2, rue du Marché - 92160 Antony
tél et fax : 01 46 74 54 10, e-mail : ariste@wanadoo.fr

MANUEL D'UTILISATION DE FIGAROIDE

	Nom	Visa	Date
Rédacteur	Mme Anne Flori		3 mai 2016
Approbateur	M. Jean Christophe Houdebine		3 mai 2016

Révision du 13 février 2018

TABLEAU DE SUIVI DES MODIFICATIONS

Version	Date	Description de la modification	Rédigé par	Approuvé par
V1.0	03/05/16	Première version	Mme A Flori	M. Jean Christophe Houdebine
V2.0	29/06/16	Version adaptée à la version 1.0.1.2 de FigaroIDE	Mme A Flori	M. Jean Christophe Houdebine
V2.1	13/02/18	Améliorations de détail	M. Bouissou	

SOMMAIRE

1 INTRODUCTION

Ce manuel suit la logique de construction d'une base de connaissances. Le fonctionnement décrit ici, correspond à la version 1.0.1.2 de FigaroIDE. Ce guide est structuré autour des chapitres présentant successivement :

- les concepts et terminologie de base,
- les principes de stockage des fichiers,
- le lancement de FigaroIDE,
- la structure générale de l'interface,
- la création d'une nouvelle base de connaissances,
- l'écriture de la base de connaissances,
- la création du fichier de paramétrage de l'interface graphique de KB3,
- la vérification de l'écriture de la base de connaissances,

2 DOCUMENTS DE REFERENCE

[1] Manuel utilisateur de KB3 V3, H-T52-2012-01065-FR, EDF, 17 septembre 2012.

[2] Manuel de référence du langage de modélisation probabiliste Figaro, EDF 6125-1612-2017-00624-FRavril 2017.

[3] Syntaxe du langage de modélisation probabiliste Figaro, EDF 6125-1612-2016-15549-FR, octobre 2016.

3 ACRONYMES

BdC	Base de Connaissances
EI	Evénement Indésirable
IDE	Integrated Development Environment
IHM	Interface Homme Machine
MCG	Macro Composant Graphique
OD	Objet Déduit
PC	PréCompilation
RM	Règle Manuelle
XML	Extensible Markup Language

4 CONCEPTS ET TERMINOLOGIE DE BASE

Pour réaliser une étude de sûreté de fonctionnement d'un système avec KB3 [1], il faut disposer d'une base de connaissances dédiée aux études de ce type de système. Les bases de connaissances (BdC) sont écrites en langage FIGARO, langage spécifique développé à EDF [2][3].

FigaroIDE est un outil d'aide au développement et à la vérification des bases de connaissances.

5 FICHIERS D'UNE BDC

Une BdC est décrite par un ensemble de plusieurs fichiers :

- un fichier d'extension .fi décrivant en langage Figaro les classes des objets associés à la catégorie de systèmes modélisés dans la BdC,
- un fichier d'extension .bdc décrivant en langage XML les paramètres de l'interface graphique de la base dans KB3 V3 et donc constituée des représentations graphiques associées à chaque objet, des modes de traitement autorisés, des changements de couleur lors des simulations (variantes graphiques d'un même objet)...
- un répertoire "icons"[1] contenant les fichiers d'extension .ico pour représenter les objets de la palette KB3 V3 et les fichiers associés d'extension .sym de même nom représentant les mêmes objets sur le schéma du système dans KB3 V3. Un fichier .ico peut ne pas être associé à un fichier .sym ; c'est le cas s'il est associé à un lien ou à un objet dit non graphique (non représentable sur le schéma du système dans KB3 V3). Un fichier .ico peut être associé à plusieurs fichiers .sym correspondant aux variantes graphiques de l'objet lorsque certaines conditions sont réalisées (par exemple pour le repérage d'un objet en panne).

6 LANCEMENT DE FIGAROIDE

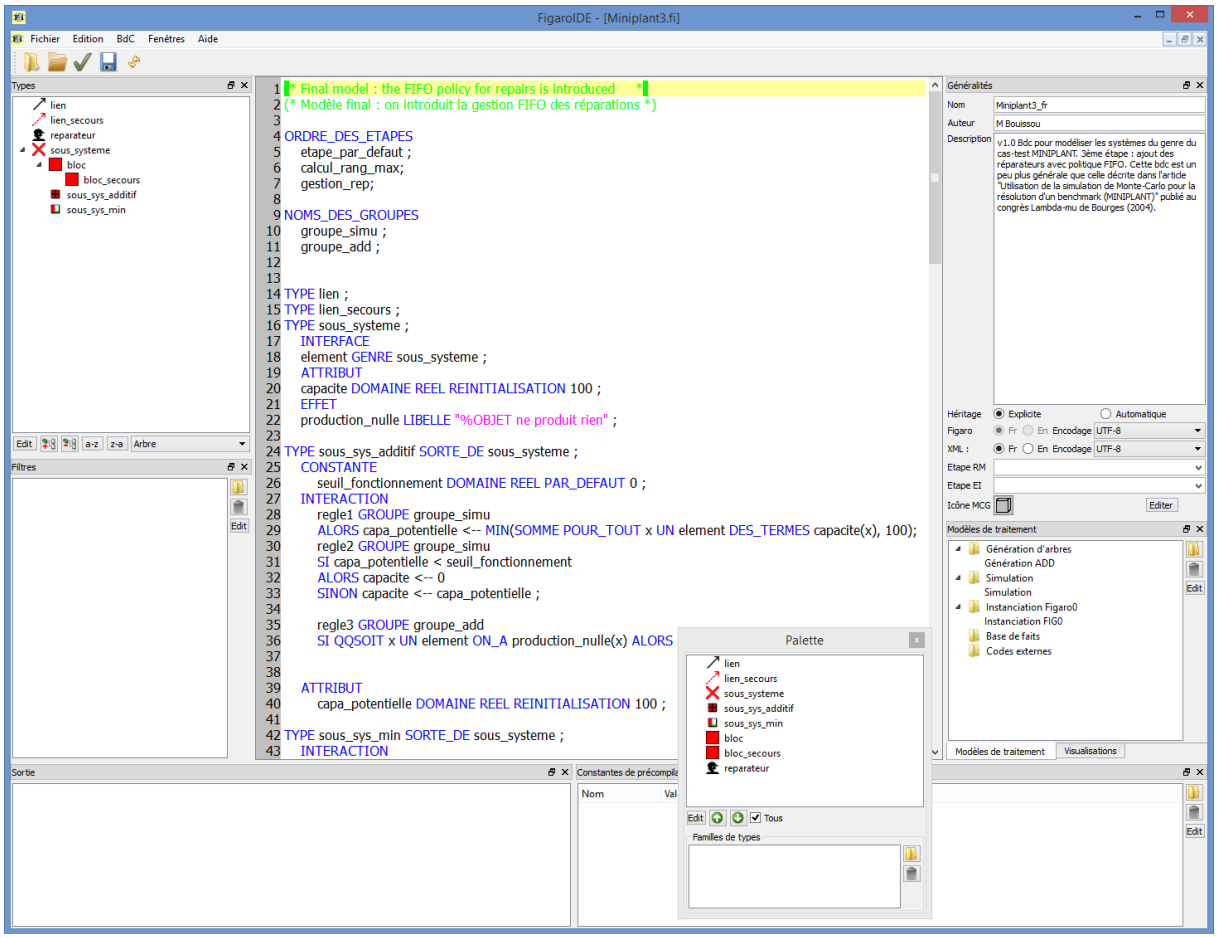
FigaroIDE peut être lancé par un double-clic sur l'exécutable lui-même ou un raccourci pointant vers lui.

7 STRUCTURE GENERALE DE L'INTERFACE

L'IHM de FigaroIDE est constituée d'une fenêtre principale comprenant les éléments suivants (cf. Figure 1) :

- une barre de menus (cf. § 7.1),
- une barre d'outils reprenant certaines fonctions principales des menus et identifiées par des icônes,
- un ensemble de widgets (panneaux ancrables) (cf. § 7.2),
- un espace de travail dans lequel s'inscrivent l'éditeur Figaro (éditeur de fichier .fi) et la fenêtre du fichier XML associé à la base de connaissances.

[1] Ce répertoire doit s'appeler "icons" ou du même nom que le fichier .fi sans l'extension pour être lisible dans KB3 V3 et se trouver à côté des fichiers d'extension .bdc et .fi.




```
1  (* Final model : the FIFO policy for repairs is introduced     *)
2  (* Modèle final : on introduit la gestion FIFO des réparations *)
3
4  ORDRE_DES_ETAPES
5     etape_par_defaut ;
6     calcul_rang_max;
7     gestion_rep;
8
9  NOMS_DES_GROUPES
10    groupe_simu ;
11    groupe_add ;
12
13
14 TYPE lien ;
15 TYPE lien_secours ;
16 TYPE sous_systeme ;
17    INTERFACE
18    element GENRE sous_systeme ;
19    ATTRIBUT
20    capacite DOMAINE REEL REINITIALISATION 100 ;
21    EFFET
22    production_nulle LIBELLE "%OBJET ne produit rien" ;
23
24 TYPE sous_sys_additif SORTE_DE sous_systeme ;
25    CONSTANTE
26       seuil_fonctionnement DOMAINE REEL PAR_DEFAUT 0 ;
27    INTERACTION
28       regle1 GROUPE groupe_simu
29       ALORS capa_potentielle <-- MIN(SOMME POUR_TOUT x UN element DES_TERMES capacite(x), 100);
30       regle2 GROUPE groupe_simu
31       SI capa_potentielle < seuil_fonctionnement
32       ALORS capacite <-- 0
33       SINON capacite <-- capa_potentielle ;
34
35       regle3 GROUPE groupe_add
36       SI QQSOIT x UN element ON_A production_nulle(x) ALORS
37
38
39    ATTRIBUT
40       capa_potentielle DOMAINE REEL REINITIALISATION 100 ;
41
42 TYPE sous_sys_min SORTE_DE sous_systeme ;
43    INTERACTION
```

Figure 1 : Présentation générale de l'interface principale de FigaroIDE

7.1 MENUS

Les menus et leurs fonctions principales sont les suivants :

- le menu Fichier permet un ensemble d'actions de gestion sur une BdC (création, ouverture, fermeture, sauvegarde) et de quitter FigaroIDE,
- le menu Édition permet d'effectuer les opérations classiques d'édition (copier, couper, coller), d'accéder à l'outil de recherche (Rechercher) et de définir certains paramètres d'édition (Options...),
- le menu BdC autorise certaines actions sur la BdC ouverte dans l'espace de travail (Vérifier : cf. § 12, Synchroniser permet de mettre en cohérence les fichiers .fi et .xml de la base avec les modifications effectuées depuis les widgets, Précompiler : cf. § 11.11). Hériter permet d'afficher dans l'espace de travail la "BdC héritée" présentant pour chaque type les interfaces, constantes, attributs, occurrences... hérités,
- le menu Fenêtres propose une présentation sous forme de mosaïque ou de cascade des fichiers ouverts dans l'espace de travail et présente la liste des fichiers ouverts. Le nom du fichier coché correspond au fichier actif dans l'espace de travail. Il suffit de cocher un autre nom de la liste pour le rendre actif.

Le clic droit de la souris donne accès à des menus contextuels regroupant un ensemble de commandes actives sur la sélection courante.

7.2 WIDGETS

Les widgets sont au nombre de 10 :

- Les widgets Types, Algorithmes OD, Généralités, Palette, Visualisations, Modèles de traitement, Constantes de précompilation, Groupes de règles de nommage et Filtres visent à aider l'utilisateur dans la construction du fichier xml d'extension .bdc qui permettra de définir les paramètres de l'interface graphique de la BdC dans KB3 V3.
- Le widget Types fournit également des outils de recherche rapide par des tris et des sélections sur les types et une présentation hiérarchique des types (présentation des héritages entre types).
- Enfin, le widget Sortie permet de visualiser les erreurs d'écriture dans la BdC suite à une vérification grammaticale et syntaxique de la base et des incohérences dans le fichier .bdc (cf. § 12).

L'utilisateur choisit les widgets à afficher en cliquant sur le bouton droit de la souris sur la barre des menus, ce qui fait apparaître la liste des widgets. Les widgets affichés sont cochés dans la liste. Il suffit de les décocher pour en supprimer l'affichage.

L'utilisateur peut redimensionner les widgets et les placer n'importe où à l'écran.

7.2.1 WIDGET TYPES

Le widget Types permet de visualiser l'ensemble des types définis dans la BdC et d'accéder à certaines opérations réalisables sur les types.

Un clic droit de la souris sur l'un des types de la liste fait apparaître les actions à réaliser. Les commandes Sélectionner fi et Sélectionner xml conduisent à l'affichage du type respectivement dans le fichier .fi et dans le fichier .xml.

La commande Editer occasionne l'ouverture de la fenêtre du type sélectionné dans laquelle est présenté l'ensemble des propriétés du type : description (texte), instance graphique dont l'icône si elle existe est présentée, les variantes graphiques et les points de connexion à l'objet (cf. § 11.1, § 11.2).

7.2.2 ALGORITHMES OD

Le widget Algorithmes OD fournit les outils nécessaires à la création des objets déduits (OD) (cf. § 11.9).

7.2.3 GENERALITES

Le widget Généralités permet de définir les noms, auteur, description et langue de la BdC, et de spécifier :

- les encodages souhaités pour les fichiers .fi et .bdc (il est recommandé d'utiliser l'encodage UTF-8+BOM, reconnu automatiquement par la plupart des éditeurs de texte et permettant de transférer les fichiers sans changement d'un environnement Windows à Linux),
- la langue des balises du fichier .bdc (celle du fichier .fi est automatiquement déterminée par les premiers mots-clés tapés par l'utilisateur),
- la représentation graphique du MCG (cf. § 11.5),
- l'étape d'ajout des règles manuelles (cf. § 11.7),
- l'étape d'ajout des événements indésirables (cf. § 11.8).

7.2.4 PALETTE

Le widget Palette présente les types qui apparaîtront dans la palette de KB3 et dans l'ordre défini par l'utilisateur (cf. § 11.2). Lorsque Tous est décoché, seuls les types apparaissant dans la palette sont affichés. Tous permet de voir tous les types déclarés comme nœud ou lien bien que n'apparaissant pas dans la palette de KB3.

Tout comme pour le widget Types, il est possible d'accéder à certaines opérations réalisables sur les types en sélectionnant un type puis en cliquant sur le bouton Edit. La commande Editer occasionne l'ouverture de la fenêtre du type sélectionné dans laquelle est présenté l'ensemble des propriétés du type : description (texte), instance graphique dont l'icône si elle existe est présentée, les variantes graphiques et les points de connexion à l'objet (cf. § 11.1, § 11.2).

Le widget Palette permet aussi de spécifier les familles de types (cf. § 11.4).

7.2.5 VISUALISATIONS

Le widget Visualisations permet de spécifier les différentes visualisations proposées par la base de connaissances (cf. §11.5).

7.2.6 MODELES DE TRAITEMENT

Le widget Modèles de traitement permet de définir des modèles pour les traitements qui seront effectués dans KB3 (cf. § 11.10).

7.2.7 CONSTANTES DE PRECOMPILATION

Le widget Constantes de précompilation permet de définir des constantes de précompilation qui modifieront le fonctionnement de la BdC dans KB3 (cf. § 11.11).

7.2.8 FILTRES

Ce widget permet de construire les filtres de recherche formant les rapports prédéfinis proposés par la base de connaissances.

7.2.9 GROUPES DE REGLES DE NOMMAGE

Le widget Groupes de règles de nommage permet de définir les groupes de règles destinés à modifier les noms et les descriptions des portes d'arbres de défaillances de manière à les adapter au logiciel de quantification utilisé.

7.2.10 SORTIE

Le widget Sortie est une fenêtre de visualisation des erreurs contenues dans la BdC (cf. § 12).

8 OUVRIR UNE BASE DE CONNAISSANCES

La commande Ouvrir... du menu Fichier permet d'accéder à la hiérarchie des répertoires et de sélectionner la base de connaissances d'intérêt. La sélection de l'un ou l'autre des fichiers .fi ou .bdc occasionne l'ouverture des deux fichiers dans l'espace de travail.

La commande Récents permet d'avoir accès directement au répertoire des dernières bases ouvertes.

9 CREATION D'UNE NOUVELLE BASE DE CONNAISSANCES

La sélection de la commande Nouveau du menu Fichier crée deux nouvelles pages .fi et .bdc dans l'espace de travail. Les fichiers correspondants n'ont pas été créés et ne seront pas créés

tant que l'utilisateur n'aura pas enregistré la base (commande Enregistrer ou Enregistrer sous... du menu Fichier).

***Attention** : le fichier .bdc ne sera sauvegardé que si un nom est donné à la base de connaissances.*

10 ÉCRITURE DE LA BASE DE CONNAISSANCES

Pour une vue d'ensemble sur les concepts et la syntaxe du langage FIGARO, l'utilisateur pourra se référer au "Manuel de référence du langage FIGARO [2][3].

L'utilisateur écrit la base dans l'éditeur de texte correspondant au fichier d'extension .fi.

FigaroIDE propose plusieurs aides à l'écriture de la base[2] :

- l'autocomplétion,
- la colorisation :

  des mots du vocabulaire du langage Figaro (bleu)

  des valeurs énumérées des constantes et attributs (vert foncé)

  des chaînes de caractères descriptions et libellés (rose)

  des commentaires (vert clair).

Voici ci-dessous un exemple de contenu de base de connaissances en langage Figaro.

```
DESCRIPTION_BDC "Petite base de connaissances pédagogique.
   Elle permet de décrire des réseaux de télécommunication
   maillés avec des défaillances sur les noeuds et les liens.
   Elle a deux utilisations possibles : simulation (avec le groupe
   de règles groupe_simu) et génération d'arbres de
   défaillances (avec le groupe de règles groupe_add)";
(* Auteur : M. Bouissou, 20 déc 2007 *)
NOMS_DES_GROUPES
groupe_simu ;
groupe_add ;

TYPE noeud GR_NOEUD ;
CONSTANTE
fonction DOMAINE 'source' 'but' 'intermediaire' PAR_DEFAUT 'intermediaire' ;
lambda DOMAINE REEL PAR_DEFAUT 1e-5 ROLE CONCEPTION;
mu DOMAINE REEL PAR_DEFAUT 0.1 ROLE CONCEPTION;
EFFET
relie ;
OCCURRENCE
GROUPE groupe_simu
IL_PEUT_SE_PRODUIRE
DEFAILLANCE def
LOI EXP(lambda);
INTERACTION
regle1
SI MARCHE ET fonction = 'source'
ALORS relie;
PANNE def
```

[2] Les mots clés peuvent être en français ou en anglais. FigaroIDE reconnaît la langue automatiquement et propose des complétions dans la langue de la base.

```
DONNEES_FIABILISTES
GROUPE groupe_add
MODELE_GLM
GAMMA 0.
LAMBDA lambda
MU mu ;

TYPE source SORTE_DE noeud;
CONSTANTE
fonction PAR_DEFAUT 'source';

TYPE but SORTE_DE noeud;
CONSTANTE
fonction PAR_DEFAUT 'but';

TYPE lien GR_LIEN ;
CONSTANTE
lambda_arete DOMAINE REEL PAR_DEFAUT 1e-5 ROLE CONCEPTION ;
mu_arete DOMAINE REEL PAR_DEFAUT 1 ROLE CONCEPTION ;
PANNE
coupure
DONNEES_FIABILISTES
GROUPE groupe_add
MODELE_GLM
GAMMA 0.
LAMBDA lambda_arete
MU mu_arete ;
OCCURRENCE
GROUPE groupe_simu
IL_PEUT_SE_PRODUIRE
DEFAILLANCE coupure
LOI EXP(lambda_arete);

TYPE arete_uni_dir SORTE_DE lien ;
INTERFACE
depart GENRE noeud CARDINAL 1;
arrivee GENRE noeud CARDINAL 1 ;
INTERACTION
regle1
SI MARCHE ET relie(depart) ET MARCHE(arrivee)
ALORS relie(arrivee) ;

TYPE arete_bi_dir SORTE_DE lien ;
INTERFACE
extremite GENRE noeud CARDINAL 2 ;
INTERACTION
regle1
SI MARCHE ET
(QQSOIT x UNE extremite ON_A MARCHE(x)) ET
IL_EXISTE x UNE extremite TELLE_QUE relie DE x
ALORS POUR_TOUT z UNE extremite FAIRE relie(z);

TYPE compteur_def ;
ATTRIBUT
nb_def DOMAINE ENTIER PAR_DEFAUT 0;
INTERACTION
regle1
GROUPE groupe_simu
```

```
ALORS nb_def <-- SOMME POUR_TOUT x UN OBJET DE_TYPE noeud DES_TERMES (1 * def(x)) +
SOMME POUR_TOUT y UN OBJET DE_TYPE lien DES_TERMES (1 * coupure(y)) ;
OBJET_SYSTEME Compteur_defaillances EST_UN compteur_def ;
```

11 CREATION DU FICHIER DE PARAMETRAGE DE L'INTERFACE GRAPHIQUE DE KB3

11.1 CREER LES REPRESENTATIONS GRAPHIQUES DES OBJETS DANS KB3

La création d'un fichier de paramétrage de l'interface graphique de KB3 (fichier .bdc) nécessite de définir les paramètres graphiques associés à chaque TYPE.

Pour cela, dans le widget Types ou le widget Palette, on sélectionne le type d'intérêt et on clique sur le bouton Edit (ou sur la commande Edit du menu contextuel). Ceci occasionne l'ouverture de la fenêtre du type sélectionné (cf.Figure 2) dans laquelle l'utilisateur viendra renseigner les propriétés graphiques des objets correspondant au type sélectionné.

Dans un premier temps, l'utilisateur choisit la forme à donner aux objets du type sélectionné. Un objet correspondant à un TYPE donné peut être de forme Lien[3] ou de forme Nœud. S'il n'a pas de forme, il ne correspond pas à un objet graphique et ne sera donc pas utilisable pour la construction du schéma du système. L'utilisateur précise également s'il veut voir apparaître dans la palette des objets de KB3 les objets de ce type.

Attention : tout type héritant d'un nœud est un nœud et tout type héritant d'un lien est un lien. En conséquence, l'utilisateur ne pourra pas modifier la forme d'un enfant.

Si l'utilisateur a déjà placé l'icône graphique qui représentera les objets du type d'intérêt dans le répertoire icons, cette dernière apparaîtra dans la fenêtre. Dans le cas contraire ou si l'utilisateur souhaite modifier l'icône, il pourra avoir accès directement à l'éditeur graphique installé sur son poste de travail en cliquant sur le bouton "Editer la représentation graphique par défaut" (cf.Figure 2).

Un objet de forme Lien ou Nœud n'a pas besoin de disposer d'instance graphique s'il n'est jamais placé sur le schéma du système. Dans ce cas l'utilisateur choisira l'instance graphique INTERDITE et dans le cas contraire, il choisira OBLIGATOIRE.

[3] Un lien relie deux nœuds. En pratique, un lien est essentiellement destiné à faciliter la saisie des interfaces dans KB3.

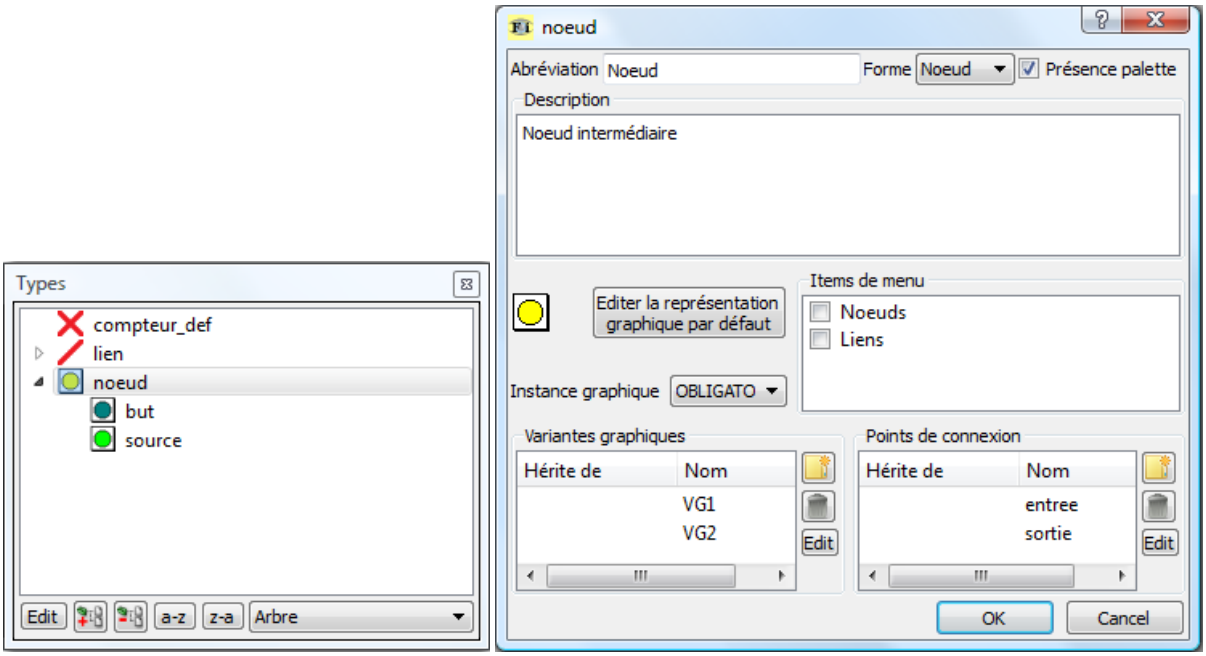


Figure 2 : exemple sur la création de la représentation graphique des objets de type nœud

Un objet peut avoir besoin de disposer de plusieurs représentations graphiques. On les appelle variantes graphiques. C'est souvent le cas lorsqu'on souhaite par exemple lors d'une simulation dans KB3 visualiser l'état d'un composant (cf. § 11.5). Ces variantes sont définies par l'utilisateur et correspondront à différents fichiers .sym ou à des aspects différents de l'instance graphique initiale. Pour ce dernier cas, l'utilisateur crée une nouvelle variante graphique en cliquant sur l'icône nouveau puis sur le bouton "Edit" qui propose alors à l'utilisateur dans une fenêtre dédiée différentes manières de modifier l'icône d'un objet sous condition et pour une visualisation dont le nom a été défini dans l'espace Visualisations du widget Généralités (cf. § 11.5 et exemple sur la Figure 3).

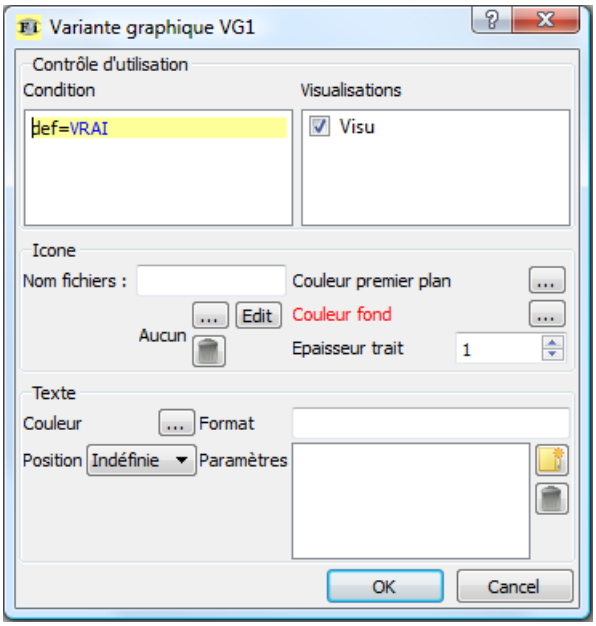


Figure 3 : exemple sur la création d'une variante graphique des objets de type nœud

11.2 CREER DES POINTS DE CONNEXION SUR LES OBJETS D'UN TYPE

Si l'objet est de forme Lien, les points de connexion sont toujours DEPART et ARRIVEE (les noms ne sont pas modifiables et aucun point de connexion ne peut être ajouté ou supprimé). FigaroIDE propose la liste des points de connexion déjà créés dans la base et a minima les points de connexion définis pour ses parents qui ne pourront être modifiés (cf.Figure 2).

Pour définir les points de connexion de nœuds auxquels un lien peut être connecté, l'utilisateur doit éditer les propriétés du point de connexion en cliquant sur le bouton Edit.

L'utilisateur peut alors ajouter des autorisations de connexions où il doit choisir le type de nœud accepté à la connexion et éventuellement définir le point de connexion accepté. Si le nom du point de connexion sur le nœud n'est pas défini, le lien pourra être connecté à tous les points de connexion du nœud (cas de la Figure 4 : pas de valeur définie pour le "Nom point connexion accepté").

Sur l'exemple de la Figure 4, le point de connexion DEPART pourra être branché sur des objets de type Composant.

Une connexion de lien à un nœud peut permettre de remplir automatiquement les interfaces des nœuds reliés et du lien. Pour cela l'utilisateur, clique sur le bouton "Ajouter remplissage interface", ce qui ajoute une règle de remplissage d'interface modifiable en double-cliquant sur la règle (cf. Figure 4).

Les principes de remplissage des interfaces reposent sur les deux mots clés contenus dans la règle. L'interface remplie est celle de l'objet correspondant au premier mot clé. Ainsi, les remplissages pour les différentes règles sont les suivants :

- REGLE_DEPART_LIEN : remplissage de l'interface nommée dans la colonne "Valeur" du nœud au DEPART du lien, par le nom du lien,
- REGLE_LIEN_DEPART : remplissage de l'interface nommée dans la colonne "Valeur" du lien par le nom du nœud au DEPART du lien,
- REGLE_ARRIVEE_LIEN : remplissage de l'interface nommée dans la colonne "Valeur" du nœud à l'ARRIVEE du lien, par le nom du lien,
- REGLE_LIEN_ARRIVEE : remplissage de l'interface nommée dans la colonne "Valeur" du lien par le nom du nœud à l'ARRIVEE du lien,
- REGLE_ARRIVEE_DEPART : remplissage de l'interface nommée dans la colonne "Valeur" du nœud à l'ARRIVEE du lien, par le nom du nœud au DEPART du lien,
- REGLE_DEPART_ARRIVEE : remplissage de l'interface nommée dans la colonne "Valeur" du nœud au DEPART du lien, par le nom du nœud à l'ARRIVEE du lien.

Pour l'exemple de la Figure 4, avec la REGLE_DEPART_ARRIVEE dans l'interface "composants_connectés" du nœud au DEPART du lien sera ajouté le nom du nœud à l'ARRIVEE du lien.

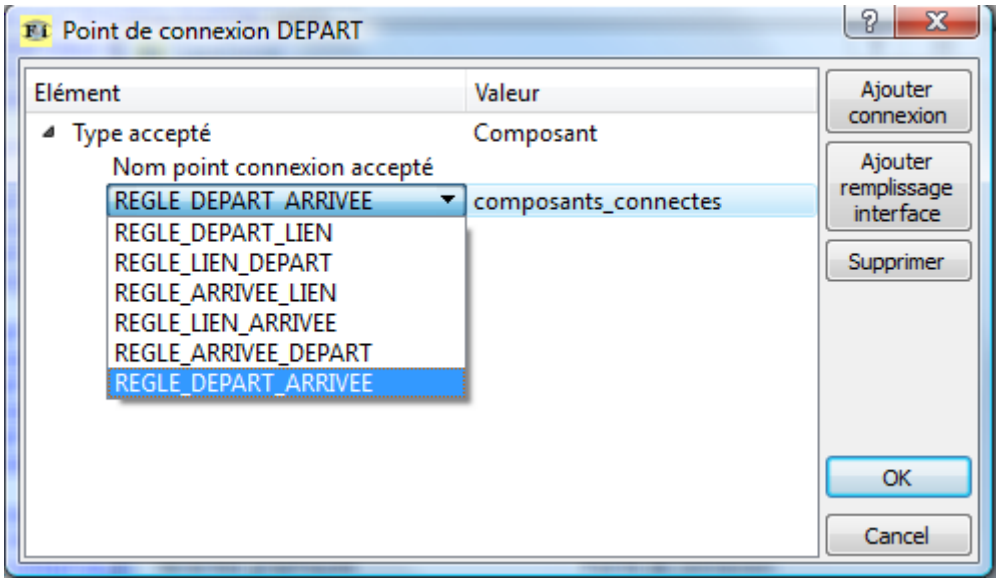


Figure 4 : exemple sur la création de points de connexion sur les objets de forme lien et sur le remplissage des interfaces

Si l'objet est de forme Nœud, plusieurs points de connexion peuvent être ajoutés au nœud par l'utilisateur qui définit leurs positions sur l'objet graphique (cf. Figure 5).

L'ajout d'un point de connexion s'obtient par le bouton "Ajouter connexion". FigaroIDE propose alors la liste de tous les types de lien de la BdC. L'utilisateur choisit alors le type de lien et le nom du point de connexion.

Si le type hérite d'un type de forme Nœud possédant lui-même des points de connexion, il lui est possible d'en hériter en choisissant le type parent dans la colonne Hérite de et le point de connexion hérité dans la colonne Nom. Dans ce cas, le point de connexion hérité ne peut pas être modifié.

Comme pour les liens, l'utilisateur doit définir les connexions autorisées sur chaque point de connexion. Pour cela l'utilisateur sélectionne le point de connexion à modifier et clique sur le bouton Edit.

L'utilisateur peut alors ajouter des autorisations de connexions où il doit choisir le type de lien accepté à la connexion et éventuellement définir le point de connexion DEPART ou ARRIVEE accepté. Si le nom du point de connexion sur le lien n'est pas défini, le nœud pourra être connecté à toutes les extrémités du lien.

Si l'utilisateur souhaite remplir une interface du nœud ou du lien connectés, il clique sur le bouton "Ajouter remplissage interface" et choisit l'interface à remplir en fonction de la règle :

- REGLE_NOEUD_LIEN : remplissage de l'interface du nœud nommée dans la colonne "Valeur" par le nom du lien,
- REGLE_LIEN_NOEUD : remplissage de l'interface du lien nommée dans la colonne "Valeur" par le nom du nœud.

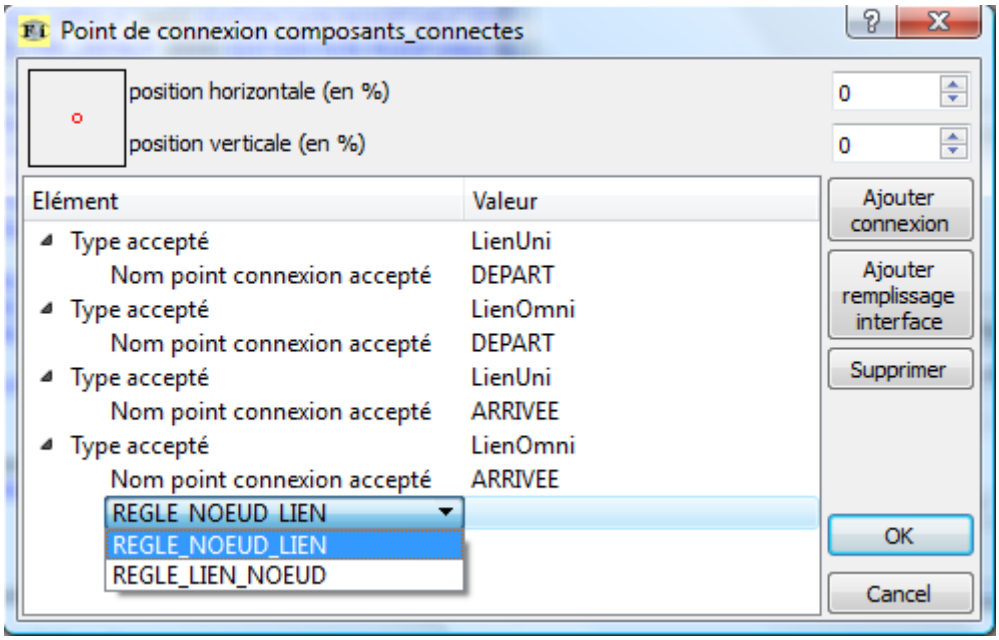


Figure 5 : exemple sur la création de points de connexion sur les objets de forme nœud et sur le remplissage des interfaces

11.3 DEFINIR LA PALETTE DES TYPES

L'utilisateur peut choisir de présenter les types dans la palette de KB3 dans un ordre non nécessairement alphabétique (par exemple, les nœuds en premier puis les liens). Pour cela, l'utilisateur sélectionne un type dans la liste et le déplace vers le haut ou vers le bas en s'aidant des flèches  .

11.4 CREER DES REGROUPEMENTS DES TYPES D'OBJETS

Pour certaines bases de connaissances un peu grosses, il peut être intéressant pour l'utilisateur de présenter dans la palette de KB3 des regroupements des types d'objets (par exemple, regrouper les nœuds et les liens dans deux répertoires séparés). C'est dans l'espace "Familles de types" du widget Palette que l'utilisateur définira ces regroupements.

11.5 CREER DES VISUALISATIONS

Le concepteur d'une BDC peut spécifier plusieurs visualisations qui permettront d'afficher dans KB3 certains résultats de la simulation sur le schéma du système (par exemple, la visualisation des composants en panne, la visualisation correspondant à la configuration initiale du système par une représentation des états des composants). Ces différentes visualisations supposent que l'utilisateur ait défini dans la base des variantes graphiques pour les types qui feront apparaître les objets de façon différente suivant la réalisation de certaines conditions (cf. § 11.1). L'utilisateur crée une nouvelle visualisation dans le Widget Visualisations.

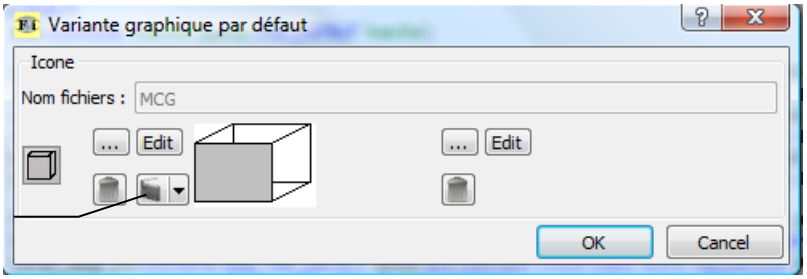
11.6 CHANGER LA REPRESENTATION GRAPHIQUE DES MCG

Outre les icônes correspondant aux types définis dans la bdc, il faut systématiquement avoir une icône de nom MCG (donc les deux fichiers MCG.ico et MCG.sym) qui servira à représenter les macro-composants graphiques (MCG). Par défaut, l'icône des MCG est la suivante :  .

- Pour modifier l'icône, l'utilisateur clique sur le bouton Editer du Widget "Généralités" en

face de l'icône MCG. Ceci a pour effet d'ouvrir la fenêtre présentée sur la Figure 6 dans laquelle FigaroIDE donne accès à deux outils de dessin par :

- le bouton Edit de gauche qui lance l'outil graphique défini par défaut sur le poste de travail de l'utilisateur pour les fichiers .ico,
- le bouton Edit de droite qui lance l'outil graphique Symbol Designer fourni avec le logiciel FigaroIDE.



Accès direct à un ensemble d'icônes

Figure 6 : fenêtre de modification de l'icône MCG

11.7 AJOUTER DES REGLES MANUELLES

Il peut arriver que l'utilisateur de KB3 désire ajouter des règles pour préciser un dysfonctionnement ou des connaissances spécifiques. Il emploiera pour cela des règles manuelles (RM) qui seront décrites de manière graphique par un arbre.

Le concepteur de la BdC peut obliger l'utilisateur de KB3 à créer ses règles manuelles dans des étapes spécifiques de la base de connaissances. Pour cela, il définit dans le widget Généralités l'étape dans laquelle seront définies les RM.

Remarque : En l'absence de spécification d'une étape pour les RM, les règles manuelles construites dans KB3 seront ajoutées dans l'étape créée par défaut : etape_par_defaut.

11.8 AJOUTER DES EVENEMENTS INDESIRABLES

Les événements indésirables (EI) sont essentiellement utilisés pour la génération des arbres de défaillances dans KB3. Ils pourraient être utilisés également pour construire des indicateurs pour l'exploitation du modèle du système, mais cette fonctionnalité est déjà fournie par Yams.

Tout comme pour les règles manuelles, Le concepteur de la BdC peut obliger l'utilisateur de KB3 à créer ses événements indésirables dans des étapes spécifiques de la base. Pour cela, il définit dans le widget Généralités l'étape dans laquelle seront définis les EI.

Remarque : En l'absence de spécification d'une étape pour les EI, les événements indésirables construits dans KB3 seront ajoutés dans l'étape créée par défaut : etape_par_defaut.

11.9 CREER DES ALGORITHMES D'OBJETS DEDUITS

Un objet déduit est un objet construit automatiquement par KB3 lors du lancement d'une génération d'arbre, d'une simulation ou d'une instanciation.

Les algorithmes d'objets déduits sont destinés à définir la construction de ces objets. Ces algorithmes peuvent être également utilisés pour compléter des remplissages d'interfaces, des calculs de valeurs initiales de constantes ou encore pour effectuer des contrôles sur la saisie du système par l'utilisateur dans KB3.

Un algorithme peut être défini sous deux formes :

- Un algorithme paramétrable de construction de blocs obstructions
- Un algorithme défini par des règles employant une extension du langage Figaro

La première forme est proposée par FigaroIDE pour assurer la compatibilité avec d'anciennes bases de connaissances, mais il est préférable d'éviter de l'employer.

Les règles et équations des algorithmes des OD sont définies dans les types de la BdC choisis par l'utilisateur. Les règles et équations ont les mêmes facettes que les règles d'interaction et équations Figaro1 standard mais peuvent utiliser des opérateurs spécifiques permettant de modifier des objets ou le contenu des interfaces et d'afficher des messages d'erreur.

Par défaut les algorithmes sont exécutés dans l'ordre de déclaration dans le widget Algorithmes OD et les règles correspondant à un algorithme dans l'ordre de déclaration choisi par l'utilisateur au sein de chaque type dans la fenêtre de définition de l'algorithme (cf. exemple de la Figure 7, la règle ControleNumSat sera exécutée avant la règle ControleDatePlan).

Il est possible de modifier l'ordre d'exécution des algorithmes en précisant au sein d'un algorithme l'algorithme à exécuter après (champ Bouclage/saut). Ceci permet de construire des boucles d'algorithmes qui seront exécutées jusqu'à stabilisation des calculs.

L'exécution d'un algorithme consistera à parcourir les objets du système pour trouver les objets possédant des règles à exécuter. Pour chacun de ces objets, KB3 exécute les règles de *Départ* puis les règles de *Sortie*. Si dans ces objets, il y a des actions TRANSFERT_VERS l'algorithme exécute les règles de l'objet désigné par TRANSFERT_VERS en commençant par les règles d'*Entrée* puis les règles de *Sortie* puis applique récursivement les TRANSFERT_VERS.

Le principe de construction d'un algorithme est le suivant :

1. Dans le widget Algorithmes OD, ajouter un nom à la liste puis cliquer sur le bouton Edit,
2. Dans la fenêtre de l'algorithme, ajouter un type en cliquant sur le bouton "Ajouter Type" puis sélectionner le type d'intérêt sur la ligne ainsi créée dans la liste des types,
3. Ajouter une règle en cliquant sur le bouton Ajouter Règle applicable soit au *Départ*, en *Entrée* ou en *Sortie* selon le choix de l'utilisateur, ou Ajouter une équation en cliquant sur le bouton Ajouter équation,
4. Pour une équation, préciser le nom du système auquel appartient l'équation dans le champ Nom système apparaissant dans la partie droite de la fenêtre lorsque l'équation est sélectionnée,
5. Décrire en langage FIGARO la règle ou l'équation en les saisissant dans les espaces Si PC, Soit, Si, Alors, Sinon pour une règle et dans les espaces Si PC, Si, Formule pour une équation.
6. Préciser l'ordre de déclaration d'une règle ou d'une équation à l'aide des flèches vertes,
7. Ajouter à la liste des Types construits tous les types des objets qui seront construits par l'algorithme.

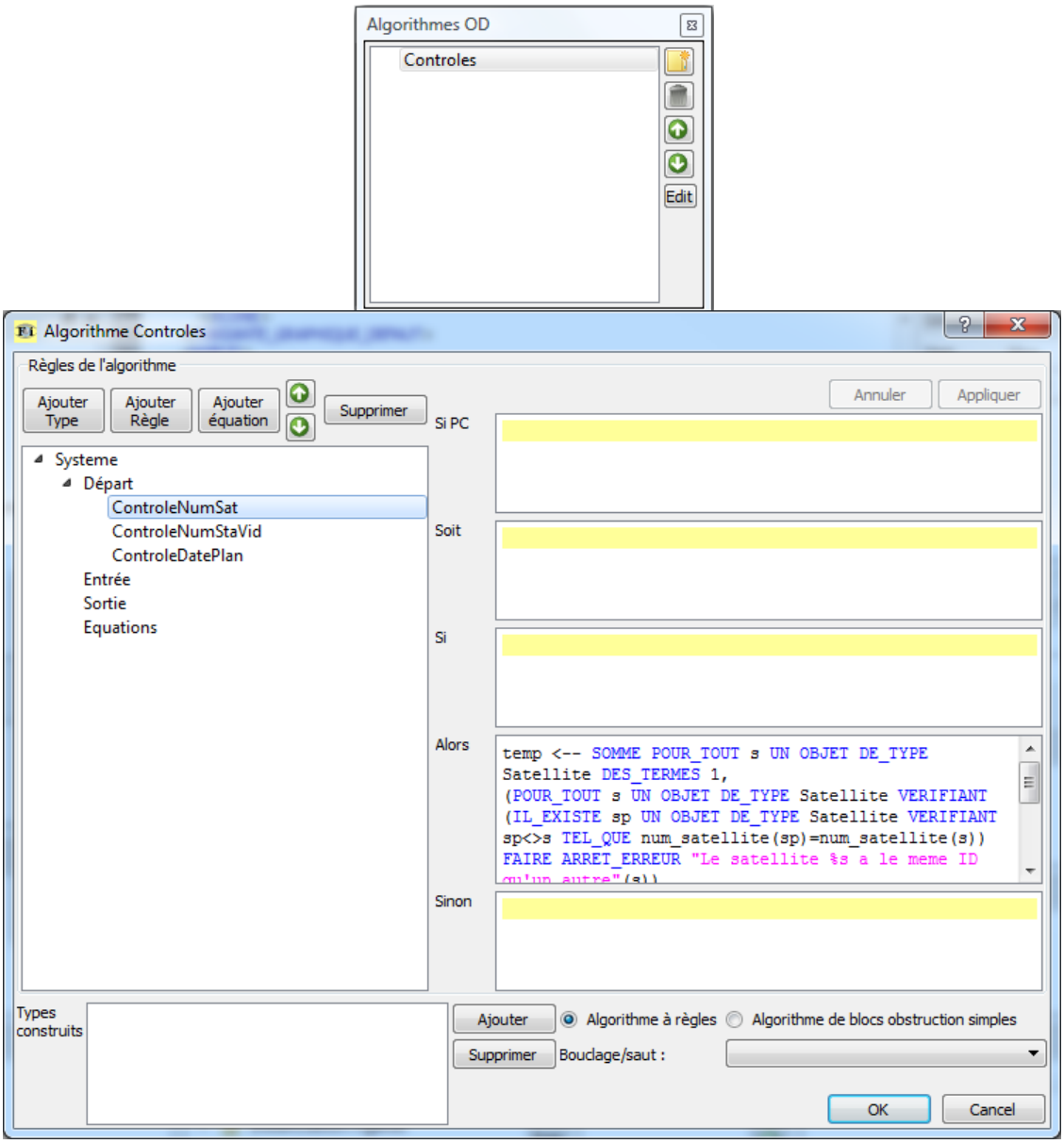


Figure 7 : widget Algorithmes OD et fenêtre de construction de l'algorithme Controle (affichage de la règle ControleNumSat)

11.10 CREER DES MODELES DE TRAITEMENT

KB3 propose cinq catégories de modèles de traitements à réaliser sur une étude (≡ modèle d'un système) dans KB3 (cf. Figure 8):

- **Génération d'arbres** de défaillances,
- **Simulation** interactive,
- **Instanciation Figaro0** correspondant à la création d'un fichier en langage Figaro0 suite à l'instanciation des règles de la BDC,
- **Base de faits** permet d'extraire la définition des objets, des interfaces remplies et des variables (constantes, attributs) modifiées,
- **Codes externes** permet le lancement de codes de calcul sur des données générées par KB3.

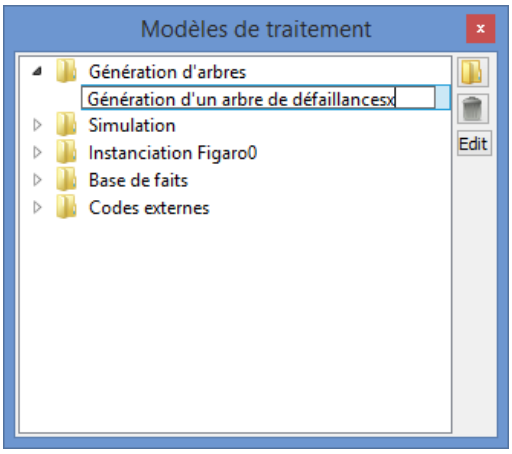


Figure 8 : widget Modèles de traitement

Pour qu'une étude ait accès à ces traitements, les modèles de traitements sont à spécifier dans la BdC liée à l'étude. Il est possible de créer plusieurs traitements d'une catégorie donnée.

Tous ces modèles apparaîtront dans les menus de l'onglet "Traitements" pour l'utilisateur de la base de connaissances sous KB3.

FigaroIDE propose des modèles par défaut pour la génération d'arbres, la simulation et l'instanciation Figaro0 (cf. Figure 9).

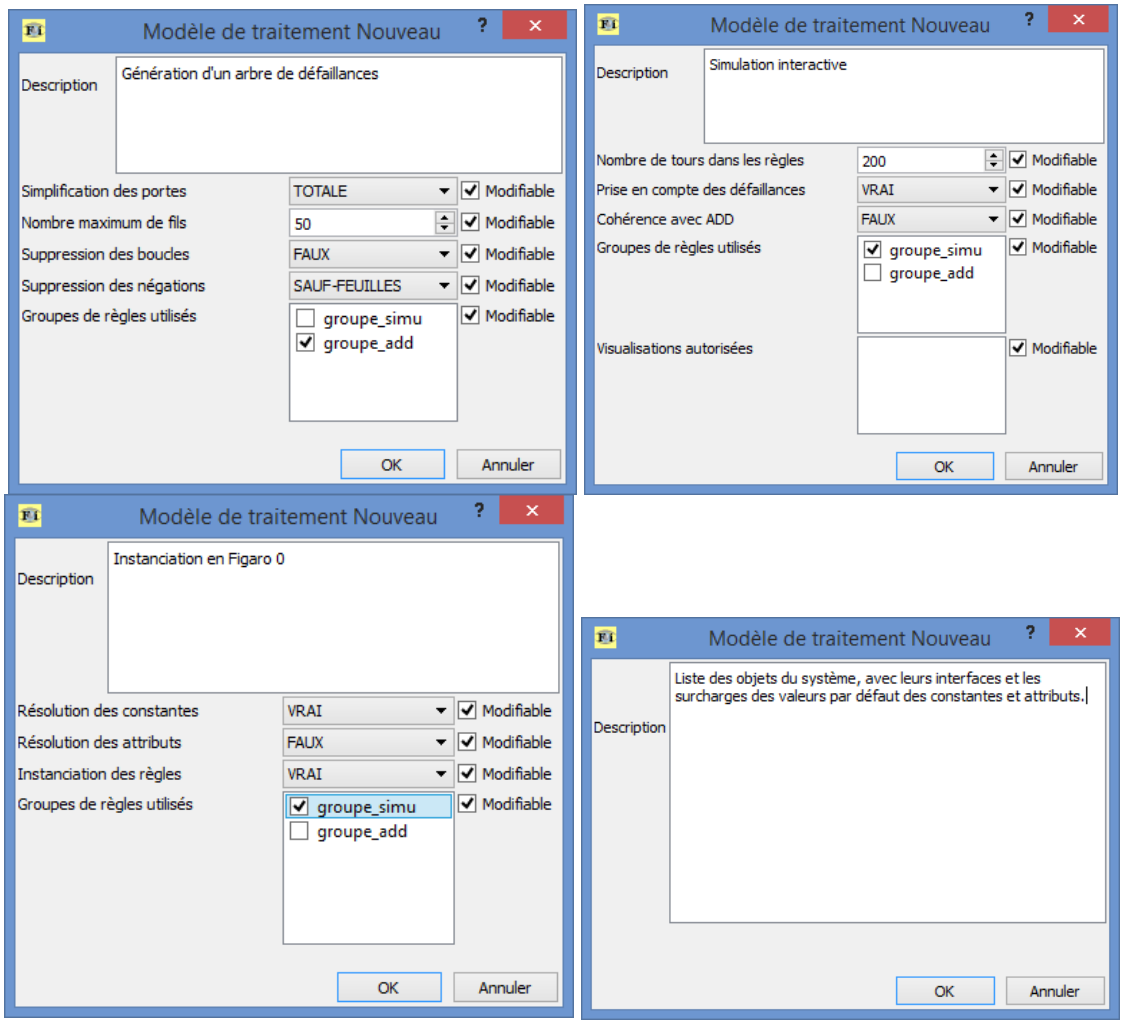


Figure 9 : Paramètres des modèles de traitement par défaut

Pour les modèles de traitement de codes externes, il faut préciser les éléments suivants (cf. Figure 10) :

- les informations nécessaires en entrée du logiciel externe : cases à cocher Figaro 0, ADD (arbre de défaillances), BdF (base de faits, c'est-à-dire liste des objets du modèle) ;
- la hiérarchie des sous répertoires utilisés pour stocker les fichiers Figaro 0 et ADD nécessaires : cases à cocher Base de connaissances, Etude, Modèle ;
- les logiciels externes utilisables (logiciels installés conjointement à KB3) : case à cocher Tous ou ajout de logiciels dans la liste (les logiciels YAMS et FIG0DEBUG sont classiquement disponibles). Les noms des codes externes doivent correspondre à des noms pris dans la liste accessible via le panneau des options de KB3, faute de quoi l'utilisateur de la base de connaissances aura un message "code inexistant" lorsqu'il essaiera de lancer ce traitement.

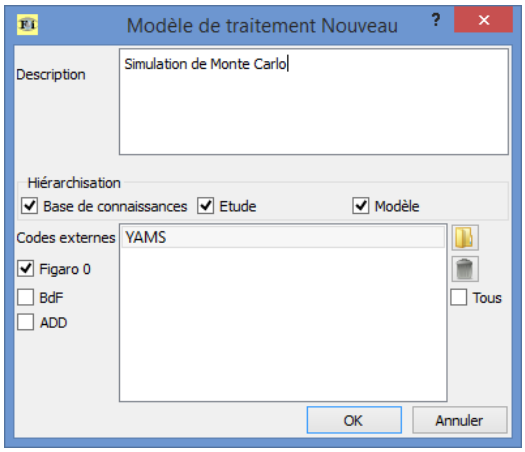


Figure 10 : Paramétrage des modèles de traitement Codes externes

11.11 CREER DES CONSTANTES DE PRECOMPILATION

Les constantes de précompilation permettent de créer des options de fonctionnement de la base de connaissances (par exemple, pour permettre de voir ou non les attributs des objets de l'étude, de contrôler ou non les constantes, de passer en mode débogage..). Dans KB3, l'utilisateur pourra alors choisir les options de fonctionnement de la BdC qu'il désire.

Les constantes de précompilation se définissent depuis le widget Constantes de précompilation. Le principe est le suivant : créer un nouveau nom puis cliquer sur le bouton Editer. Ceci occasionne l'ouverture de la fenêtre de définition de la constante (cf. Figure 11). L'application de la constante de précompilation est conditionnée par une valeur ("Mode débogage" dans le cas de la Figure 11) pouvant être entière si Valeur entière est cochée et/ou pouvant être obligatoire si Valeur obligatoire est cochée. Dans ce dernier cas, l'utilisation de la BdC pour lequel la constante de précompilation a été créée ne sera possible que si l'utilisateur de KB3 donne une valeur (choix de conception de la base).

Pour exemple, prenons le cas de la constante _DEBUG_ de la Figure 11. Le concepteur de la base a défini une utilisation de la constante dans la BdC (fichier .fi) par :

```
#ifdef __DEBUG__
#define _VISIBILITE_ EDITION NON MODIFIABLE
#else
#define _VISIBILITE_ EDITION NON VISIBLE
#endif
```

et a attribué `_VISIBILITE_` à toutes les constantes et/ou attributs qu'il souhaitait rendre visibles lors de l'application de la constante, par exemple :

```
capacite DOMAINE REEL PAR_DEFAUT 100 _VISIBILITE_;
```

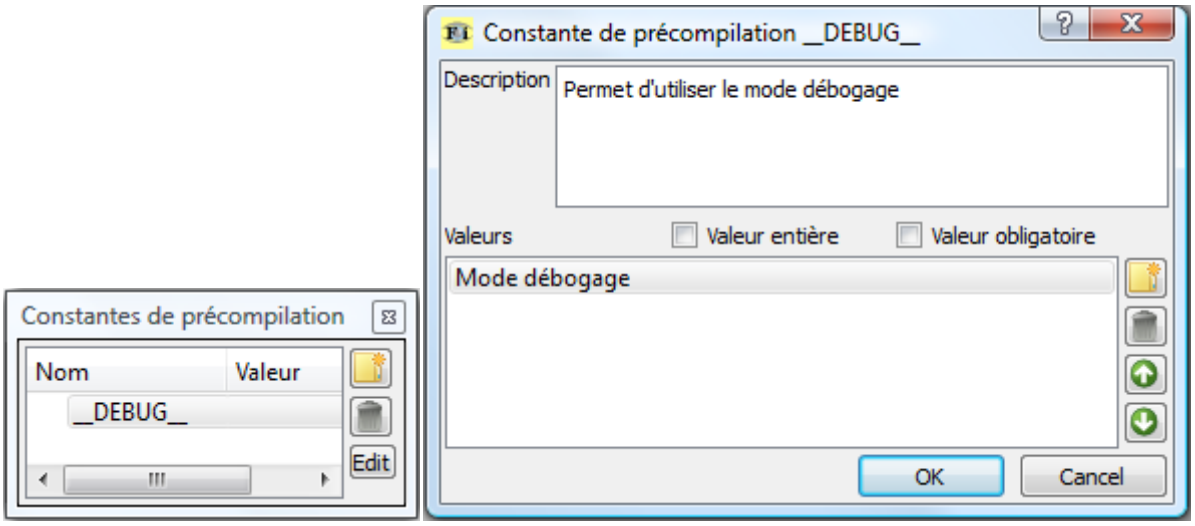


Figure 11 : Création d'une constante de précompilation

L'utilisateur peut visualiser dans la base (fichier .fi), l'application de la constante. Pour cela, il sélectionne la commande Précompiler du menu BdC après avoir choisi une valeur pour la constante, ce qui occasionne l'affichage de la BdC précompilée dans l'espace de travail. Pour l'exemple précédent, nous aurons dans la BdC précompilée :

Si la valeur associée dans le widget est non définie

```
capacite DOMAINE REEL PAR_DEFAUT 100 EDITION NON VISIBLE;
```

Si la valeur associée dans le widget est définie (valeur quelconque)

```
capacite DOMAINE REEL PAR_DEFAUT 100 EDITION NON MODIFIABLE;
```

11.12 CREER DES FILTRES

Un filtre est l'expression d'une recherche au sein d'un système construit à partir de la base de connaissances. Les filtres sont utilisés pour construire des rapports sur le système.

Un filtre est défini par le type d'élément recherché (Objet, Constante, Attribut, Panne, Effet, Interface, Interface inverse) ; des masques de filtrage du nom et de la description de l'élément, masques exprimés sous forme d'expressions régulières et d'une liste de types au sein desquels les éléments sont recherchés.

Les filtres sont créés dans le widget Filtres et définis dans un outil dédié.

Dans l'exemple ci-dessous, le filtre « Capacités » recherche les attributs dont le nom commence par "capa" au sein des objets dérivant du type sous_systeme. Ce filtre trouvera donc les attributs capacite et capa_potentielle.

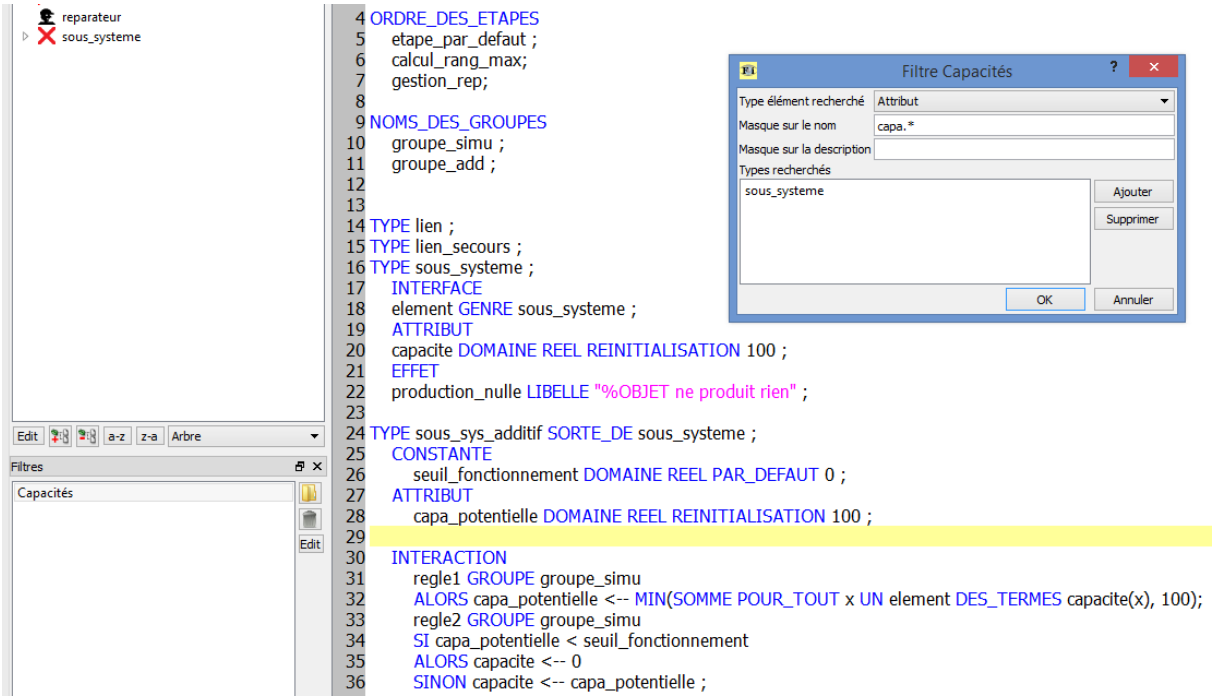


Figure 12 : Création d'un filtre

11.13 CREER DES GROUPES DE REGLES DE NOMMAGE

Les groupes de règles de nommage sont un ensemble de règles permettant de modifier les noms et les descriptions des éléments d'arbres de défaillances (portes, drapeaux, événements de bases…) avant leur exportation vers un code de quantification.

Chaque règle d'un groupe est chargée de modifier un type d'éléments d'arbre.

La modification du nom de l'élément consiste :

- à concaténer les chaînes de caractères suivantes : préfixe, nom de l'arbre, séparateur, nom de l'élément, suffixe
- à appliquer la règle de substitution sed sur la chaîne de caractères obtenue précédemment.

La modification de la description consiste à appliquer la règle de substitution sed à la description de l'élément.

Dans le cas d'une règle de drapeau, d'événement de base ou de renvoi externe, la règle de nommage peut aussi contenir une règle sed de modification de type. Cette règle est appliquée au nom concaténé obtenu précédemment pour obtenir une chaîne désignant le nouveau type de l'élément :

- #BE : désigne une transformation en événement de base
- #HE : désigne une transformation en drapeau
- #ET : désigne une transformation en renvoi externe

Les groupes de règles de nommage sont créés dans le widget Groupes de règles de nommage et définis dans l'outil dédié.

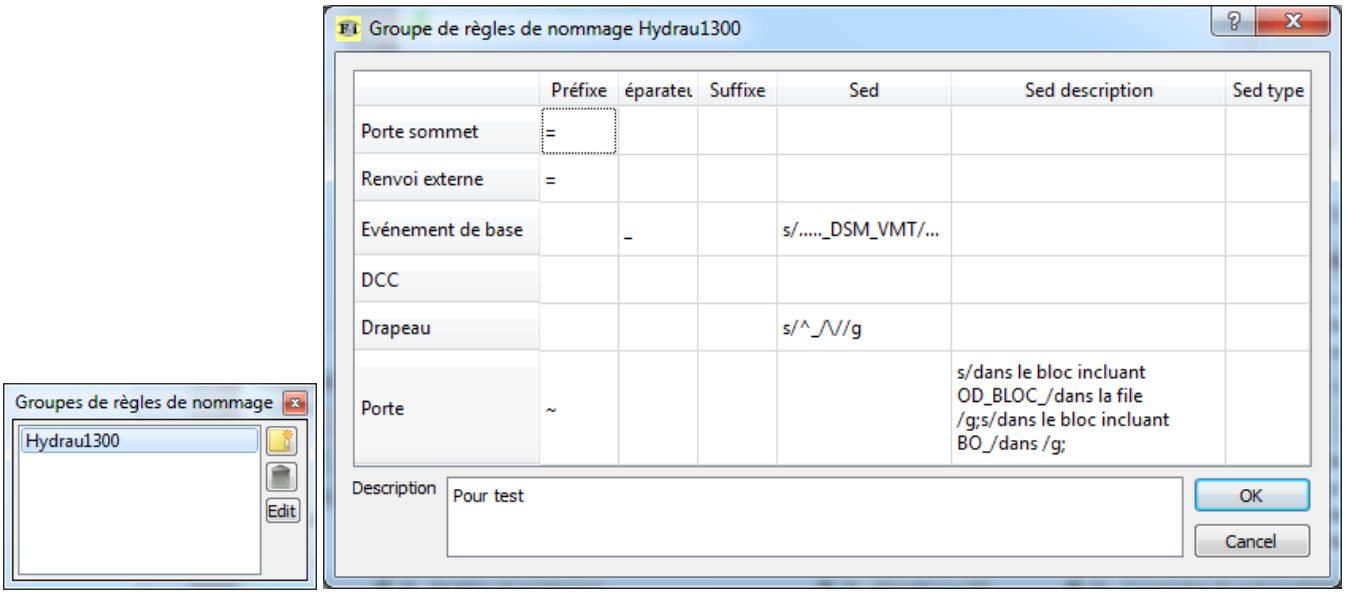


Figure 13 : Création d'un groupe de règles de nommage

12 VERIFIER L'ECRITURE DE LA BASES DE CONNAISSANCES

L'écriture de la BdC peut être vérifiée à tout moment en choisissant la commande Vérifier dans le menu BdC ou en cliquant sur l'icône correspondante. FigaroIDE affiche alors dans le widget Sortie les erreurs détectées dans le fichier .fi (cf. Figure 14) et les incohérences du fichier .bdc.

```
12 TYPE noeud GR_NOEUD ;
13 CONSTANTE
14 fonction DOMAINE 'source' 'but' 'intermediaire' PAR_DEFAUT 'intermediaire' ;
15 lambda DOMAINE REEL PAR_DEFAUT 1e-5 ROLE CONCEPTION;
16 mu DOMAINE REEL PAR_DEFAUT 0.1 ROLE CONCEPTION;
17 EFFET
18 relie ;
19 OCCURRENCE
20 GROUPE groupe_simu
21 IL_PEUT_SE_PRODUIRE
22 DEFAILLANCE def
23 LOI EXP(lambda);
24 INTERACTION
25 regle1
26 SI MARCHE ET fonction = 'source'
27 ALORS relie;
28 PANNE def
29 DONNEES_FIABILISTES
30 GROUPE groupe_add
31 MODELE_GLM
32 GAMMA 0.
33 LAMBDA lambda
34 MU mu ;
35
```

```
Sortie
bdc.fi(12) : Lors de la verification du type noeud,
 bdc.fi(25) : Lors de la verification de la regle d'interaction regle1,
  bdc.fi(25) : L'expression SI de regle1 est incorrecte,
   Erreur  : L'acces a cet element est impossible : fonction
bdc.fi(36) : Lors de la verification du type source,
 bdc.fi(25) : Lors de la verification de la regle d'interaction regle1,
  bdc.fi(25) : L'expression SI de regle1 est incorrecte,
   Erreur  : L'acces a cet element est impossible : fonction
bdc.fi(40) : Lors de la verification du type but,
 bdc.fi(25) : Lors de la verification de la regle d'interaction regle1,
  bdc.fi(25) : L'expression SI de regle1 est incorrecte,
   Erreur  : L'acces a cet element est impossible : fonction
Erreur  : Erreur pendant la verification de la BDC.
```

Figure 14 : exemple sur la vérification du vocabulaire et de la syntaxe d'un BdC

Un double-clic sur une ligne commençant par bdc permet d'afficher le texte signalé que ce soit dans le fichier .fi ou dans le fichier .bdc.