状況を Figure 3 に示す．ここでは，D-Case Editor を使用する際のアクターの表記方法を使ってアクターを記述している．

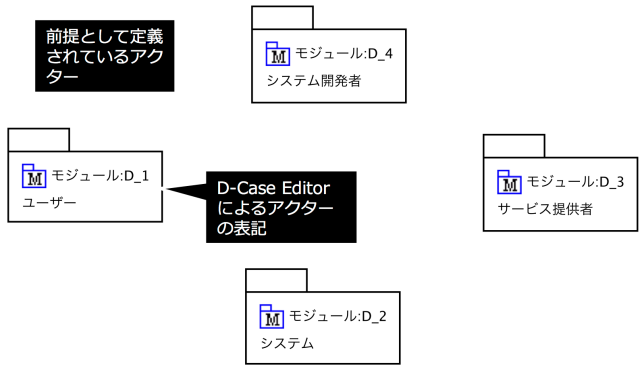

Figure 3　パターンの利用例①

次に，パターンを使ってユーザーとシステムの間にゴールを追加した例を Figure 4 に示す．AP01 より，ユーザーとシステムの間には，“サービス信頼性に関する責任関係”が必要であることがわかる．そのため，この例では“信頼のおけるサービスが提供される”というゴールを定義している．ここでは，D-Case Editor を使用した時のアクター間の責任関係の表記方法を使って，責任関係を記述している．

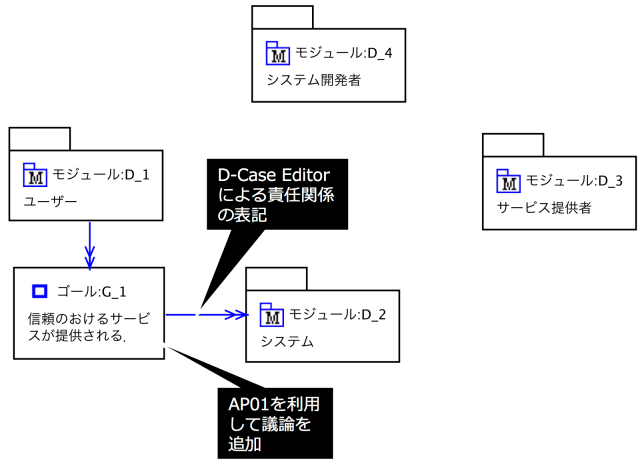

Figure 4　パターンの利用例②

次に，パターンを使って，その他のアクター間にもゴールを追加した例を Figure 5 に示す．AP02，AP03，AP04 より，3 つのゴールが追加されている．

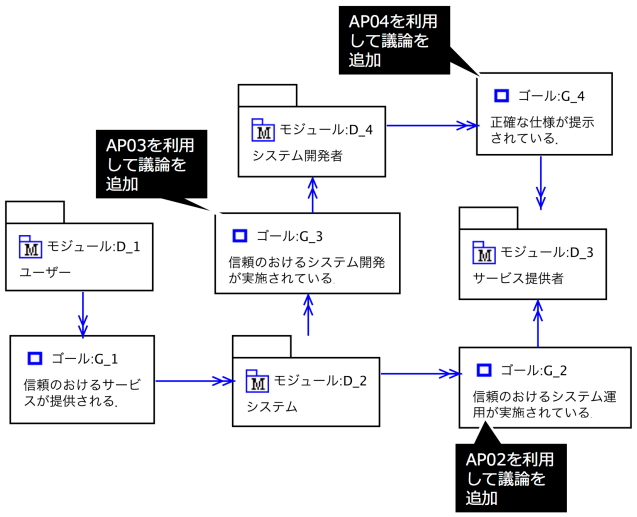

Figure 5　パターンの利用例③

最後に，パターンを使った議論分解の例を Figure 6 に示す．図では，ユーザーとシステムの間に定義したゴールに対するパターンを使った議論分解を示している．責任関係パターンより，このゴールは信頼性の下位属性による属性分解によって，議論を進められることがわかる．そこで，信頼性の下位属性である成熟性，障害許容性，回復性，標準適合性に関するゴールを下位ゴールとして定義している．これらの例に示すように提案したパターンを使用してディペンダビリティに関する議論を進めていくことができる．

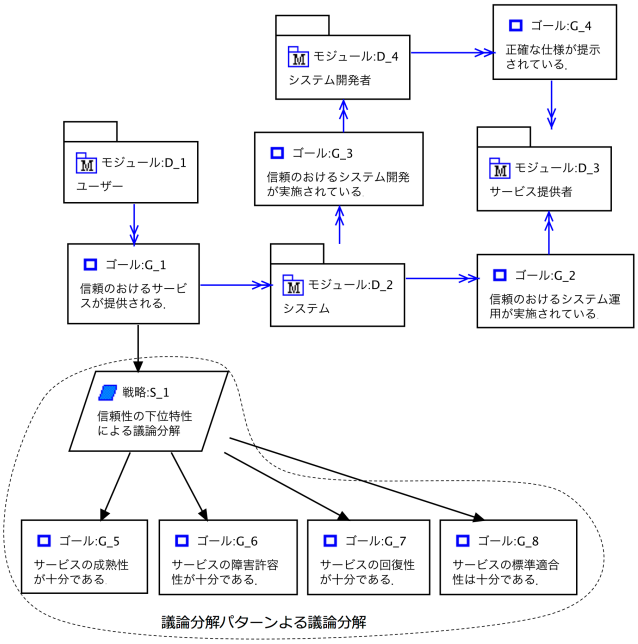

Figure 6　パターンの利用例④

5. 考察

5.1 下位パターンのサイズ

本稿では，アシュアランスケースにおける責任関係パターンを，階層的に定義した．すなわち，システムの責任関係パターンを，複数の下位パターンにより定義した．この下位パターンは，2 つのアクターと責任関係，及びその責任関係の議論分解パターンの組として定義されている．このようにパターンを階層的に定義することで，下位パターンのサイズを揃えることが可能となる．すなわち，下位パターンのサイズは，アクターのサイズにより規定される．また，下位パターンのサイズは 2 つのアクター間に存在する責任関係の大きさになる．パターンを検討する際，このようなパターンのサイズをどのように規定すれば良いのかが，しばしば問題となる．例えば，パターンが複数の要素（ゴールや戦略等）から構成される場合，パターンのサイズを自由に選択することができる．しかし，どのサイズが最適なのかを，その都度検討し決定することは難しい．また，様々なサイズのパターンが混在している状況では，それらパターンの利用に支障を来たす恐れもある．本稿で定義した責任関係パターンは，パターンを階層的に定義することで，アクターが決まると自動的に下位パターンのサイズが決定される．従って，新しいパターンを検討する際，そのサイズを考慮する必要がない．また，該当するアクターが定義されていれば，パターンを使用できるので，パターンの利用が容易になることが期待できる．

5.2 下位パターンの組み合わせ利用

本稿では，アシュアランスケースにおける責任関係パターンを複数の下位パターンにより定義している．ここでは，一つのアシュアランスケースに対して一つの下位パターンを使用するのではなく，複数の下位パターンを組み合わせて使用することを想定している．すなわち，具体的なサービスへの適用を考える際，必要な下位パターンを組み合わせて使用したアシュアランスケースを検討することを想定している．具体的なサービスにより不必要だと考えられる下位パターンを無理に使用する必要はない．このようにシステムを利用するサービスの責任関係パターンを複数の下位パターンにより定義することにより，真に必要となるパターンのみの使用が可能となる．

パターンの組み合わせ利用に関して，典型的なサービスで必要となる下位パターンのセットをパターン集合として規定しておき，使用することも可能である．この場合，典型的なサービスで必要となるパターン集合を予め定義しておく必要がある．その際，下位パターンはそれらパターン集合の個々の要素となる．

5.3 下位パターンの追加

本稿で定義した下位パターンは，2 つのアクターとアクター間の責任関係，及びその責任関係の議論分解パターンの 4 つの要素で構成されている．5.1 の考察で述べた通り，2 つのアクターが決まれば下位パターンのサイズが決定される．従って，新しい下位パターンを検討する際，パターンのサイズを気にする必要がなく，比較的簡単に下位パターンの追加を実施することができる．特に，現状では下位パターンの十分性は担保されていない．従って，不足すると考えられる下位パターンについては，随時追加する必要がある．そのため，下位パターンの追加が容易であることは有効であると考えられる．

具体的なアシュアランスケースが存在する場合，その具体的な議論内容を抽象化して下位パターンとして抽出することも可能である．その場合，具体的に実施されている議論

分解を参考にして，そのパターンで利用できる議論分解パターンを定義することができる．

5.4 限界

本研究は下記の点で限界がある．これら限界に対する対策は今後の課題となる．

1. 本稿で提案したパターンには，具体的な適用事例が無い．従って，提案したパターンに対する現実のサービスを対象とした評価が実施されていない．
2. 本稿で提案したパターンの適用先は，システムを利用するサービスに限定されている．また，アシュアランスケースに，ユーザー，システム，サービス提供者，サービス開発者の何れか 2 つ以上のアクターが定義されている必要がある．上記の状況にない場合，定義したパターンを適用することができない．
3. 本稿で提案したパターンを構成する下位パターンは，その十分性が担保されていない．従って，今後必要に応じてパターンを追加していく必要がある．

6. まとめ

システムのディペンダビリティに関する議論を整理して記述する技術としてアシュアランスケースが注目されている．本稿では，アクターを導入したアシュアランスケースにおける，アクター間の責任関係に関するパターンについて検討し定義した．このパターンは，システムを利用するサービスの責任関係パターンとして定義されている．また，このパターンは，階層的に定義されている．すなわち，定義したパターンは，2 つのアクターと，アクター間の責任関係，及びその責任関係の議論分解パターンにより定義される複数の下位パターンより構成されている．

今後は，具体的なアシュアランスケースを構築する場面において，定義したパターンを適用しパターンの有用性等を評価する必要がある．また，新しい下位パターンの検討と追加も今後の課題の一つである．

7. 謝辞

本研究の一部は 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）の支援を受けて行われた．

8. REFERENCES

[1] Alexander, C. 1984. パタン・ランゲージ―環境設計の手引, 鹿島出版会

[2] Kelly, T. A Six-Step Method for the Development of Goal Structures. York Software Engineering.

[3] Kelly, T. 1998. Arguing Safety - A Systematic Approach to Managing Safety Cases. PhD thesis, University of York.

[4] Yamamoto, S. and Matsuno, Y. 2012. d*framework: Inter-Dependency Model for Dependability. DSN 2012.

[5] Saruwatari, T. and Yamamoto, S. 2013. Definition and application of an assurance case development method (d*). SpringerPlus 2.1.

[6] Kelly, T. and Weaver, R. 2004. The Goal Structuring Notation - A Safety Argument Notation. Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases.

[7] Bloomfield, R. and Bishop, P. 2010. Safety and assurance cases: Past, present and possible future – an Adelard perspective, in proceedings of 18th Safety-Critical Systems Symposium.

[8] Yamamoto, S. and Matsuno, Y. 2013. An Evaluation of Argument Patterns to Reduce Pitfalls of Applying Assurance Case, In proceedings of ASSURE2013, co-located ICSE2013, San Francisco, CA, USA.

[9] 松野裕，高井利憲，山本修一郎，2012, D-Case 入門～ディペンダビリティ・ケースを書いてみよう!～，株式会社ダイテックホールディング

[10] D-Case Editor A Typed Assurance Case Editor, http://www.dependable-os.net/tech/D-CaseEditor/.