D-Case Editor の機能拡充に関する開発
機能仕様書

31/JAN/2013

AXE, Inc.

改訂履歴

更新日	版	内容	担当
31/JAN/2013	0.7	• パターンが単一のツリーかどうかのチェックを、メニューの表示段階で行うよう変更(4.4) • Desc 属性に設定される旨を追記(5.4) • クリップボードにコピーする機能を追加(5.5)	臼田@AXE
18/JAN/2013	0.6	• モジュールビューにノード数を追加(3.8) • モジュールの参照の同期をとる処理を追加(3.8) • 「3.9 ノード一覧の出力」を追加 • 「3.10 日本語化」を追加 • パターンは、モジュールではなくそのまま追加するよう変更(4) • パラメータ定義時にデフォルト値を設定(5.2) • 「5.5 パラメーター覧の表示」を追加	臼田@AXE
28/DEC/2012	0.5	• main もモジュールとして扱うよう記述を変更(3.2～3.4) • モジュールの Refresh 機能は不要のため削除(3.7)	臼田@AXE
11/OCT/2012	0.4	• 「3 モジュール」のファイルの格納場所を変更 • 「3.3.2 既存ノードのモジュール化」で、ルートノードを選択してモジュール化する手順に変更 • 「4 パターン」を単一のファイル前提として各処理を変更	臼田@AXE
5/OCT/2012	0.3	• 「3 モジュール」の構成を変更 • 「3.2 モジュールの構造」を追加 • Module ノードおよび Away Goal ノードに、参照先のパスの表示を追加(3.4) • ダイアグラム全体のモジュール内再表示を追加(3.7) • 「3.8 モジュールの管理」を追加 • 「4.3 パターンの構造」を追加 • 「4.4.3 パターンの追加処理」に詳細の手順を追加	臼田@AXE
2/OCT/2012	0.2	新規作成(ドラフト)	臼田@AXE

目次

1 はじめに

1.1 概要

本書は、独立行政法人 科学技術推進機構(以下、JST)が行う戦略的創造研究推進事業の研究領域である「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」(以下、DEOS プロジェクト)において開発中の、Dependability cases(以下、D-Case)の作成を支援するツール「D-Case Editor」の拡張機能に関する基本設計書である。

1.2 用語の定義

名 称	意 味
ダイアグラム	(D-Case を)図表化したもの。
D-Case 文書	D-Case およびモジュールを表すファイル。下記の GMF ダイアグラム情報ファイルおよび GMF モデル情報ファイルの 2 ファイルからなる。
GMF ダイアグラム情報ファイル	D-Case もしくはモジュールの各要素(ノードやリンク)の位置や大きさ、色などの情報を表すファイル。 「D-Case 名もしくはモジュール名.dcase_diagram」のファイル名で記録される。
GMF モデル情報ファイル	D-Case もしくはモジュールの各要素の論理的な構造を表すファイル。 「D-Case 名もしくはモジュール名.dcase_model」のファイル名で記録される。
属性	ノードやリンクがそれぞれ固有に持つ性質。 Name, Desc, Attachment, Userdef001～016 などがある。
パレットビューア	Eclipse 上で動作するグラフィカルエディタ(D-Case Editor)内で、ノードやリンクを選択するツールを提供する部分。
ビュー	Eclipse 内で何らかの情報を提供する部分。通常はタブ形式でいずれか 1 つの情報が表示されている。
プリファレンスストア	Eclipse でプラグインの設定を保存するための領域。ワークスペース(作業領域)内に設けられ、プラグイン毎に分けて記録される。

1.3 関連文書

- 発注仕様書
- D-Case Editor 機能仕様書
- D-CASE エディタ データ仕様書
- D-Case 入門
- GSN COMMUNITY STANDARD VERSION1

  http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf
- Eclipse3.4 プラグイン開発 徹底攻略 毎日コミュニケーションズ (ISBN978-4-8399-2972-5)
- Eclipse Documentation

  http://www.eclipse.org/documentation/

- Graphical Modeling Framework – Eclipsepedia
  http://wiki.eclipse.org/Graphical_Modeling_Framework
- Eclipse Modeling Framework – Eclipsepedia
  http://wiki.eclipse.org/Eclipse_Modeling_Framework
- Graphical Editing Framework – Eclipsepedia
  http://wiki.eclipse.org/GEF
- Java Platform Standard Edition 7 Documentation
  http://docs.oracle.com/javase/7/docs/index.html

2　拡張機能概要

D-Case Editor は、DEOS プロジェクトにおいて開発中の、D-Case の作成を支援するためのツールである。
オープンソースの統合開発環境である「Eclipse」のプラグインであり、基本的な機能はすでに実装されている。
しかし、企業などで実際に開発されているシステムの構造は複雑であり、記述される D-Case は大きくなる傾向がある。そのため、実際に本ツールを利用する際には、以下の機能が必要となる。

- 大規模な D-Case を単一のダイアグラムで管理することが困難なとき、モジュールとして分割することで、ダイアグラムを統合・簡素化し、管理しやすくする
- 優れた部分 D-Case をパターンとして登録し、パターンを利用することで、D-Case の記述を行いやすくする
- パラメータのスコープをより柔軟にすることで、D-Case の汎用化を促進し、パターンの再利用性を高める

上記のそれぞれの拡張機能の外部仕様を、以降で述べる。

3 モジュール

3.1 概要

大規模な D-Case を効率的に管理するため、モジュール機能を追加する。あるノードをルートとするサブツリーを「モジュール」というひとつのまとまりにして、単一の「Module ノード」に置き換えられるようにする。Module ノードは、通常では単一のノードで表示されるが、展開することによりモジュール内部のノードを表示し、確認することもできる。モジュールは D-Case 文書として保存し、参照元の D-Case とは別のダイアグラムで編集する。

また、モジュールのルートノード以外を参照する「Away Goal」を実現するため、各ノードに public/private 属性を設ける。Away Goal は、public 属性を持つノードのみ参照可能とする。

3.2 モジュールの構造

3.2.1 ファイルの格納場所

モジュールは、従来の D-Case と同様、GMF ダイアグラム情報ファイル(dcase_diagram)および GMF モデル情報ファイル(dcase_model)の形式で保存される。モジュールのファイル名は、「モジュール名.dcase_diagram」および「モジュール名.dcase_model」とし、参照元のノードを含む D-Case 文書が格納されるフォルダに格納する(図 1)。

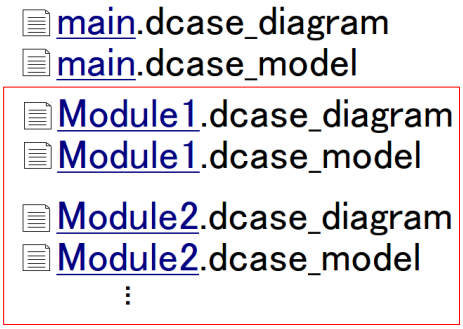

図 1: D-Case 文書の保存場所

このため、D-Case は、あるフォルダ直下にある D-Case 文書群からなる。

また、トップのノードを含む D-Case 名を「main」とし、ファイル名を「main.dcase_diagram」および「main.dcase_model」とする。ただし、main もモジュールの一種として扱う。

3.2.2 モジュールの参照

Module ノードおよび Away Goal ノードが参照するモジュールやモジュール内のノードは、属性「Attachment」に記録する。また、参照されるモジュールやノードは、参照元のノードを属性「Userdef011」に記録する。図 2 に例を示す。

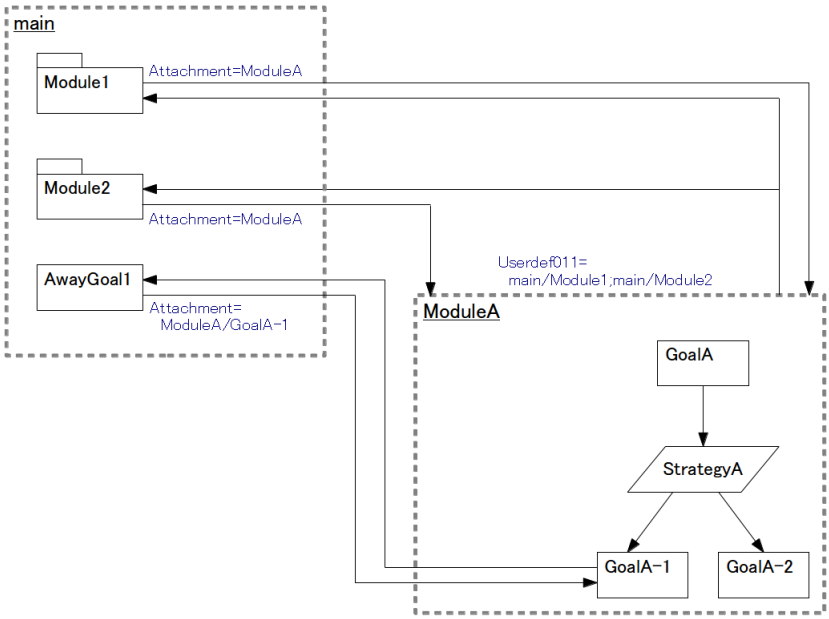

図 2: ノードとモジュールの参照の例

3.3 モジュールの作成

3.3.1 Module ノードの作成

Module ノードを作成するには、他のノードと同様、パレットビューアの「GSN Nodes」にある「Module」を選択し、ダイアグラム上で設置場所を指定する。(モジュールを指定する手順は 3.4.1 に記載)

3.3.2 既存ノードのモジュール化

ダイアグラムにある複数のノードをモジュール化する手順は、以下の通りである。

モジュール化するサブツリーのルートノードを選択して右クリックしたとき、コンテキストメニューに「Create Module」を表示する(図 3)。これを選択したとき、以下の処理を行う。

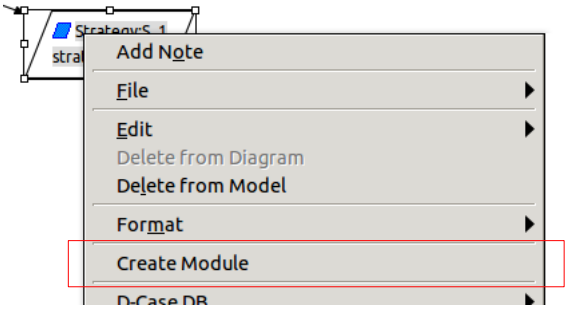

図 3: モジュール化のためのメニュー

1. 選択されたノードをルートとするサブツリーをモジュール化の対象とするため、選択状態にする。

   ルートノードから再帰的に子孫ノードに向かうリンクをたどり、ルートノードを含めた子孫ノードを選択状態にする。

2. 1.で選択状態にしたサブツリーがループになる(子孫ノードをたどる過程でルートノードに到達する)場合、エ

ラーログに以下を出力して、モジュール化の処理を終了する。

```
Not a single tree.
```

3. モジュール名を入力するためのダイアログを表示し(図 4)、モジュール名を得る。キャンセルボタンが押された場合は、モジュール化の処理を終了する。

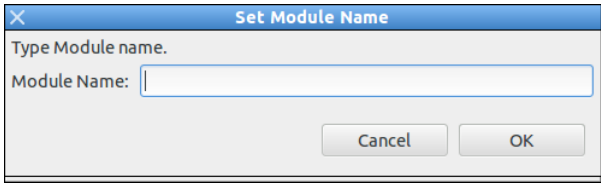

図 4: モジュール名設定ダイアログ

すでに同名の D-Case 文書が存在する場合は、再度ダイアログを表示する。

4. 1.で選択された単一のサブツリーの内容の D-Case 文書を、現在編集中のダイアグラムと同フォルダ(以下、カレントフォルダ)に、3.で入力されたモジュール名を用いた以下のファイル名で保存する。

   モジュール名.dcase_diagram

   モジュール名.dcase_model

   ファイルシステムの問題などにより上記ファイルを作成できない場合は、下記のエラーログを出力して、モジュール化の処理を終了する。

   モジュール名: create failed.

5. 1.で選択したサブツリーを、現在のダイアログから削除する。

   ルートノード以外で、サブツリー以外から参照されているノードがある場合は、そのノードを Away Goal ノードとして残す。Away Goal ノードに関しては、3.4.2 を参照されたい。

6. Module ノードを追加する。Name 属性および Attachment 属性に、3.で入力したモジュール名を設定する。また、1.のサブツリーがあるノードの子ノードだった場合は、Module ノードをそのノードの子ノードとする。

7. 4.で生成したモジュールの Argument の属性「Userdef011」に、参照元の Module ノードのパスを設定する。

   モジュール名/Module ノード名

3.4 モジュールの参照

3.4.1 Module ノードの参照先の指定

既存の Module ノードの参照先モジュールを設定する手順は、以下の通りである。

Module ノードを選択して右クリックしたときに表示されるコンテキストメニューの「Attachment」に、「Select from Module...」を表示する(図 5)。

図 5: モジュールを選択するためのメニュー

これを選択すると、カレントフォルダにある D-Case 文書から、モジュール名一覧を作成し、メニューとして表示する。これらのいずれかを選択すると、Attachment 属性に、選択したモジュール名を設定する。

また、選択したモジュールの Argument の属性「Userdef011」に、参照元の Module ノードとなる下記を設定する。

すでに設定されている(別のノードから参照されている)場合は、「;」を追加した上で下記を追加設定する。

モジュール名/Module ノード名　　(新規の場合)
既存の設定;モジュール名/Module ノード名　　(設定されている場合)

そして、Module ノード内にモジュール名を表示する(図 6)。

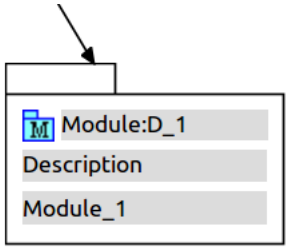

図 6: Module ノードの表示

3.4.2 Away Goal

Away Goal は、モジュール内のルートでないノードを参照するためのノードである。

D-Case Editor では、モジュール内のノードを参照する Goal ノードを、Away Goal ノードとして扱う。

既存の Goal ノードの参照先ノードを設定し、Away Goal ノードとする手順は、以下の通りである。

3.4.1 の場合と同様、Goal ノードを選択して右クリックしたときに表示されるコンテキストメニューの「Attachment」に、「Select from Module...」を表示する(図 5)。これを選択すると、カレントフォルダにある D-Case 文書から、public 属性を持つノード名の一覧を作成し、「モジュール名/ノード名」の形式でメニューとして表示する。これらのいずれかを選択すると、Attachment 属性に、選択したノードのパスを設定する。パスの形式は以下の通りである。

モジュール名/ノード名

また、選択したノードの Argument の属性「Userdef011」に、参照元の Away Goal ノードとなる下記を設定する。
すでに設定されている場合は、「;」を追加した上で下記を追加設定する。

モジュール名/Away Goal ノード名　　（新規の場合）
既存の設定;モジュール名/Away Goal ノード名　　（設定されている場合）

Away Goal ノードの表示を、通常の Goal ノードと区別するため、図 7 のように緑の背景色とする。

また、Module ノードと同様、Attachment 属性の内容をノード内に表示する。

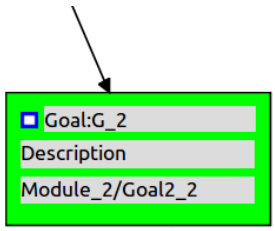

図 7: Away Goal ノード

同様に、Away Goal ノードから参照されているノードの表示を、図 8 のように薄緑の背景色とする。

図 8: Away Goal から参照されたノード

3.5 public/private 属性

public/private 属性を設定する手順は、以下の通りである。

ノードを右クリックしたとき、コンテキストメニューに「Set Flags」を表示する(図 9)。これを選択すると、「Private」および「Public」を表示する。これらを選択すると、ノードに public/private 属性を設定する。

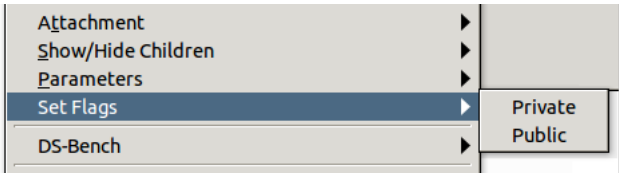

図 9: public/private 属性設定のためのメニュー

public/private 属性は、ノード属性「Userdef010」を使用して実現する。Userdef010 属性に「P」の文字が含まれている場合は public、含まれていない場合は private とする。デフォルトでは private が設定されているとする。

このため、すべてのノードにおいて、「Properties」ビューの Userdef010 の属性の表示を、「Flags」とする。

3.6 モジュールの解除

Module ノードを削除し、モジュールのツリーを直接展開する手順は、以下の通りである。

Module ノードを選択して右クリックしたとき、コンテキストメニューに「Restore Module」を表示する(図 10)。これを選択したとき、以下の処理を行う。

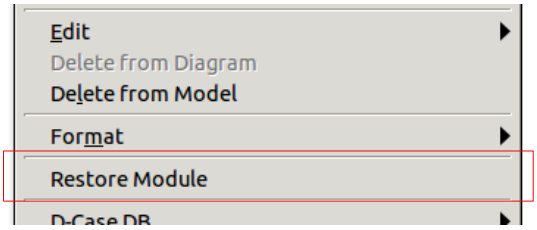

図 10: モジュール解除のためのメニュー

1. Module ノードを削除する。
2. モジュール化されていたツリーを、Module ノードが存在した箇所に(子ノードだった場合は親ノードの下に、ダイアグラム直下だった場合はダイアグラムに)追加する。
3. モジュールの Argument の属性「Userdef007」および「Userdef009」に設定された値を、モジュールのルートノードのそれぞれの属性値にマージする。ルートノードに同名のパラメータが設定されている場合は、そちらを優先する。(パラメータの詳細については 5 章に記載)
4. モジュールの Argument の Userdef011 属性から、削除した Module ノードのパスを削除する。
5. 2.で追加したツリーを選択状態とする。

なお、ここでは、参照先モジュールの D-Case 文書の削除は行わない。(削除の手順は 3.8 に記載)

3.7　モジュールの展開表示

Module ノードおよび Away Goal ノードが参照するモジュールの内容(サムネイル)を、そのノードの中に表示する(あるいは表示をやめる)手順は、以下の通りである。

Module ノードを右クリックしたとき、コンテキストメニューに「Show/Hide Module」を表示する(図 11)。これを選択すると、「Show Module」および「Hide Module」を表示する。

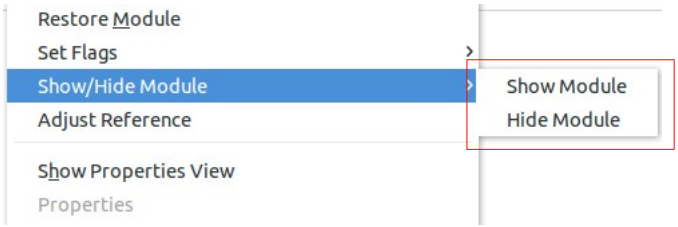

図 11: モジュールを展開するためのメニュー

「Show Module」を選択すると、Module ノードの中にモジュールの内容を表示する(図 12)。すでにモジュールの内容を表示している場合は、何も実施しない。

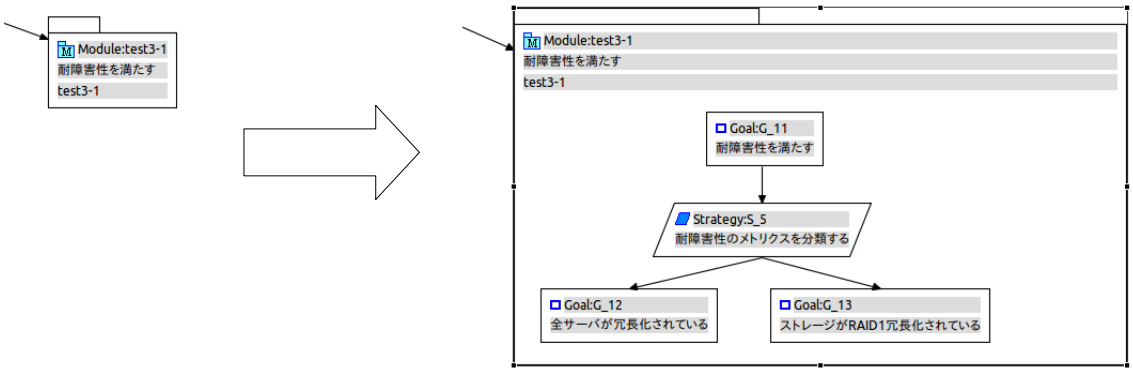

図 12: モジュールの展開

「Hide Module」を選択すると、モジュールの内容を表示している場合は、もとの Module ノードの表示に戻す。

Away Goal ノードの場合も、同様にコンテキストメニューを表示し、同様に表示・非表示の処理を行う。ただし、「Show Module」を選択したときには、参照先のノードをルートノードとするサブツリーを表示することになる。

3.8　モジュールの管理

「Modules」ビューを表示してモジュールの管理を行う手順は、以下の通りである。

「Window」メニューの「Show View」で「Other...」を選択し、「D-Case Editor」を展開すると、「Modules」が表示される(図 13)。

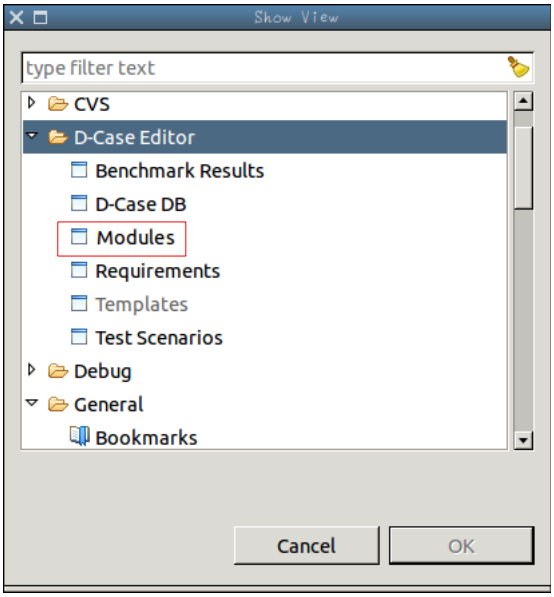

図 13: モジュールビュー表示の選択

これを選択すると、Modules ビューを表示する(図 14)。

Modules ビューでは、現在表示されているダイアグラムが属する D-Case のモジュールおよび public ノードと、モジュール内のノード数、それぞれのリンク数(Userdef011 属性の「;」の個数+1)、参照元のノード(Userdef011 属性値)を表形式で表示する。

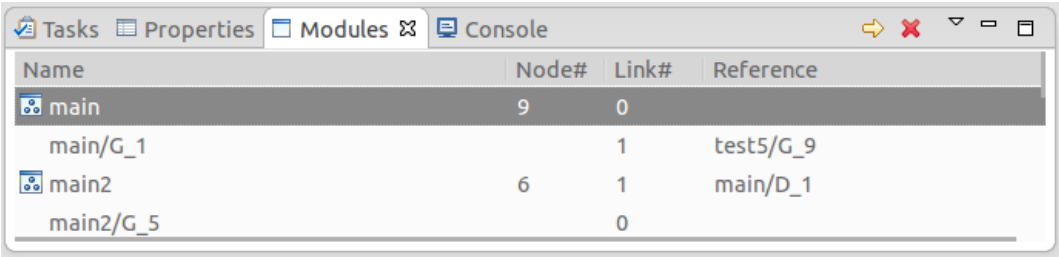

図 14: Modules ビュー

ここで、モジュールを選択して、右上にあるツールバーの FORWARD アイコン(矢印)を押下すると、指定したモジュールをオープンする。また、モジュールを選択して、ツールバーの DELETE アイコン(×印)を押下すると、モジュールおよび public ノードのリンク数がいずれも 0 の場合に、指定したモジュールの D-Case 文書を削除する。

また、属性を直接編集するなどして、モジュール間の参照が双方向にならなくなることがある。

これを修正するには、Attachment を右クリックしたとき、コンテキストメニューに「Adjust Reference」を表示する(図 15)。これを選択すると、以下の処理を行う。

1. モジュールもしくは public ノードが Argument に設定されていて、モジュールもしくは public ノードがすでに存在しない場合、Attachment の設定を削除する。
2. モジュールもしくは public ノードが Argument に設定されていて、参照されている Argument もしくは public ノードの Userdef011 にそのノードが設定されていない場合、Attachment の設定を削除する。

3. Userdef011 に登録されている各ノードに対して、そのノードがすでに存在しない場合、そのノードの設定を削除する。
4. Userdef011 に登録されている各ノードに対して、そのノードの Attachment に自ノードが登録されていない場合、そのノードの設定を削除する。

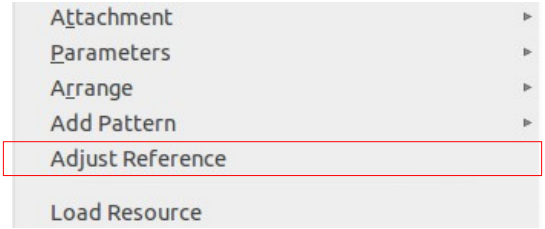

図 15: モジュールの参照を同期させるためのメニュー

3.9 ノード一覧の出力

モジュールにどのようなノードが含まれているか確認できるよう、ノードの一覧をテキストファイルに出力する手順は、以下の通りである。

「D-Case」メニューの「Convert D-Case File」を選択したとき、「Convert GMF To Text」を表示する(図 16)。

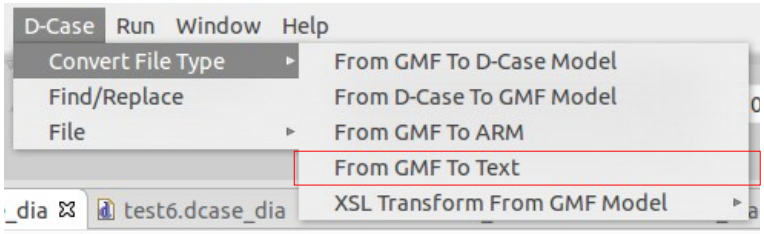

図 16: ノード一覧を出力するためのメニュー

これを選択すると、テキストファイルに出力するためのウィザードを表示する(図 17)。

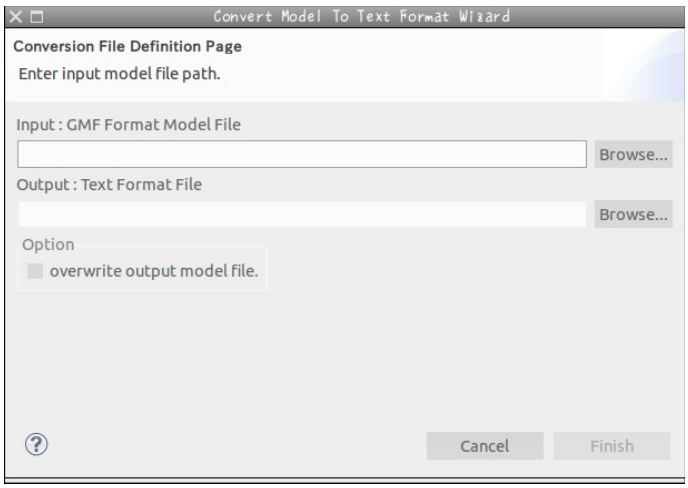

図 17: ノード一覧をテキストファイルに出力するためのウィザード画面

ここで、入力となる GMF モデル情報ファイルと、出力となるテキストファイルを入力し、「Finish」ボタンを押下すると、ノードの種類ごとに分けた、下記のフォーマットのテキストファイルを出力する。

[ノード名]

“Name”,"Desc","Attachment"

…

[ノード名]

“Name”,"Desc","Attachment"

…

3.10 日本語化

Eclipse プラグインの国際化機能を使用して、ノード名の日本語化を行う。

ノード名は、下記のように日本語に変換する。

ノード名(英語)	ノード名(日本語)
Goal	ゴール
Context	前提
Strategy	戦略
Evidence	証拠
Justification	正当化
Assumption	仮定
Undeveloped	未達成
Module	モジュール
Contract	契約
Monitor	モニタ
Policy	ポリシー

3.11 その他

下記については、Module ノードを単一のノードとして扱うことで、基本的には現在の D-Case Editor と同等に処理を行うこととする。

3.11.1 差分表示

Module ノードは単一のノードとして扱い、モジュールの内部は比較対象としない。

3.11.2 ダイアグラム印刷

現在表示されている状態で印刷される。(モジュールを展開している場合は、展開した状態で印刷される。)

3.11.3 自動整列

Module ノードは、他のノードと同様に扱われる。

3.11.4 検索・置換

Module ノードは単一のノードとして扱うため、モジュールの内部は検索対象としない。

4 パターン

4.1 概要

現在の D-Case Editor には、「テンプレート」を追加する機能が実装されている。ただし、現在のテンプレート機能では、ダイアグラムのどの位置に追加されるかがわからない。これを、ダイアグラム上の所望する位置に追加できるようにする。さらに、指定したノードの子ノードとして追加できるようにもする。

4.2 表記の変更

他の文献では、当機能を「パターン」と呼ぶことが多いため、今後は「パターン」という表記で統一する。そのため、下記で「Template」と記載されている箇所を、「Pattern」に変更する。

ただし、パターンが格納されているプロジェクトの名称「D-CaseTemplate」は、混乱を避けるため、変更しない。

- テンプレートビューの名称

  (Windows メニューの「Show View」で表示されるビューの名称(Templates)を含む)
- Templates ビューのプルダウンメニュー内の「Add Template」
- ノードを右クリックしたコンテキストメニュー「Add Child」内の「Add a Template」

4.3 パターンの構造

パターンは、単一のツリーであり、かつ単一の D-Case 文書であることを前提としている。そのため、内部に Module ノードおよび Away Goal ノードを含まない。

4.4 パターンの追加

4.4.1 ダイアグラムへの追加

パターンビュー(旧テンプレートビュー)からパターンを追加する際の処理は、現在と同様である。ただし、パラメータの処理があるため、4.4.3 の処理は実施する。

ダイアグラム上の指定した位置に追加する手順は、以下の通りである。

ダイアグラムを右クリックしたとき、コンテキストメニューに「Add Pattern」を表示する(図 18)。ここで、登録されているパターンのうち、単一のツリーのものをコンテキストメニューに一覧表示する。パターンのいずれかを選択すると、右クリックした位置にパターンを追加する(詳細は 4.4.3 に記載)。

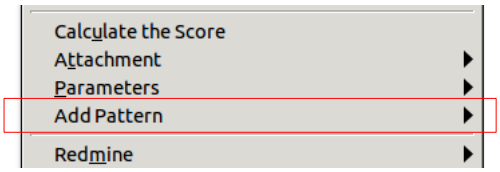

図 18: ダイアグラムにパターンを追加するためのメニュー

4.4.2　ノードへの追加

指定したノードの下にパターンを追加する手順は、以下の通りである。

ノードを選択して右クリックしたときに表示されるコンテキストメニューの「Add Child」に、メニュー「Add Pattern to node」を表示する(図 19)。このとき、接続可能なパターンを候補として表示する。接続可能かどうかの判断は、現在の子ノードの候補の補完機能と同じ手順で行う。(パターンが単一のツリーであり、ルートノードが、選択されたノードの子ノードとして接続可能かどうかで判断する。)

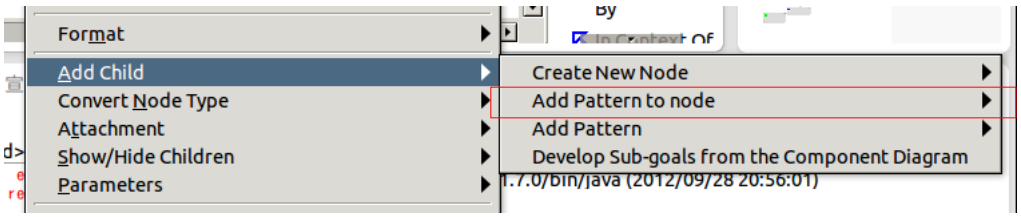

図 19: ノードにパターンを追加するためのメニュー

パターンを選択すると、ノードにパターンを追加する(詳細は 4.4.3 に記載)。

ノードを選択して右クリックし、コンテキストメニューの「Add Child」→「Add Pattern」を選択した際の処理は、現在と同様である。

4.4.3　パターンの追加処理

パターンの追加処理手順の詳細は、以下の通りである。

1. 選択されたパターンが単一のツリーであることを確認する。そうでない場合は、エラーログに下記を出力し、パターン追加処理を終了する。

   パターン名: not a single tree.

2. パターンの Argument およびルートノードで定義されているパラメータが、パターン内の各ノードの「Desc Format String」および「Script」属性内で使用されているか確認する。使用されている場合は、現在と同様、パラメータ設定ダイアログを表示し、得られたパラメータ値を使用する。各ダイアログで「キャンセル」ボタンが押された場合は、パターン追加処理を終了する。ルートノード以外のノードで定義されているパラメータについては、設定されているパラメータ値をそのまま使用する。
3. 4.4.1 や 4.4.2 で指定された箇所に、パターンを追加する。4.4.2 の場合は、選択したノードの子ノードとしてパターンを追加する。

5 パラメータ

5.1 概要

現在の D-Case Editor では、パラメータの定義は workspace 全体で共通、パラメータの値はグローバルもしくは各ノードでのみ参照可能となっている。

モジュールおよびパターンの機能を追加するにあたり、パラメータのスコープを、各ノードをルートとするサブツリー単位に設定できるよう、機能を拡張する。そのため、現在はプリファレンスストアにパラメータ定義を格納しているが、本機能拡張ではノードの属性に格納する。

また、現在は、ノードにパラメータ値を設定していても、Argument で設定されたグローバルパラメータがある場合、後者が参照される。これを、自ノードか、もっとも近い先祖ノードで設定されたパラメータ値を参照するようにする。

5.2 パラメータ定義の設定

指定したノードでパラメータを定義する手順は、以下の通りである。

ノードもしくは Argument を選択して右クリックしたときに表示されるコンテキストメニューの「Parameters」に、メニュー「Define Parameters」を追加する(図 20)。これを選択すると、パラメータの定義を行うためのダイアログを表示する(図 21)。

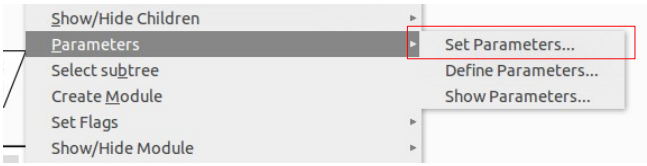

図 20: パラメータ定義のためのメニュー

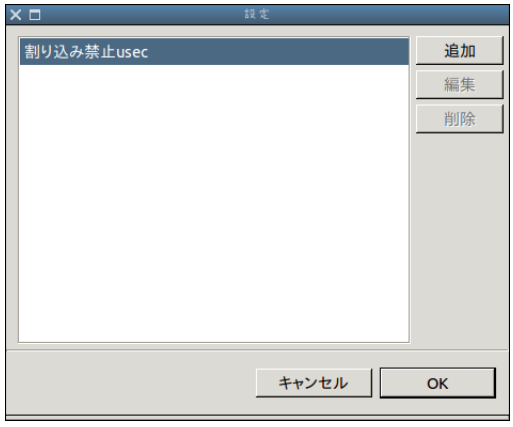

図 21: パラメータ定義を行うダイアログ

図 21 のダイアログでは、現在のパラメータの定義を行う画面と同様に、パラメータ定義の追加、編集および削除を行うことができる。また、現在はパラメータ定義の XML ファイルの Import/Export 機能があるが、本機能拡張ではダイアグラムファイルにパラメータ定義が格納されるため、Import/Export ボタンは設置しない。

パラメータ定義を追加した場合、デフォルトのパラメータ値を設定する。

string および raw の場合はサイズが 0 の値、int および double の場合は下限値、enum の場合は最初の項目を設定する。

なお、現在は、Windows メニューの「Preference」→「D-Case Diagram」→「Parameters」でパラメータの定義を行っているが、前述の通り、各ノードで定義を行うようにするため、これを廃止する。

5.3 パラメータ値の設定

現在の D-Case Editor と同様、ノードに対して、パラメータ値の設定を行う。

グローバルパラメータがなくなるため、ノードを選択して右クリックしたときに表示されるコンテキストメニューの「Parameters」から、「Configure Parameters...」を削除する。

同メニューの「Set Parameters...」では、選択したノードで定義されているパラメータ名と値を表示して、パラメータ値を変更できるようにする。

Argument の場合も同様に処理する。グローバルパラメータではなくなるため、現在のコンテキストメニューに表示される「Configure Global Parameters...」を削除し、「Set Global Parameters...」の表示を「Set Parameters...」に変更する。同様に、「Properties」ビューの「Userdef007」属性の表示を、「Global Parameters」から「Parameters」に変更する。

5.4 パラメータ値の参照

「Desc Format String」属性内で使用されているパラメータを、以下の順に検索し、設定されたパラメータ値に置換して「Desc」属性に設定する。

1. パラメータ値が自ノードで設定されている場合、その値に置換する。
2. 親ノードを順にたどり、設定されているパラメータ値があれば、その値に置換する。

   親ノードが複数ある場合は、リンクの生成順にたどる。

   現在のダイアグラムを参照する Module ノードがある(Argument の Userdef011 属性が設定されている)場合は、そのノードをたどる。
3. 最上位のダイアグラムの Argument に達し、いずれのノードにもパラメータ値が設定されていなかった場合は、エラーログに下記メッセージを出力する。

   ノード名: cannot found パラメータ名.

5.5 パラメータの一覧表示

指定したノードで使用可能なパラメータを表示する手順は、以下の通りである。

ノードもしくは Argument を選択して右クリックしたときに表示されるコンテキストメニューの「Parameters」に、メニュー「Show Parameters」を追加する(図 20)。これを選択すると、パラメータの一覧を確認するためのダイアログを表示する(図 22)。

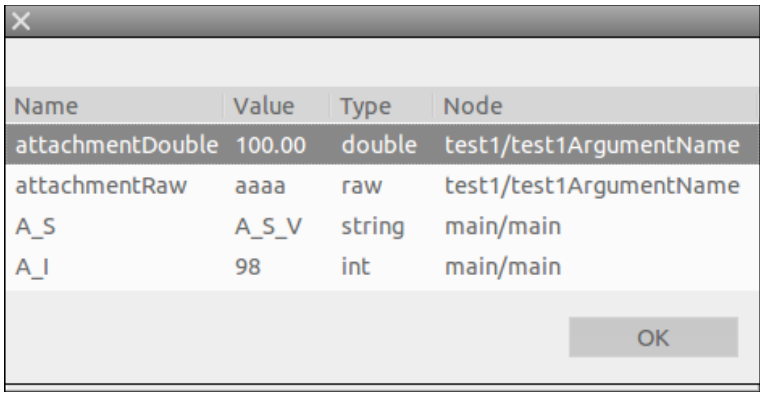

図 22: パラメーター覧ダイアログ

このダイアログでは、パラメータの名前、値、型および設定されているノード(モジュール名/ノード名の形式)を表形式で表示する。表の行をクリックすると「パラメータ名」を、ダブルクリックすると「{パラメータ名}」をクリップボードにコピーする。

5.6 パラメータ定義および値の格納

パラメータの定義は、ノード属性「Userdef009」に、以下の形式で格納する。

パラメータ名1,パラメータ名2,...;パラメータ名1の属性;パラメータ名2の属性;...

パラメータ名の使用禁止文字および名称は、現在の D-Case Editor と同じである。

パラメータの属性は、以下の形式で格納する。基本的には、プリファレンスストアに格納されている形式と同等である。

型	形 式
string	type=string,min=文字数の下限,max=文字数の上限
raw	type=raw,min=文字数の下限,max=文字数の上限
int	type=int,min=下限,max=上限
double	type=double,min=下限,max=上限,digit=小数点以下の桁数,inc=増減幅
enum	type=enum,items={項目1,項目2,...}

パラメータ値は、現在と同様、ノード属性「Userdef007」に、現在の形式で格納する。

このため、すべてのノードにおいて、「Properties」ビューの Userdef009 の属性の表示を、「Parameter Definition」とする。