D-Case 構文定義書[*1]

DEOS 協会　D-Case 部会[*2]

2015 年 4 月 20 日

[*1] version 1.0

[*2] 主執筆者: 松野裕 (日大、matsuno.yutaka@nihon-u.ac.jp)、高井利憲 (奈良先端大、takai@is.naist.ac.jp)

はじめに (Informative)

本文書は D-Case の構文を定義する。D-Case の定義は以下である。

> 開発、運用、保守、廃棄などのシステムライフサイクルを通じて、システムのディペンダビリティをステークホルダが合意し、社会に説明責任を果たすための手法とツール。主として アシュアランスケース [1] を記述するための手法とツールを提供する。ドキュメント自体も D-Case と呼ぶ。(所真理雄編、DEOS:変化し続けるシステムのためのディペンダビリティ工学、近代科学社、2014)

D-Case はアシュアランスケースの記法の一つである GSN (Goal Structuring Notation)[2] を用いる。D-Case は GSN の定義文書である [2] の要件を満たす（ただし、[2] は曖昧な定義が多くあり、適宜解釈している。また曖昧すぎて定義できなかった部分、現時点では必要とないと思われる部分は D-Case では定義していない)。D-Case はさらに、システムのディペンダビリティに不可欠なモニタリングや責任関係などの概念を持つ。複雑になり過ぎない範囲で、パターンやモジュールなどプログラミング言語の基礎概念を導入し、[2] で曖昧であったそれらの定義を形式化した [3]。

D-Case は基本と標準の 2 つの構文レベルを持つ。基本構文はゴールやストラテジなどの基本ノードおよびリンクのみを定義する。日常業務におけるシステムのディペンダビリティ（システムの安全性やセキュリティなど）に関する合意形成において、最小限必要となる。標準構文は基本構文に加えて、パターンやモジュール、撤回可能 GSN 構文の形式的定義を持つ。本構文定義書は、D-Case の構文定義であることだけでなく、GSN の初の (準) 形式的定義書である。

本文書の構成は以下である。1 節において基本 D-Case 構文を定義する。2 節において、標準 D-Case 構文を定義する。3 節において、D-Case の XML Schema を示す。4 節において、本構文定義書の参照実装について述べる。

1 基本 D-Case 構文定義 (Normative)

D-Case は図 1 のような木構造をした図で表される。UML 図のように、数種類のノードがあり、矢印 (リンク) はノードを結ぶ。尚、図のうち、本構文定義書の定義のためではなく、説明に用いる図には、(Informative) と注記する。

1.1 ノード

すべてのノードは長方形、平行四辺形などの形状を持ち、属性として、*id*、文章を持つ。id、文章はそれぞれ string 型である。ノードによっては、*id*、文章以外の属性を持つ。ノードの定義を以下の形式で表す。

- (カタカナ表記、英語表記), 形状
- (ゴール、Goal): 長方形
- (ストラテジ、Strategy): 平行四辺形
- (コンテキスト、Context): 角が丸みを帯びた長方形もしくは樽型[*1]

[*1] GSN Community Standard [2] 8 ページ参照

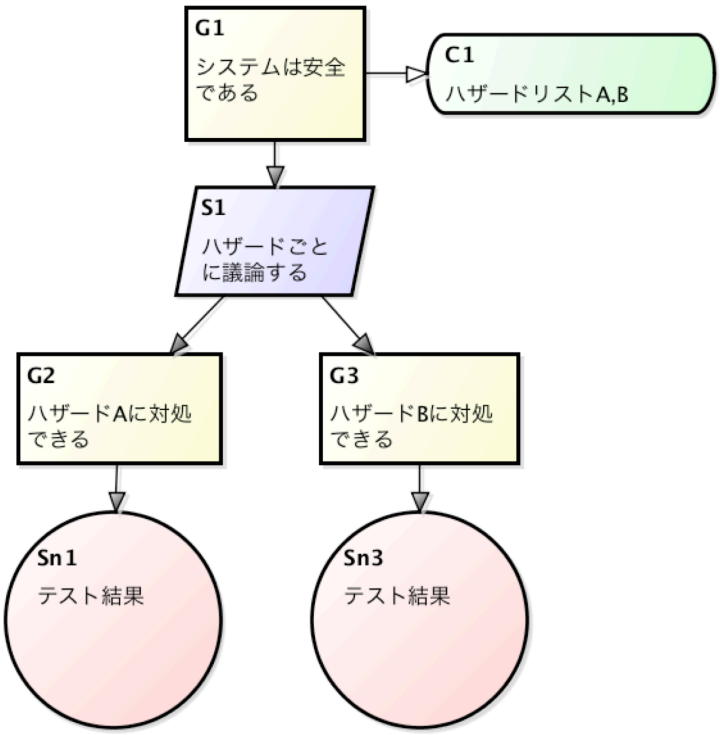

図 1　D-Case の例 (Informative)

- (エビデンス もしくは ソリューション、Evidence もしくは Solution): 楕円もしくは円
- (アンデベロップド、Undeveloped): 菱型

一つのツール、ドキュメントにおいて、カタカナ表記、英語表記を統一して用いる。例えば、一つのツールで、ストラテジをカタカナで、エビデンスを Evidence と表記してはならない。エビデンス、ソリューションの 2 通りの呼び方はあるのは、GSN Community Standard[2] では “Solution”が用いられているのに対し、アシュアランスケースの分野では一般に “Evidence”が用いられていることから、また DEOS において Evidence の概念がディペンダビリティに重要であるとの議論からアシュアランスケースが採用された経緯から、エビデンス、ソリューション両方を用いて良いことにした。ただし、一つのツール、ドキュメントではエビデンスもしくはソリューションのどちらかで統一して用いる。

図 2 にノードの例を示す。これらは D-Case Editor([*2]) での表現例である。`G_1`, `S_1`, `E_1` などは id、“System is Dependable”などは文章である。アンデベロップドノードは定義せず、後述するように、ゴールに属性をアンデベロップド属性を加えてもよい。

1.2　リンク

ノードはリンク（有向エッジ）によって結ばれる。リンクは 2 種類ある。リンクもノードと同様な定義形式をとる。

[*2] `http://www.dcase.jp`

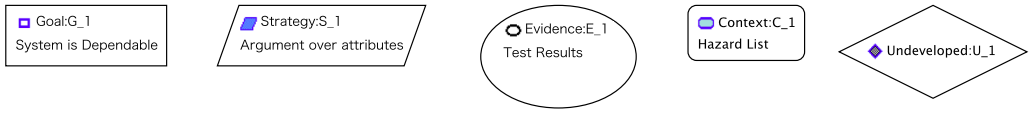

図 2　ノードの例 (Informative)

- サポート(Supported by) リンク: 塗りつぶし有りの矢印
- コンテキスト(In Context Of) リンク: 塗りつぶし無しの矢印

リンクの例を示す。

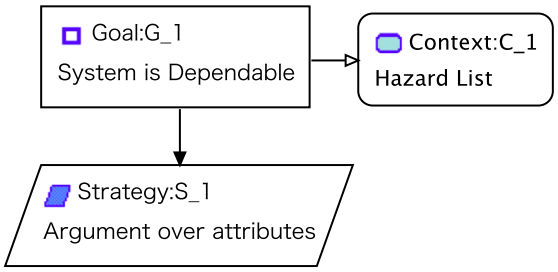

図 3　リンクの例 (Informative)

サポートリンクはゴールからストラテジ、ストラテジからゴール、ゴールからエビデンス、ゴールからアンデベロップドをつなげる。コンテキストリンクはゴールからコンテキスト、ストラテジからコンテキストをつなげる。リンクのつなげ方の規則を表 1, 2 にまとめる。読み方は、縦軸はリンクの始点ノードで、横軸は終点ノードを表す。◯ のつなげ方のみ許される。図 1 は正しいつなげ方の例である。

表 1　サポートリンクのつなげ方 (Normative)

	ゴール	ストラテジ	コンテキスト	エビデンス	アンデベロップド
ゴール		◯		◯	◯
ストラテジ	◯				
コンテキスト					
エビデンス					
アンデベロップド					

D-Case では、リンクの数に以下に制約をつける (Normative)。

- 一つのゴールは、唯一のストラテジ、エビデンス、アンデベロップドにつながる。
- 一つのストラテジは、一個以上のゴールにつながる。
- 一つのゴールは、任意個のコンテキストとつながる。
- 一つのストラテジは、任意個のコンテキストとつながる。

表 2　コンテキストリンクのつなげ方 (Normative)

	ゴール	ストラテジ	コンテキスト	エビデンス	アンデベロップド
ゴール			○		
ストラテジ			○		
コンテキスト					
エビデンス					
アンデベロップド					

アンデベロップドノードは、ゴールの一つの属性として定義し、独立のノードとして定義しなくてもよい。これは OMG のアシュアランスケースのメタモデル国際標準 SACM(Structured Assurance Case Metamodel)[4] での定義に従った場合である。図 4 では、これに従った Astah/GSN の例を示す。ゴールの属性として undeveloped を設定し、これに利用者がチェックを入れると、ゴールの下に小さな菱型が付けられ、undeveloped であることを示す。

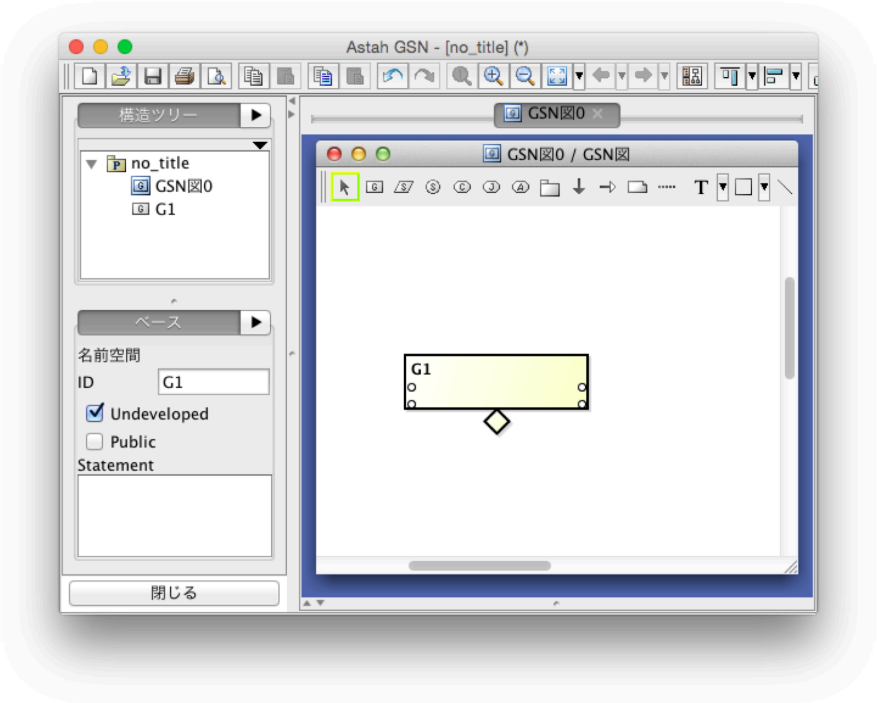

図 4　Astah/GSN での Undeveloped の実装例 (Informative)

2　標準 D-Case 構文定義

2.1, 2.2, 2.3, 2.4 節はそれぞれ独立している。別個に準拠して良い。

2.1 GSN 拡張ノード、D-Case 拡張ノード (Normative)

GSN[2] では、基本構文で示した以外のノードがある。D-Case では、システムのモニタリング情報を持つモニタノードなどがある。標準 D-Case 構文では、これらも定義する。パターン、モジュールについては、[3] を元に定義する。

標準 D-Case 構文では、ゴールは id、文章 に加え、$flag$: bool を属性として持つ。基本 D-Case 構文で定義した GSN ノード以外に、[2] で定義されているノードは以下である。

- (ジャスティフィケーション、Justification): エビデンスノードと同じ形状。ジャスティフィケーションノードであることを示すためにアイコンなどを用いる。
- (アサンプション、Assumption): エビデンスノードと同じ形状。アサンプションノードであることを示すためにアイコンなどを用いる。
- (モジュール、Module): 長方形に、左上に小さな長方形が乗る形状。
- (コントラクト、Contract): モジュールノードに、さらに右下に小さい長方形が乗る形状。

図 5 にジャスティフィケーション、モジュールノードの D-Case Editor での表示例を示す。コントラクトノードは、モジュール間図 (Inter-module notation) において、モジュール間をつなぐリンク上に定義される（後述する)。

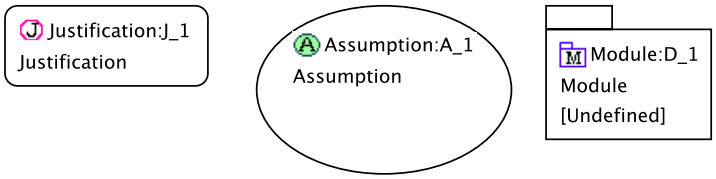

図 5　GSN 拡張ノード [2] の表示例 (Informative)

D-Case で新たに定義したノードは以下である。

- (モニタ、Monitor): エビデンスノードと同じ形状。モニタノードであることを示すためにアイコンなどを用いる。
- (アクション、Action): エビデンスノードと同じ形状。アクションノードであることを示すためにアイコンなどを用いる。
- (エクスターナル、External): モジュールノードと同じ形状。エクスターナルノードであることを示すためにアイコンなどを用いる。
- (パターン、Pattern): コンテキストノードと同じ形状。パターンノードであることを示すためにアイコンなどを用いる。パターンについては後に詳細を示す。
- (レスポンシビリティ、Responsibility): モジュールノードと同じ形状。レスポンシビリティノードであることを示すためにアイコンなどを用いる。レスポンシビリティには、以下の属性を id と文章以外に持つ。
  - 責任属性名：string

– 責任属性アドレス：string
– 責任属性アイコン：string

レスポンシビリティノードは、モジュール間図 (Inter-module notation) において、モジュール間をつなぐリンク上に定義される（後述する）。

レスポンシビリティノードを除く、D-Case 拡張ノードの D-Case Editor での表示例を示す (図 6)。

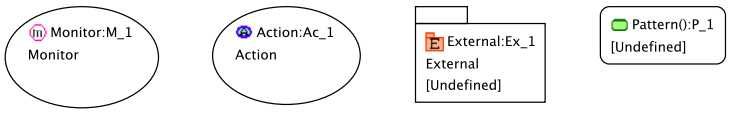

図 6　D-Case 拡張ノードの表示例 (Informative)

GSN 拡張ノード, GSN 拡張ノードを含めたリンクの関係を示す (図 3, 4)。ジャスティフィケーション、アサンプション、パターンはコンテキスト、モニタ、アクションはエビデンスを特殊化したノードであり、リンクの関係はそれぞれコンテキスト、エビデンスに準じる。エクスターナルはモジュールの特殊化である。

表 3　GSN, D-Case 拡張ノードを含めたサポートリンクのつなげ方

	G	St	Ct	Ev	U	J	As	Md	Mo	Ac	Ex	P
ゴール (G)		○		○	○				○	○		
ストラテジ (St)	○							○			○	
コンテキスト (Ct)												
エビデンス (Ev)												
アンデベロップド (U)												
ジャスティフィケーション (J)												
アサンプション (As)												
モジュール (Md)												
モニタ (Mo)												
アクション (Ac)												
エクスターナル (Ex)												
パターン (P)												

リンク数の制約は以下のように拡張される。

- 一つのゴールは、唯一のストラテジ、エビデンス、モニタ、アクション、アンデベロップドにつながる。
- 一つのストラテジは、一個以上のゴール、モジュール、エクスターナルにつながる。
- 一つのゴールは、任意個のコンテキスト、ジャスティフィケーション、アサンプションとつながる。
- 一つのゴールは、高々一つのパターンとつながる。
- 一つのストラテジは、任意個のコンテキスト、ジャスティフィケーション、アサンプションとつながる。
- 一つのストラテジは、高々一つのパターンとつながる。

標準 D-Case 構文では、パターン及びモジュールを定義する。詳細は [3] を参照のこと。まず、GSN を形式的に定義する。

表 4　GSN, D-Case 拡張ノードを含めたコンテキストリンクのつなげ方

	G	St	Ct	Ev	U	J	As	Md	Mo	Ac	Ex	P
ゴール (G)			○			○	○					○
ストラテジ (St)			○			○	○					○
コンテキスト (Ct)												
エビデンス (Ev)												
アンデベロップド (U)												
ジャスティフィケーション (J)												
アサンプション (As)												
モジュール (Md)												
モニタ (Mo)												
アクション (Ac)												
エクスターナル (Ex)												
パターン (P)												

定義 1（GSN 項 (Normative)）

$$T \ ::= \Diamond \mid (g, \Diamond) \mid (g, e) \mid (g, st, (T_1, \ldots, T_n))$$

g はゴールノード、st はストラテジノード、e はエビデンスノードを表すメタ変数である。この定義では、簡単のため、他のノードは省略してある (特にコンテキストノードは省略している)。$\Diamond$ は空の GSN を表す (後述のループパターンの定義のために必要になる)。$(g, \Diamond)$ は一つのゴール g がアンデベロップドノード $\Diamond$ につながっている GSN を表す。(g, e) は一つのゴール g が一つのエビデンスノード e につながっている GSN を表す。$(g, st, (T_1, \ldots, T_n))$ は一つのゴール g がストラテジ st を通して子供の GSN 項 $T_1, \ldots, T_n$ につながっていることを示している。

2.2　パターン構文 (Normative)

D-Case 構文定義書では、[2] で定義されている、以下のパターン構文を形式的に定義する。

- パラメータ (Parameter)
- マルチプリシティ (Multiplicity)
- ループ (Loop)
- チョイス (Choice)

定義 2（GSN パターン構文 P(Normative)）

$$\begin{aligned} d \ &::= \epsilon \mid [x : \tau = v] \\ P \ &::= \alpha \mid \Diamond \mid (g, \Diamond, d) \\ &\quad \mid (g, e, d) \mid (g, st, (P_1, \ldots, P_n), d) \\ &\quad \mid (g, st, \mathsf{c}[i, j](P_1, \ldots, P_n), d) \\ &\quad \mid (g, st, \mathsf{m}[i, j](P), d) \mid \mu\alpha.P \end{aligned}$$

$[x : \tau = v]$ はパラメータ $x : \tau$ に値 v を代入するパラメータ構文である。$(g, st, \mathsf{c})[i, j](P_1, \ldots, P_n), d)$ はチョイス構文である。$P_1, \ldots, P_n$ の子供のパターンから、$i \leq k \leq j$ を満たす整数 k 個のパターンを利用者が選択すると、トップゴールを g、ストラテジを st、選択された子供のパターンを子供とするパターンに変換される。

$(g, st, \mathsf{m}[i,j](P), d)$ はマルチプリシティ構文である。利用者が $i \leq k \leq j$ を満たす整数 k 個の同じ子供のパターン P を持つ、トップゴールが g、ストラテジが st であるパターンに変換される。$\mu\alpha.P$ はループ構文である。利用者がループの展開を選択すると、P の中の α が $\mu\alpha.P$ 自身に変換される。変換規則は $P_1 \longrightarrow P_2$ (図 7) の形で定義される。直感的な意味は、パターン P_1 から P_2 へ変換されるという意味である。

$$
\begin{gathered}
(g, \Diamond, [x : \tau = \bot]) \xrightarrow{v} (g[v/x], \Diamond, [x : \tau = v]) \\
(g, e, [x : \tau = \bot]) \xrightarrow{v} (g[v/x], e[v/x], [x : \tau = v]) \\
(g, st, (P_1, \ldots, P_n), [x : \tau = \bot]) \xrightarrow{v} \\
(g[v/x], st[v/x], (P_1[v/x], \ldots, P_n[v/x]), [x : \tau = v]) \\
(g, st, \mathsf{c}[i,j](P_1, \ldots, P_n), d) \overset{\{s_1, \ldots, s_k\}}{\longrightarrow} (g, st, (P_{s_1}, \ldots, P_{s_k}), d) \\
(g, st, \mathsf{m}[i,j](P), d) \xrightarrow{k} (g, st, (P, \ldots, P), d) \\
(P \text{ repeats } k \text{ times}) \\
\mu\alpha.P \xrightarrow{\mu} P[\mu\alpha.P/\alpha] \\
\mu\alpha.P \xrightarrow{\Diamond} \Diamond
\end{gathered}
$$

図 7　パターン変換規則 $P_1 \rightarrow P_2$(Normative)

D-Case では、これらのパターン構文を GSN に定義するために、パターンノードを導入する。パターンノードは以下の属性を持つ。

- サブタイプ: パラメータ、チョイス、マルチプリシティ、ループのいずれか。
- i, j: Subtype がチョイスかマルチプリシティである場合、それらの定義にある i, j。
- 葉ノード: サブタイプがループであった場合、ループの出口のノード。

これらの詳細は D-Case Editor、および論文 [3] を参照のこと。

パターンの例を示す (Informative)。$P_1 = (g_1, st_1, \mathsf{m}[1,3]((g_2, e_2, [FunctionName : string = \bot])))$ とする。g_1 = “System is dependable,” st_1 = “Argument over functions,” g_2 = “[$\bot$] is dependable,” e_2 = “Evidence for [$\bot$].”である。 P_1 の D-Case Editor での表現は図 8 である。

このように、パターンノードを対応するゴール（もしくはストラテジ）ノードに、つなげることにより、パターンとして定義する。P_1 の変換例を図 9 に示す。

2.3　モジュール構文 (Normative)

モジュールを含めた GSN の構文を以下に定義する。モジュール M は以下で定義される (Normative):

$$M ::= (T, f)$$

TGSN 項であり、f はフラグである。f が true であるモジュールは、他のモジュールから参照可能である。GSN 項 T は以下に拡張されて定義される。

$$
\begin{aligned}
T \ ::= \Diamond \mid (g, \Diamond) \mid (g, e) \mid (g, st, (T_1, \ldots, T_n)) \\
\mid M \mid \mathsf{ref}(M) \mid \mathsf{away}(M.g)
\end{aligned}
$$

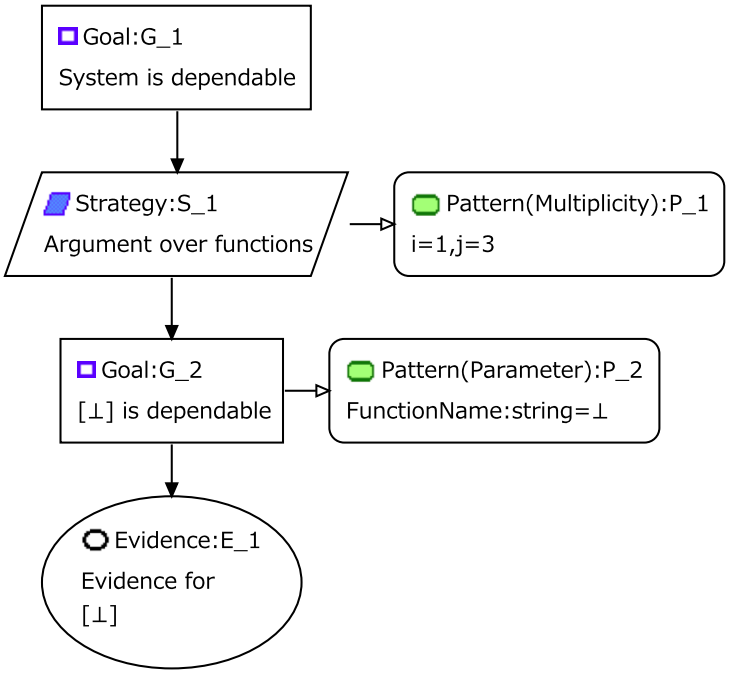

図 8　パターンの表現例 (Informative)

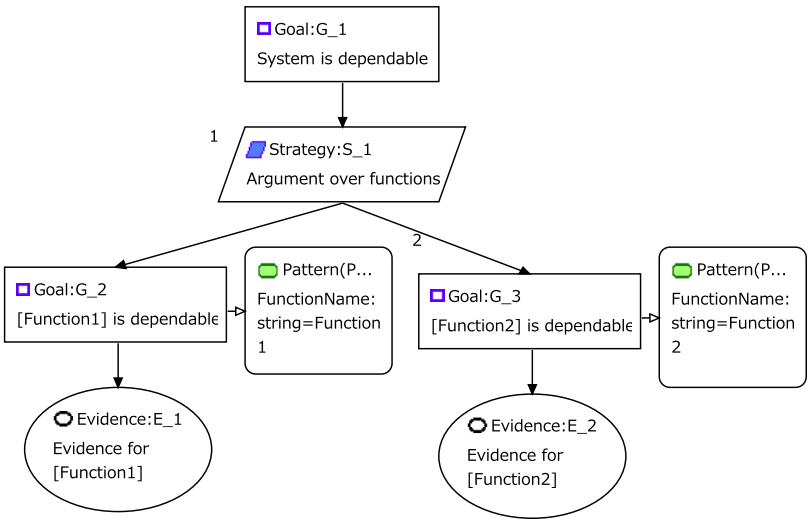

図 9　変換されたパターンの表現例 (Informative)

ref(M) は、参照されるモジュールである。away($M.g$) は [2] で定義されている away goal である。モジュール M の中のゴール g をトップとする GSN を参照している。away goal として参照されるゴールの $flag$ は true でなくてはならない。

モジュールシステム $\mathcal{M}$ はモジュールの集合として定義される (Normative)。

$$\mathcal{M} = \{M_1, \ldots, M_n\}.$$

Inter-module notation[2]$\mathcal{I}$ は $\mathcal{M}$ 上の Control Flow Graph として定義される (Normative)。

$$\mathcal{I} = (\mathcal{M}, \rightarrow_{cr})$$

エッジの添字 cr はコントラクトノードもしくはレスポンシビリティノードのいずれかである。M_2 が M_1 の中に現れるとき、$M_1, M_2 \in \mathcal{M}$ で、$M_1 \rightarrow_{cr} M_2 \in \mathcal{I}$ である。$M_1 \rightarrow_{cr} M_2 \in \mathcal{I}$ において、cr がコントラクトノードであるとき、M_1 と M_2 の依存関係に関する GSN がそのコントラクトノードにある。M_1 と M_2 の責任属性が定義されていて、それらが異なるとき cr はレスポンシビリティノードになる。M_1 と M_2 のレスポンシビリティ関係に関する GSN がそのレスポンシビリティノードにある。

例を示す (Informative)。$M_{\text{dependability}} = (T_1, f_1)$, $M_{\text{security}} = (T_2, f_2)$ とする。ここで

$$T_1 = (g_1, st_1, ((g_2, e_2), M_{\text{security}}.g_3))$$
$$T_2 = (g_3, e_3)$$

である。T_1 と T_2 を図示すると図 10 になる。T_1 の G_3 が away goal(緑色) であり、T_2 の G_1 が参照されて

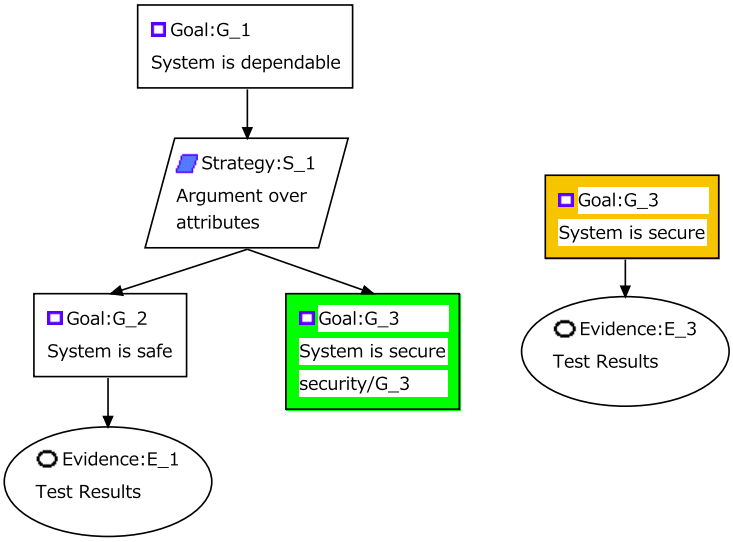

図 10　GSN 項 $T_1(M_{\text{dependability}})$ と $T_2(M_{\text{security}})$(Informative)

いるゴール (オレンジ色) である。

$\mathcal{M} = \{M_{\text{dependability}}, M_{\text{security}}\}$ の Inter-module notation は図 11 のようになる。図 11 では、コントラクトノードが M_1 と M_2 のエッジ上に定義されている。

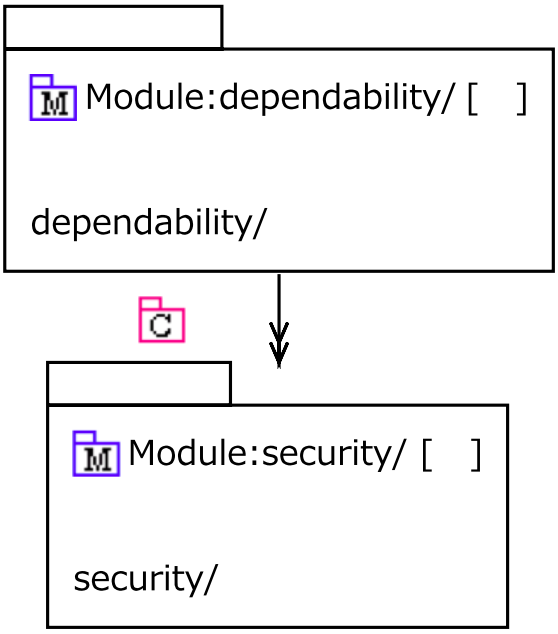

図 11　$M_{\text{dependability}}$、M_{security} の Inter-module notation(Informative)

2.4　撤回可能 GSN 構文 (Normative)

D-Case において、ゴールの主張に対してそれが成り立たないことを示す議論を展開したり、ストラテジに対して、例外を指摘したりするなど、反論を記述できるように補助的な記法を追加したものが、撤回可能 GSN 構文である。また、追加された補助的記法によって、ゴールやストラテジの状態を決定する意味論も与えることができる。例えば、ゴールの状態は、受理、却下、保留の三つがあり、それらが GSN 成長に従い、変化するため、撤回可能 GSN(defeasible GSN, dGSN) と呼ぶ。本稿では、撤回可能 GSN の構文のみを与える。各ノードの状態を決定する意味の与え方などは、文献 [5] を参照してほしい。

定義 3（撤回可能 GSN 項 T(defeasible GSN, dGSN)(normative)）

$$
\begin{aligned}
T &::= \Diamond \mid (G, \Diamond) \mid (G, (A_1, \ldots, A_n)) \\
A &::= E \mid (ST, (T_1, \ldots, T_m)) \\
G &::= (g, \texttt{presumption}) \mid (g, \texttt{except}) \\
ST &::= (st, \texttt{pro}) \mid (st, \texttt{con}) \\
E &::= (e, \texttt{pro}) \mid (e, \texttt{con})
\end{aligned}
$$

ここで、T は、撤回可能に拡張された GSN、A は、ストラテジをルートとする GSN の部分項、G は属性付きゴールノード、ST は属性付きストラテジノード、E は属性付きエビデンスノード、g は、ゴール、st は、ストラテジ、e は、エビデンスを表す。定義 1 と同じように、コンテキストなどの他のノードは省略しているが、同じようにゴールやストラテジにコンテキストを対応付けてもよい。ここで、定義 1 の GSN との違いを

以下に列挙する。

1. 一つのゴールが複数のストラテジまたはエビデンスを持つことを許す。
2. ゴールが属性 presumption または except を持つ。
3. ストラテジが属性 pro または con を持つ。
4. エビデンスが属性 pro または con を持つ。

複数のストラテジまたはエビデンスを持つゴールは、一般には、の一つのゴールに対して複数の視点から議論する際に用いる。例えば、図 12 の dGSN は、システムの安全性について、ハザードリストに基づくものと、安全ライフサイクルに基づくものの、二つの視点から議論している。ここで注意すべきは、一つのストラテジ内では、サブゴール分割などで網羅性が問題になるが、視点がいくつあるのかは、ステークホルダーに依存するため、網羅性の問題にはならない。ゴールの属性のうち、presumption は通常の役割である、上位ストラテジをサポートするゴールを表す。一方、except は、上位ストラテジに対して、そのストラテジが成立しない事柄を主張するゴールで、dGSN における反論の記述法の一つである。トップゴールは必ず presumption である。ツール上で表示する際には、presumption は省略できる。ストラテジの属性のうち、pro は通常のストラテジの役割である、上位ゴールをサポートするストラテジを表す。一方、con は、上位ゴールに対して反論を展開したり、ネガティブな側面を指摘するもので、dGSN における反論の記述法の一つである。ツール上で表示する際には、pro は省略できる。エビデンスの属性のうち、pro は通常のエビデンスの役割である、上位ゴールをサポートするエビデンスを表す。一方、con は、上位ゴールに対して反例を示したり、ネガティブな側面を表す証拠を示すもので、dGSN における反論の記述法の一つである。ツール上で表示する際には、pro は省略できる。

始めに、一つのゴールに複数のストラテジを持つ GSN の例

$$
\begin{array}{l}
((\texttt{G1}, \texttt{presumption}),\\
\quad((\texttt{S1}, \texttt{pro}), (((\texttt{G2}, \texttt{presumption}), (\texttt{Sn1}, \texttt{pro})),\\
\quad\quad((\texttt{G3}, \texttt{presumption}), (\texttt{Sn2}, \texttt{pro})))),\\
\quad((\texttt{S2}, \texttt{pro}), (((\texttt{G4}, \texttt{presumption}), (\texttt{Sn3}, \texttt{pro})),\\
\quad\quad((\texttt{G5}, \texttt{presumption}), (\texttt{Sn4}, \texttt{pro})))))
\end{array}
$$

のツール上での表現例を図 12 に示す。ここで、上に述べたように、presumption のゴール、pro のストラテジおよびエビデンスは、属性名は省略できるため、図中では特に示されていない。トップゴール G1 は二つのストラテジ S1 と S2 とによって支えられている。次に、反論を表すノードをもつ GSN の例

$$
\begin{array}{l}
((\texttt{G1}, \texttt{presumption}),\\
\quad((\texttt{S1}, \texttt{pro}), (((\texttt{G2}, \texttt{presumption}), (\texttt{Sn1}, \texttt{pro})),\\
\quad\quad((\texttt{G3}, \texttt{presumption}), (\texttt{Sn2}, \texttt{pro})),\\
\quad\quad((\texttt{G4}, \texttt{except}),\\
\quad\quad\quad((\texttt{S2}, \texttt{pro}), ((\texttt{G5}, \texttt{presumption}), (\texttt{Sn3}, \texttt{pro}))),\\
\quad\quad\quad((\texttt{S3}, \texttt{con}), ((\texttt{G6}, \texttt{presumption}), (\texttt{Sn4}, \texttt{pro})))))))
\end{array}
$$

のツール上での表現例を図 13 に示す。ここで、G4 は、属性 except を持つゴールであり、ストラテジ S1 に対する反論を意味し、ステレオタイプにより属性 except を表している。また、S3 は、属性 con を持つストラテジであり、ゴール G4 に対する反論を意味し、ステレオタイプにより属性 con を表している。 さらに、撤

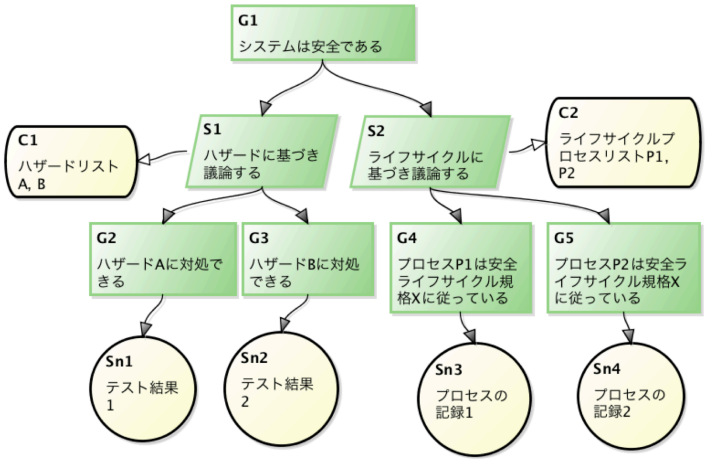

図 12　一つのゴールに複数のストラテジを持つ GSN の例 (Informative)

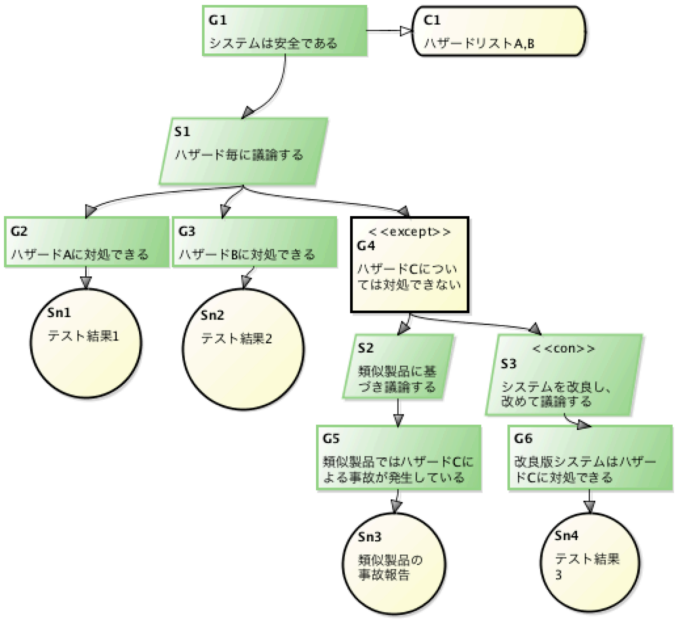

図 13　撤回可能 GSN の例 (Informative)

回可能 GSN においては、各ノードに対して、受理や却下、保留などの状態を決定することができるが、本稿では構文を与えることを主な目的としているため、ここでは省略する。詳しくは文献 [5] を参照してほしい。

3 D-Case XML Schema

3.1 D-Case XSD

D-Case の XML での定義を示す (Normative)。

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace=
          "http://www.dependable-os.net/2013/11/dcase" elementFormDefault="qualified"
   xmlns="http://www.dependable-os.net/2013/11/dcase" xmlns:xs=
       "http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.dependable-os.net/dre" schemaLocation="dre.xsd"/>
  <xs:element name="dcase">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="description" type="xs:string" maxOccurs="1" minOccurs="1" />
        <xs:element name="properties" type="PropertiesType" />
        <xs:element name="parameters" type=
            "ParametersType" maxOccurs="1" minOccurs="0"/>
        <xs:element name="responsibility" type=
            "ResponsibilityType" maxOccurs="1" minOccurs="0"/>
        <xs:element name="nodes" maxOccurs="1" minOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="node" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="description"
                       type="xs:string" maxOccurs="1" minOccurs="1" />
                    <xs:element name="properties"
                       minOccurs="1" maxOccurs="1" type="PropertiesType"/>
                    <xs:element name="parameters"
                       type="ParametersType" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="responsibility"
                       type="ResponsibilityType" maxOccurs="1" minOccurs="0"/>
                    <xs:element name="d-script" minOccurs="0" maxOccurs="1"/>
                  </xs:sequence>
                  <xs:attribute name="name" type="xs:string" use="optional" />
                  <xs:attribute name="type" use="required">
                    <xs:simpleType>
                      <xs:restriction base="xs:string">
                        <xs:enumeration value="Goal" />
                        <xs:enumeration value="Strategy" />
                        <xs:enumeration value="Evidence" />
                        <xs:enumeration value="Undeveloped" />
                        <xs:enumeration value="Context" />
                        <xs:enumeration value="Monitor" />
                        <xs:enumeration value="Justification" />
```

```
                <xs:enumeration value="Assumption" />
                <xs:enumeration value="Module" />
                <xs:enumeration value="Contract" />
                <xs:enumeration value="Action" />
                <xs:enumeration value="External" />
                <xs:enumeration value="Pattern" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="subtype" use="optional">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="Parameter" />
                <xs:enumeration value="Loop" />
                <xs:enumeration value="Choice" />
                <xs:enumeration value="Multiplicity" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="id" use="required" type="UUID">
          </xs:attribute>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="links" maxOccurs="1" minOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="link"
        maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="description" type="xs:string" />
            <xs:element name="properties" type="PropertiesType" />
          </xs:sequence>
          <xs:attribute name="source" type="xs:string" use="required" />
          <xs:attribute name="target" type="xs:string" use="required" />
          <xs:attribute name="id" type="UUID" use="required" />
          <xs:attribute name="name" type="xs:string" />
          <xs:attribute name="type" use="optional" default="SupportedBy">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="SupportedBy" />
                <xs:enumeration value="InContextOf" />
                <xs:enumeration value="Responsibility" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
```

```
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="id" type="UUID" use="required" />
  <xs:attribute name="name" type="xs:string" use="optional" />
  <xs:attribute name="status" type="xs:string" use="optional" />
 </xs:complexType>
</xs:element>
<xs:simpleType name="UUID">
 <xs:restriction base="xs:string">
  <xs:minLength value="1" />
 </xs:restriction>
</xs:simpleType>
<xs:complexType name="PropertiesType">
 <xs:sequence>
  <xs:element name="property"
   minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
    </xs:sequence>
    <xs:attribute name="name" use="required">
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:enumeration value="Attachment" />
       <xs:enumeration value="Message" />
       <xs:enumeration value="ParameterizedDesc" />
       <xs:enumeration value="Parent" />
       <xs:enumeration value="RefSource" />
       <xs:enumeration value="Flag" />
       <xs:enumeration value="Stakeholder" />
       <xs:enumeration value="IsNormal" />
       <xs:enumeration value="LeafNode" />
       <xs:enumeration value="I" />
       <xs:enumeration value="J" />
       <xs:enumeration value="SiblingOrder" />
       <xs:enumeration value="ValidUntil" />
      </xs:restriction>
     </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="value" type="xs:string" use="required" />
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="ParametersType">
 <xs:sequence>
```

```
      <xs:element name="parameter"
        maxOccurs="unbounded" minOccurs="1">
        <xs:complexType>
          <xs:attribute name="name" type="xs:string" />
          <xs:attribute name="type" use="required">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:enumeration value="string" />
                <xs:enumeration value="int" />
                <xs:enumeration value="double" />
                <xs:enumeration value="enum" />
                <xs:enumeration value="raw" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
          <xs:attribute name="min" type="xs:string" />
          <xs:attribute name="max" type="xs:string" />
          <xs:attribute name="digit" type="xs:string" />
          <xs:attribute name="inc" type="xs:string" />
          <xs:attribute name="items" type="xs:string" />
          <xs:attribute name="value" type="xs:string" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ResponsibilityType">
    <xs:attribute name="name" type="xs:string" />
    <xs:attribute name="address" type="xs:string" />
    <xs:attribute name="icon" type="xs:string" />
  </xs:complexType>
</xs:schema>
```

3.2 OMG SACM への変換 (Informative)

OMG SACM(Structured Assurance Case Metamodel) [4] は UML の標準化を行っている OMG(Object Management Group)[*3]で標準化されているアシュアランスケースのメタモデルである。もともとは GSN および他の表記法を統合したメタモデルであったが、現在は、エビデンスのメタモデルと統合されている。しかしながら、アシュアランスケースの部分ではモジュールやパターンなどは標準化されていない。さらに、SACM と GSN との対応関係は Normative として定義されておらず、Informative に紹介されているだけである [4]。D-Case では、基本 D-Case 構文と SACM の対応を、[4] の Informative をもとに定義している (表 5)。D-Case の Schema は、OMG SACM への変換が容易であるように定義している。また、**??** 節で導入した撤回可能 GSN は、OMG SACM における *AssertedChallenge* や *AssertedCounterEvidence* の表現を与えていると考えられるが、具体的な対応関係は今後与える予定である。

[*3] http://www.omg.org

表 5　GSN と SACM の対応 [4](Informative)

GSN	SACM
ゴール	*Claim*
コンテキスト	*InformationElement*
サポートリンク	*AssertedInference* もしくは *AsseretedEvidence*
コンテキストリンク	*AsseretedContext*
アンデベロップド	*Claim* ノードの *ToBeSupported* 属性 = `true`

3.2.1　D-Case Editor と Astah/GSN での SACM 変換の違い (Informative)

日本製のアシュアランスケースツールは主に D-Case Editor と Astah/GSN がある。表 6 に両者の SACM 対応の違いを示す。

4　参照実装

D-Case Editor[*4]には D-Case 構文定義書の定義内容が参照実装されている（ただし、2.4 節は除く）。

参考文献

[1] Robin Bloomfield and Peter Bishop. Safety and assurance cases: Past, present and possible future - an Adelard perspective. In *Proceedings of the Eighteenth Safety-Critical Systems Symposium, Bristol, UK*, 2010.

[2] GSN contributors. GSN community standard version 1.0, 2011. `http://www.goalstructuringnotation.info`.

[3] Yutaka Matsuno. A design and implementation of an assurance case language. In *44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, June 23-26, 2014*, pages 630–641, 2014.

[4] Object Management Group (OMG). Structured assurance case metamodel (SACM), 2012. `http://www.omg.org/spec/SACM/`.

[5] Toshinori Takai and Hiroyuki Kido. A supplemental notation of gsn aiming for dealing with changes of assurance cases. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops, Workshop on Open Systems Dependability (WOSD)*, pages 461–466, 2014.

[*4] www.dcase.jp

表 6　D-Case Editor と Astah/GSN での SACM 対応の違い (Informative)

モデル要素	Astar GSN	D-Case Editor (Astar GSNとの差異)
Goal	<argumentElement xsi:type="ARM:Claim" xmi:id="XMIでユニークなID" id="Goal.Id" description="" content="Goal.Statement" assumed=false toBeSupported="true|false"/> ※ assumedはfalseとする。 ※ toBeSupportedはUnDevelopedがtrueの場合はtrue	以下の属性に違いがあります。 xmi:id=Goal.id id=Goal.name content=Goal.desc ※ toBeSupportedは、targetにUndevelopedがある場合trueという意味でしたら同じです。
Strategy	<argumentElement xsi:type="ARM:ArgumentReasoning" xmi:id="XMIでユニークなID" id="Strategy.Id" description="" content="Strategy.Statement" describedInference="接続先とGoalが接続しているAssertedInferenceのxmi:id(スペースをデリミタとして複数指定可)"/> ※ ArgumentReasoningにはtoBeSupportedが指定不可であるため、Undevelopedの情報は出力できない。 ※ サンプルではdescribesであるが、最新のXSDで変更となっている。 ※ 他にdescribedChallenge/structure属性、または同じ名前で子要素(形式はAssertedInferenceと同じ)としても指定可能のようである。 ※ インポートはdescribedInferenceの属性指定と子要素指定の両方対応が必要であると思われるが、子要素がどのように指定される	以下の属性に違いがあります。 xmi:id=Strategy.id id=Strategy.name content=Strategy.desc describedInference=(AssertedInferenceではなく)tagetのGoalのxmi:id
Solution (Evidence,Monitor,Contract,Action)	<argumentElement xsi:type="ARM:InformationElement" xmi:id="XMIでユニークなID" id="Solution.Id" description="" content="Solution.Statement" url=""/> ※ url属性が必須要素であるため空で指定。	以下の属性に違いがあります。(<type>はモデル要素) xmi:id=<type>.id id=<type>.name content=<type>.desc
Context	<argumentElement xsi:type="ARM:InformationElement" xmi:id="XMIでユニークなID" id="Context.Id" description="" content="Context.Statement" url=""/> ※ url属性が必須要素であるため空で指定。	以下の属性に違いがあります。 xmi:id=Context.id id=Context.name content=Context.desc
Justification	<argumentElement xsi:type="ARM:Claim" xmi:id="XMIでユニークなID" id="Justification.Id" description="" content="Justification.statement" assumed=false toBeSupported=false> Goalと同様にClaimに変換する。Goalとの識別は、「InContextOfのソースであること」でできる。JustificationとAssumptionはassumedで識別する assumedはfalse	以下の属性に違いがあります。 xsi:type="ARM:InformationElement" xmi:id=Justification.id id=Justification.name content=Justification.desc assumed未設定 (InformationElementのため) toBeSuppored未設定 (InformationElementのため)
Assumption	<argumentElement xsi:type="ARM:Claim" xmi:id="XMIでユニークなID" id="Assumption.Id" description="" content="Assumption.statement" assumed=true toBeSupported=false> Goalと同様にClaimに変換する。Goalとの識別は、「InContextOfのソースであること」でできる。JustificationとAssumptionはassumedで識別する assumedはtrue	以下の属性に違いがあります。 xsi:type="ARM:InformationElement" xmi:id=Assumption.id id=Assumption.name content=Assumption.desc assumed未設定 (InformationElementのため) toBeSuppored未設定 (InformationElementのため)
Module,External	(ない)	<argumentElement xsi:type="ARM:Claim" xmi:id=<type>.id id=<type>.name content=<type>.desc parameterizedContent=<type>.parameterizedDesc flag=<type>.flag refSource=<type>.refSource attachment=<type>.attachment assumed=true toBeSupported=false /> ※ <type>はモデル要素
Pattern	(ない)	<argumentElement xsi:type="ARM:InformationElement" xmi:id=Pattern.id id=Pattern.name content=Pattern.desc parameterizedContent=Pattern.parameterizedDesc flag=Pattern.flag refSource=Pattern.refSource attachment=Pattern.attachment url="" subType=Pattern.subType leafNode=Pattern.leafNode i=Pattern.i j=Pattern.j parameterDefs=Pattern.parameterDefs parameterVals=Pattern.parameterVals />
SupportedBy (Responsibility,DcaseLink004も)	接続先がGoal/Strategyの場合 <argumentElement xsi:type="ARM:AssertedInference" xmi:id="XMIでユニークなID" id="SupportedBy.Id" source="接続先のxmi:id" target="接続元のGoal(スペースをデリミタとして複数指定可)" description="" content="SupportedBy.Definition"/> ※ Strategyを経由している場合でもtargetはGoalであることに注意。そのため、Goal->Strategy->Goalの場合に出力するAssertedInferenceは1つとなる。 ※ また、その場合に設定するidは、Strategyとsourceを接続しているSupportedByのIDとする。 ※ SupportedByがGoal->Strategyのみで完結している場合のAssertedInferenceは、出力形式が明確ではないため出力しない。 接続先がSolutionの場合 <argumentElement xsi:type="ARM:AssertedEvidence" xmi:id="XMIでユニークなID" id="InContextOf.Id" source="接続先のxmi:id" target="接続元のGoal" description="" content="SupportedBy.Definition"/>	sourceかtargetが、MonitorかActionの場合 (<type>はモデル要素) <argumentElement xsi:type="ARM:AssertedEvidence" xmi:id=<type>.id id=<type>.name content=<type>.desc source="接続元のxmi:id" target="接続先のxmi:id" description="" /> それ以外の場合 (<type>はモデル要素) <argumentElement xsi:type="ARM:AssertedInference" xmi:id=<type>.id id=<type>.name content=<type>.desc source="接続元のxmi:id" target="接続先のxmi:id" description="" /> ※ 上記とはxsi:typeが異なるだけです。
InContextOf	<argumentElement xsi:type="ARM:AssertedContext" xmi:id="XMIでユニークなID" id="InContextOf.Id" source="接続先のxmi:id" target="接続元のGoal or Strategy" description="" content="InContextOf.Definition"/>	以下の属性に違いがあります。 xmi:id=InContextOf.id id=InContextOf.name content=InContextOf.desc