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
KBSE2012-81(2013-3)

アシュアランスケースツールへのプログラミング言語技術の適用

松野　裕†　山本修一郎†

† 名古屋大学 情報連携統括本部 情報戦略室, JST CREST

あらまし　システム保証のための手法・ドキュメントとしてアシュアランスケースが注目を集めている。アシュアランスケースのグラフィカルな表記法としてGSN(Goal Structuring Notation) などが提案されており、いくつかのツールが開発されている。しかしGSNの定義は、曖昧なところが多く、より高度なツールの機能の実装が困難である。本論文では、GSNの標準的な定義であるGSN Community Standardの基本的な形式化を試み、それを元にしたモジュール機能などのツール拡張機能の試験実装を報告する。

キーワード　システム保証、ディペンダビリティ、アシュアランスケース、アシュアランスケースツール、GSN (Goal Structuring Notation)

Applying Techniques of Programming Languages to Assurance Case Tool

Yutaka MATSUNO† and Shuichiro YAMAMOTO†

† Strategy Office, Information and Communications Headquarters, Nagoya University

Abstract　GSN (Goal Structuring Notation) [12] is a graphical notation widely used to construct *assurance cases*, which are required for the system assurance of safety critical systems specially in Europe, and now worldwide as the importance of system assurance has been growing and several safety standard such as ISO 26262 [10] mandates the use of safety case (assurance case for system safety). The syntax and extensions for module and patterns have been defined in GSN Community Standard [8]. In this paper we report our preliminary implementation partly satisfying the syntax in [8]. The prototype implementation has been done on *D-Case Editor*, an Eclipse based assurance case editor. The prototype implementation is also available from the webpage [13]. Among syntax defined in [8], we have implemented "away goal" and "module node" as the basis for the module system, together with parameters with scope and pattern instantiation function which are extensions of our previous work ([14], [15]). Due to some ambiguity in [8], several design choices could exist. In this paper we report our design choice for the prototype implementation.

Key words　System Assurance, Dependability, Assurance Case, Assurance Case Tool, GSN (Goal Structuring Notation)

1. はじめに

様々な分野において、システム保証の重要性が増している。安全性ケース (Safety Case, システムの安全性を議論するためのアシュアランスケース) は、高い安全性を求められるシステム (自動車、鉄道、防衛、原子力発電、油田など) の開発運用の際に認証機関に提出が求められるほどにヨーロッパ、特にイギリスで普及している。安全性ケースの提出を義務付けている規格などとして、EUROCONTROL(EUの航空管制システムに関する規格) [6]、Rail Yellow Book [16] (イギリスの鉄道に関する規格)、MoD Defence Standard 00-56(イギリス国防省の規格) などがある。

アシュアランスケース (Assurance Case, [4]) にはいくつか定義がある。その一つを示す。definitions as follows [1].

> "a documented body of evidence that provides a convincing and valid argument that a system is adequately dependable for a given application in a given environment."

和訳すると、以下になる。

> 与えられた環境と適用先において、システムが十分にディペンダブルであることを、確かで正しい議論を提供する、エビデンスを元にした文書

表記にあたっては、グラフィカルな表記法を用い構造的な記述により、記述や認証などに伴う困難さを軽減する試みがなさ

れている。*Goal Structuring Notation* (GSN) [12] はグラフィカルな表記法の一種である。アシュアランスケースを記述し、再利用性などを高めて、コストを抑えることは、アシュアランスケースを用いる組織にとって重要である。GSN では、パターンや、モジュール、パラメータ付き表現などが提案されている。最近では GSN の基本構文およびパターンやモジュールの基本構造が、GSN Community Standard として発行された [8]。

本論文では [8] への準拠を目指した基礎的な実装を報告する。基礎的な実装は JST CREST DEOS [2] プロジェクトで開発中の D-Case Editor [13] 上で行なっている。D-Case Editor は Eclipse のプラグインとして開発されており、今回開発した拡張も [13] よりダウンロード可能である。[8] で定義されている構文の内、"away goal"および "module node"を、モジュールシステムの基本要素として、またスコープ付きのパラメータ、パターンインスタンス化など、パターンの基本要素として実装した。後者は我々の既存研究 ([14], [15]) の拡張である。しかしながら [8] にはいくつかの曖昧な部分がある。本論文では曖昧な部分を適切に解釈し、実装を行った。

本論文の構成は以下である。2 節において GSN、GSN モジュール、GSN パターンについて、技術的背景を説明する。3 節において、我々の基礎実装を説明する。4 節において [8] で定義されている構文をすべて実装するにあたっての課題を示す。5 節においてまとめる。

2. 技術的背景

2.1 Goal Structuring Notation (GSN)

Goal Structuring Notation (GSN) は York 大学の Tim Kelly らによって提案された、アシュアランスケースのグラフィカルな表記法の一つである [12]。

GSN は安全性ケースを記述するために広く使われている。GSN によって記述された安全性ケースのいくつかは公開されているが、数は多くない ([3])。簡単に GSN を説明する。一つの GSN はいくつかのノードの種類による、木構造をなす。主なノードは、議論する命題であるゴール (goal) ノード、ゴールをサブゴールに分割する説明を行う戦略 (strategy) ノード、ゴールを直接保証するエビデンスへの参照である証拠 (evidence) ノード、そしてゴールや戦略の前提情報をしめすための前提 (context) ノードである。図 1 は GSN の簡単な例である。GSN のルートノードはゴールノードでなくてはならない (トップゴールという)。図 1 では G_1 であり、この GSN で議論すべき命題である。G_1 を議論するための前提として C_1 が G_1 に付与されている。G_1 は戦略ノード S_1 を通してサブゴール G_1, G_2 に分割される。戦略ノードはあるゴールがなぜ、サブゴールが満たされるとき成り立つのか、説明、もしくは理由等を示す。S_1 は、前提ノードにある欠陥 A および B に分けて議論することが示されている。直接の証拠を持つゴールに対してはその証拠を参照する証拠ノード (G_2 に対する E_1) が下につく。ここでは、Fault Tree Analysis (FTA) 解析の結果が用いられている。その他に未達成 (undeveloped) ノードがあり、ゴールを議論するためのサブツリー、証拠ノードが現時点でない場合用いる。ここでは G_3 に対して未達成ノード U_1 が付けられている。図 1 の GSN は D-Case Editor を用いて記述されている。

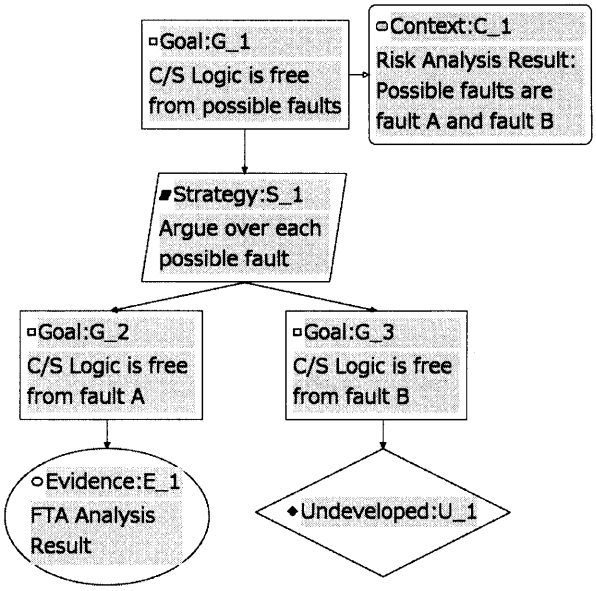

図 1 GSN の例

2.2 GSN モジュール

GSN Community standard [8] では、複数の GSN でノードやサブツリーを共有するためのモジュールシステムが定義されている。

図 2 は GSN モジュールシステムの主な要素であるモジュール参照、away goal, および contract module である。もしあるモジュール A のゴール G が、モジュール B のトップゴールによって (部分的に) 議論されている場合、G に対して A に対応するモジュールノードが付けられる。Away goal は他のモジュール B にあるゴールノード参照するとき、モジュール A にあるゴールを議論する場合に用いられる。さらに、contract module は二つのモジュールの関係を示すときに使われる。GSN community standard では "contract module は二つのモジュールの関係を定義し、あるモジュールにあるゴールがどのようにもう一つのゴールにおける議論を支援するかを定義する" と書かれている。しかしながら contract module に対する記述は [8] には少ない。Contract node が、GSN によって記述されるのか、単に自然言語によるのか明確に書かれていない。

2.3 GSN パターン

一般にアシュアランスケースを記述し、確認することは多くのコスト、各分野ごとの知識が必要になる。それらの困難を軽減するため、いくつかの GSN パターン ([5], [11], [17]) が提案されている。図 3 は GSN パターンの簡単な例である ([5])。トップゴールであるシステムの安全性 ($G1$) は戦略ノード $S1$ を通して、いくつかの機能安全ゴール ($G2$) にまず分けられる。この分割の前提として、Functional Hazard Analysis などにより解析されたすべての安全機能のリストが必要である ($C1$)。さらに、すべての安全機能を個別に議論できることを、それぞれの機能が独立しているか ($G3$)、依存関係がなんらハザードを

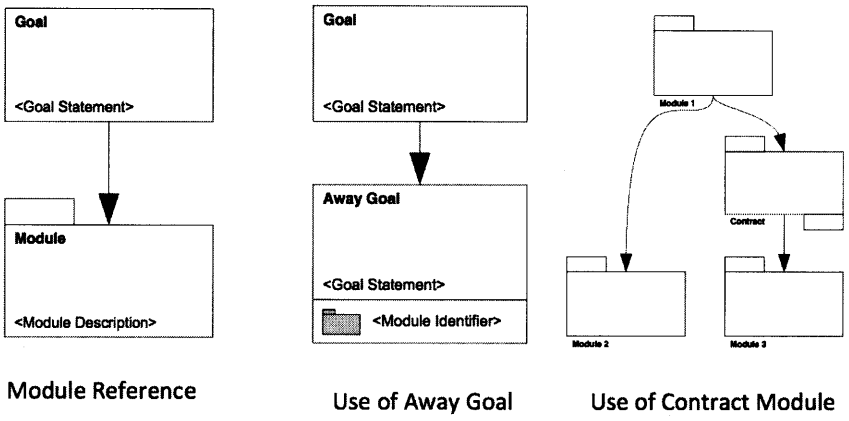

図 2　Main constructs of GSN modules [8]

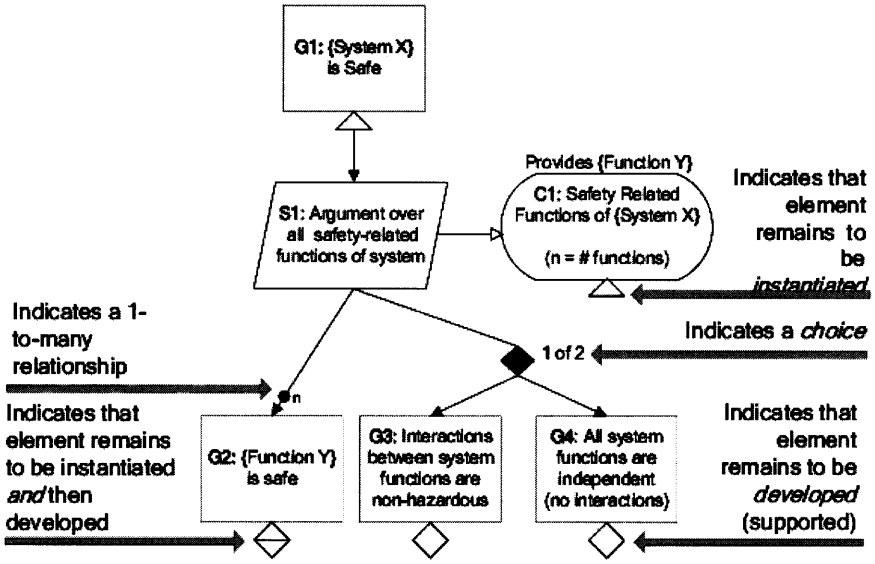

図 3　GSN パターンの例 [5]

生み出さないか ($G4$)、どちらかのゴールを選ぶようになっている。

図 3 は [8] で定義されている GSN パターンの主要な拡張を用いている。Figure 3 includes main GSN extensions for GSN patterns, as defined in [8]:

- パラメータ。{System X} および {Function Y} はパラメータである。X と Y はそれぞれ適当なシステム名、機能名に置き換えられる。
- 未置換。ノードに付けられる △ は、そのノードにまだ具体的な名前に置き換えられていないパラメータがあることを示している。
- 1 to many expressions (multiplicity)。機能の数はシステムによって異なる。機能の数 (ここでは n) ごとに、サブゴールを生成する。
- 選択。$G3$ と $G4$ のように、どちらかを選択する場合に用いる。

3. 基礎実装

3.1 GSN モジュールの基礎実装

基礎実装にあたって、まずひとつのモジュールは、一つのトップゴールを持つ GSN を持つと仮定した。[8] では、モジュールは “モジュールへの参照は一つの argument を指す” と書かれているが、一つの argument が一つのトップゴールを持つ GSN に対応するかは明記されていない。我々の実装では、一つのモジュールは一つの D-Case Editor ファイル (.dcase_diagram ファイル) に対応する。次に、各ゴールノードに public/private 属性をつけた。もしあるゴール G が “public” 属性を持つなら、G は他のモジュールのゴールから参照されうる。これは [8] で定義されている *public indicator* に対応する。

Away goal の例を図 4 に示す。図 4 では、モジュール “NASA” にある GSN はある飛行機のディペンダビリティを議論している。ここではディペンダビリティは安全性とセキュリティとして定義されている。セキュリティに対する議論のために、別のモジュール “nasa2” のトップゴールが参照されている。我々の実装では、参照されている GSN は、参照元の GSN から直接みることができるようになっている。Goal G_8 が away goal である。

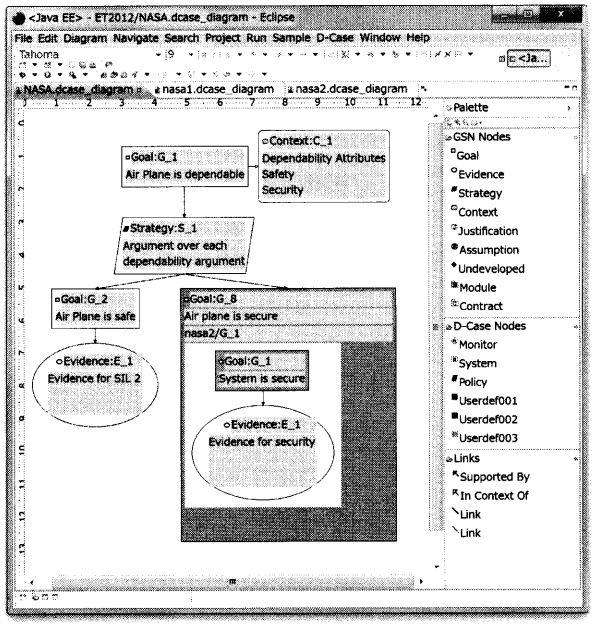

図 4 Away goal の例

モジュール参照には 2 通りあると考えられる。一つは既に存在しているモジュールを参照する場合であり、もう一つはサブモジュールを作る場合である。図 5 はある組込みシステムの CPU 利用率に関する GSN である。一つの GSN は時として大きくなり、管理が大変になることがある。その場合、サブツリーを (サブ) モジュールにするとよいことがある。図 6 は図 5 の右のサブツリーをモジュール化したあとの図である。

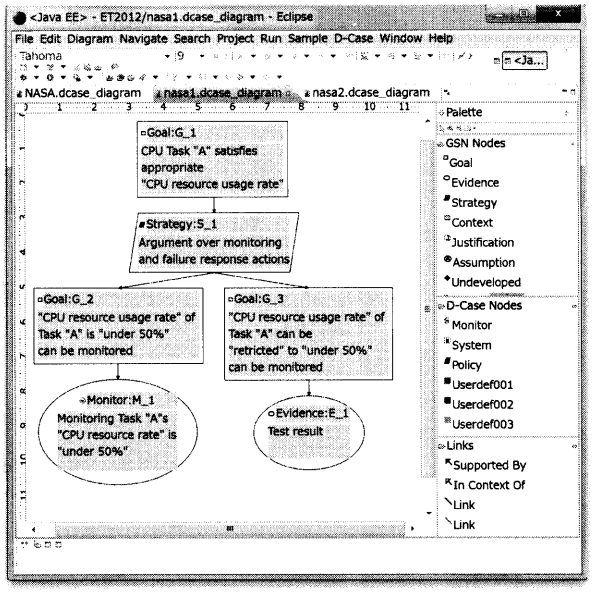

図 5 モジュールの例

3.2 型つきパラメータによるパターン置換

[8] では、パターンにおけるパラメータの説明があまりない。我々は [14] において [8] を部分的な形式化を試み、D-Case Editor 上で型つきのパラメータの実装を行った。しかしながら複数の GSN にまたがるグローバルなパラメータと、ノード内のみのローカルパラメータのみ実装されていた。今回の実装では、サブツリーをスコープに持つローカルパラメータを定義できるように拡張した。

以下の様な設計方針によった。パラメータはゴールにおいて定義される。パラメータのスコープはそのゴールのサブツリーである。パラメータは 5 種類の型を持ちうる。int, double, string, enum および raw 型である。raw 型はは int, double, string, および enum 型以外の型として定義した。

図 4 では、“air plane” は “System” という名前のパラメータの値である。図 7 は “System” パラメータの設定ウインドウのスナップショットである。ここでは “System” パラメータは、“car”, “air plane”, “space shuttle” を値として持ちうる enum 型のパラメータとして定義されている。

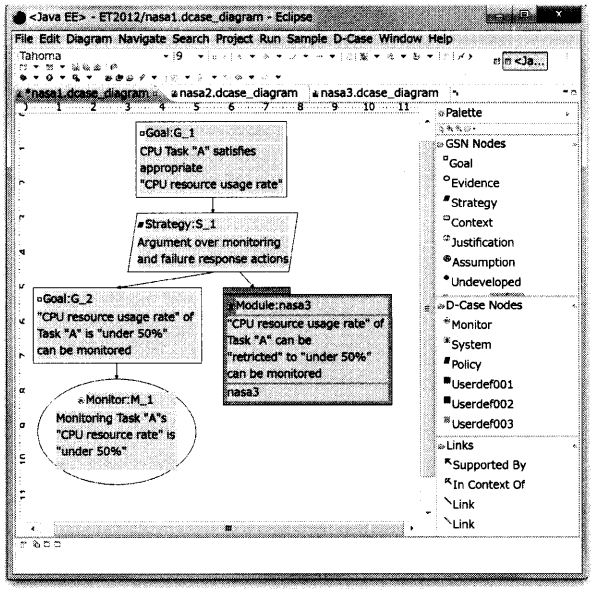

図 6　サブツリーのモジュール化

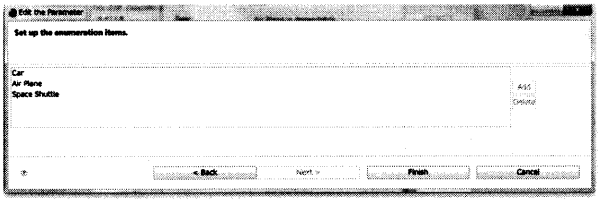

図 7　パラメータ設定ウインドウ

D-Case Editor はパターンライブラリを持っている。ユーザは既存もしくはユーザが定義したパターンをパターンライブラリから選択して利用できる。利用にあたっては、そのパターンのトップゴールに定義されているパラメータに値を代入することによって、パターンを置換する。

4. 今後の課題

これまでに議論してきたように GSN community standard [8] にはいくつか曖昧なところがある。ここでは今後の実装にあたっての課題の一つであるパラメータ情報のモジュール間の伝搬について示す。図 4 において、あるパラメータ “X”がトップゴールで定義されていたとする。このとき、“X”は away goal G_8 を通じて、モジュール nasa2 のゴール G_1 でも使用できるだろうか？そうではないと考える。なぜなら、プログラミング言語の観点から言えば、参照されているモジュール側からは、参照しているモジュールで定義されているパラメータは通常わからないからである。一方、図 6 において、パラメータ “Y” がトップゴールで定義されていたとする。このとき “Y”は右のサブモジュールで使用できるだろうか。この場合は使用出来るべきと考える。なぜならそのサブモジュールはもともと図 6 の GSN の一部だったからである。

もともとは GSN は非常に簡単な表記法として定義された。しかしながら GSN は現在モジュール、パターン、パラメータなどを持ち、一般的なプログラミング言語と同様なレベルまで拡張されている。簡潔さを失わず、プログラミング言語で用いられているようなレベルの形式性で、今後 GSN は定義されていくべきであると考える。

5. ま と め

本論文では GSN community standard [8] の基礎的な実装を示した。いくつかの国際規格、例えば ISO 15026 [9] や OMG SACM (structured assurance case metamodel) [7] ではアシュアランスケースの基本構造や GSN などのもととなる記法のメタモデルなどが定義されている。GSN community standard [8] の内容は、これらの国際標準の中に含まれていくべきである。我々の実装によって ([13] で公開中である)、アシュアランスケースの国際標準化、ツール実装の促進によるアシュアランスケースのより広範囲な普及に貢献出来ればと考えている。現在 GSN standard のその他の定義の実装を行なっている。近いうちに報告したい。

文　献

[1] http://www.adelard.com/web/hnav/ASCE.
[2] http://www.crest-os.jst.go.jp.
[3] http://dependability.cs.virginia.edu.
[4] *Workshop on Assurance Cases: Best Practices,Possible Obstacles, and Future Opportunities, DSN 2004*, 2004.
[5] Robert Alexander, Tim Kelly, Zeshan Kurd, and John McDermid. Safety cases for advanced control software: Safety case patterns. Technical report, Department of Computer Science, University of York, 2007.
[6] European Organisation for the Safety of Air Navigation. Safety case development manual. European Air Traffic Management, 2006.

[7] OMG System Assurance Task Force. http://sysa.omg.org.
[8] GSN contributors. GSN community standard version 1.0, 2011.
[9] ISO. ISO/IEC15026-2: 2010, systems and software engineering-systems and software assurance-part 2: Assurance case, 2010.
[10] ISO. ISO 26262 road vehicle - functional safety -, part 1 to part 10, 2011.
[11] Tim Kelly and John McDermid. Safety case patterns - reusing successful arguments. In *IEE Colloquium on Understanding Patterns and Their Application to System Engineering*, 1998.
[12] 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.
[13] Yutaka Matsuno. http://www.dependable-os.net/tech/D-CaseEditor/.
[14] Yutaka Matsuno and Kenji Taguchi. Parameterised argument structure for GSN patterns. In *Proc. IEEE 11th International Conference on Quality Software (QSIC 2011)*, pages 96–101, 2011.
[15] Yutaka Matsuno, Hiroki Takamura, and Yutaka Ishikawa. A dependability case editor with pattern library. In *Procs. IEEE 12th International Symposium on High-Assurance Systems Engineering (HASE)*, pages 170–171, 2010.
[16] Railtrack. Yellow book 3. Engineering Safety Management Issue3, Vol. 1, Vol. 2, 2000.
[17] Robert Andrew Weaver. *The Safety of Software - Constructing and Assuring Arguments.* PhD thesis, Department of Computer Science, University of York, 2003.