RUNE II

–

Objectifs & Philosophie

Index

Partie 1 - Démarrer avec RUNE II

1. Installation

Après avoir téléchargé RUNE II, sous la forme du fichier *rune-v2.xx.tar.gz*, décompressez cette archive, en tapant :

$> tar xvfz rune-v2.xx.tar.gz	(sous UNIX)

ou bien en utilisant un logiciel de décompression (WinRar...) sous Windows.

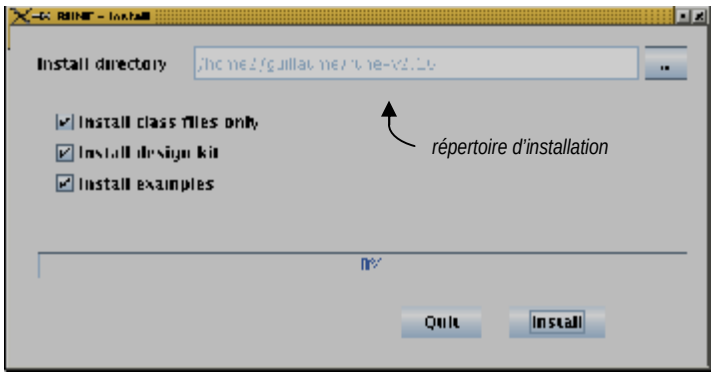
Vous obtenez alors un répertoire *install_rune*, contenant deux script (*install.bat* et *install.sh*), l'archive JAVA du programme (*install.jar*) et un fichier *README.txt*, expliquant succinctement la procédure à entreprendre pour installer l'application.

*Attention : pour pouvoir installer RUNE II, vous devez vous assurer que vous possédez sur votre ordinateur la version **1.5.0.03** de Java Runtime Environment (JRE), ou une version supérieure.*

Ensuite, vous pouvez alors lancer l'installation de RUNE II avec le script *install.sh* (sous système UNIX), ou *install.bat* (sous Windows).

–

Tout d'abord, vous devez choisir le répertoire d'installation ("*Install directory*"), RUNE II sera alors installé dans ce répertoire (ce répertoire peut être choisi par l'utilisateur à l'aide d'une fenêtre de sélection de fichiers).



Ensuite, plusieurs options s'offrent à vous.

§ *Install class files only*

Vous pouvez décider d'installer uniquement le programme (et dans ce cas, sélectionnez "*Install class files only*"), ou bien le programme ainsi que les sources. Vous pourrez alors apporter toutes les modifications dont vous avez besoin à l'application.

Attention : RUNE II est sous licence GNU/General Public Licence (G.P.L.), présentée en Annexe B. Si vous utilisez les sources, vous devez donc respecter les clauses de cette dernière.

§ *Install design kit*

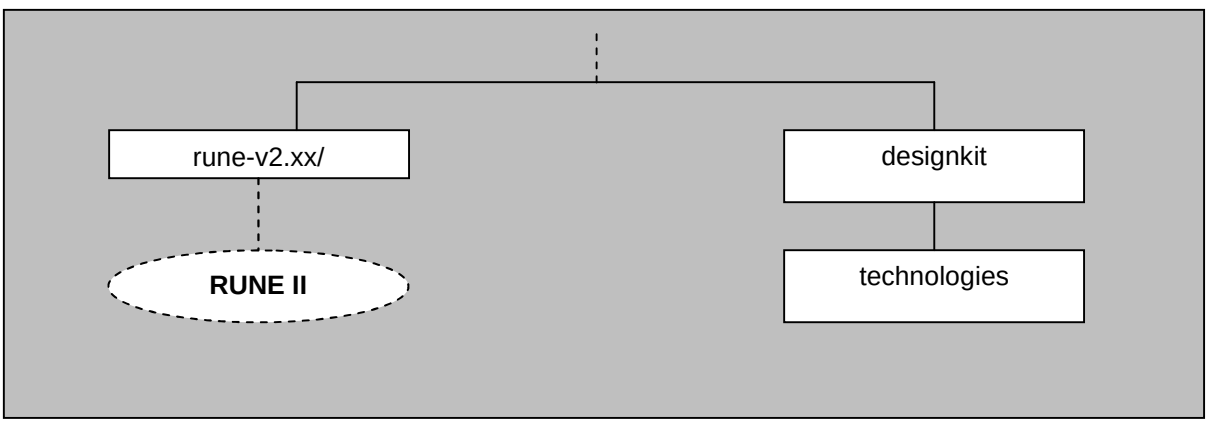
Ensuite, vous devez indiquer si vous désirez installer le "design kit" (dans ce cas, sélectionnez "*Install design kit*"). Le contenu de ce répertoire sera détaillé plus loin dans le manuel.
Seulement, si c'est la première fois que vous installer RUNE II, vous devez impérativement l'installer, sinon, vous pourrez utiliser celui que vous avez déjà installé. D'ailleurs, si le programme d'installation ne détecte pas de répertoire "design kit", vous serez averti.

§ *Install examples*

Finalement, vous pouvez choisir d'installer, en même temps que l'application, quelques exemples simples (dans ce cas, sélectionnez "*Install examples*"). Ceux-ci se trouveront alors dans le répertoire "*Examples*" du répertoire d'installation de RUNE II.

2. Architecture utilisée par RUNE II

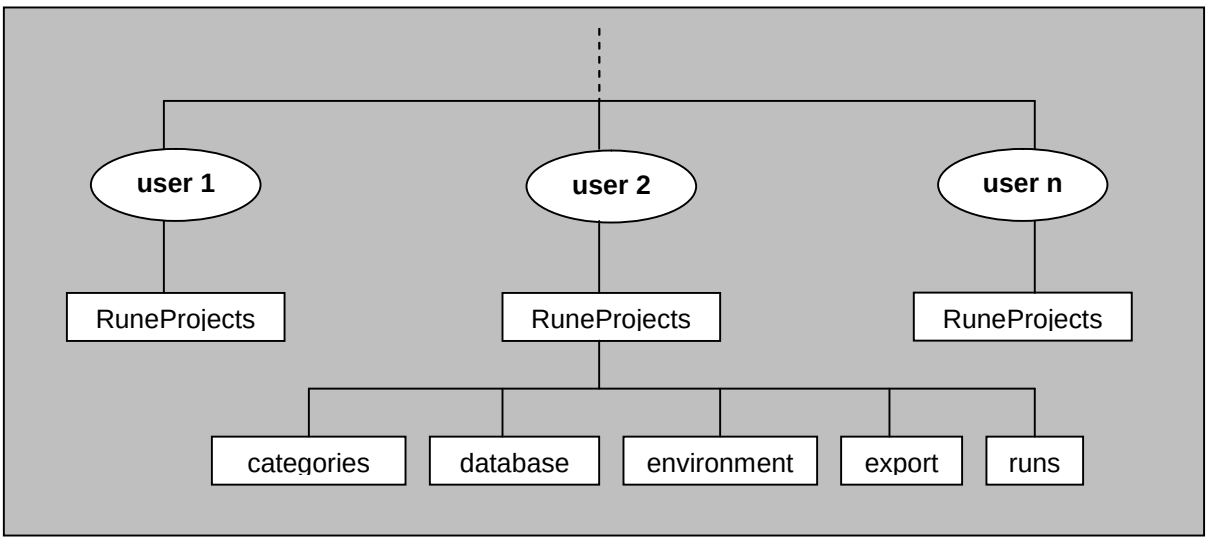
2.1. Répertoire d'installation de RUNE II



Après l'installation, RUNE II se situe donc dans un répertoire rune-v2.xx/ (ce nom peut bien évidemment être modifié par l'utilisateur). Et, dans le cas où l'utilisateur a également installé un design kit, ce dernier se situe dans le répertoire designkit/, au même niveau que le répertoire d'installation de RUNE II.

2.2. Répertoires de travail utilisateurs

Comme nous le verrons dans le paragraphe 3 (*Configuration*), chaque fois qu'un nouvel utilisateur lance RUNE II, il doit configurer son environnement de travail. Lors de ce premier lancement, RUNE II crée, par défaut, un répertoire de travail *RuneProjects/* à la racine du compte utilisateur, comme l'illustre le schéma ci-dessous.



3. Configuration

Au premier lancement de RUNE II par un utilisateur, ce dernier doit configurer son environnement de travail. L'environnement de travail se situe, par défaut, dans un répertoire *RuneProjects* à la racine du compte utilisateur. Ce répertoire se compose d'un ensemble de sous-répertoires.

3.1. Configuration du répertoire de travail

Le répertoire "work directory" représente le répertoire de travail proprement dit. C'est lui qui contient l'ensemble des éléments créés par l'utilisateur. Il se compose d'un ensemble de sous-répertoires :

- ***categories*** : répertoire qui contient les catégories/topologies créées par l'utilisateur

- ***database*** : répertoire qui contient les bases de données créées lors du lancement d'une synthèse avec utilisation de base de données (cet aspect de l'application est détaillé dans la partie 6)

- ***environment*** : ce répertoire contient les fichiers d'environnement (simulateurs et plans de conception) propre à l'utilisateur

- ***export*** : ce répertoire est le répertoire, par défaut, de stockage des éléments exportés par l'utilisateur

- ***runs*** : ce répertoire contient l'ensemble des fichiers de configuration de synthèse et de résultats de synthèse de l'utilisateur

3.2. Configuration du répertoire d'environnement

Le répertoire "environment directory" contient les fichiers d'environnement effectivement utilisé par l'utilisateur (qui peuvent être différents de ses fichiers d'environnement par défaut). Ces fichiers représentent :

- la liste des simulateurs accessibles par l'utilisateur
- les plans de conception qu'il a créé

3.3. Configuration du "design kit"

Le répertoire "design kit directory" contient la liste des éléments utilisés par l'utilisateur pour réaliser une synthèse (principalement, les technologies). Par défaut, ce répertoire est situé au même niveau que le répertoire d'installation de RUNE II.

3.4. Configurations complémentaires

Une fois les répertoires de l'environnement de travail configurés, l'utilisateur doit renseigner le compilateur JAVA qu'il veut utiliser, ainsi qu'un éditeur de texte par défaut (*emacs*, *nedit*, *notepad*...).

La fenêtre permettant de saisir ces informations sera présentée plus loin dans le manuel (Partie 3 – Configuration de l'environnement).

L'ensemble de ces informations est finalement sauvegardées dans un fichier *.runrc.xml* situé à la racine du compte utilisateur.

4. Lancement

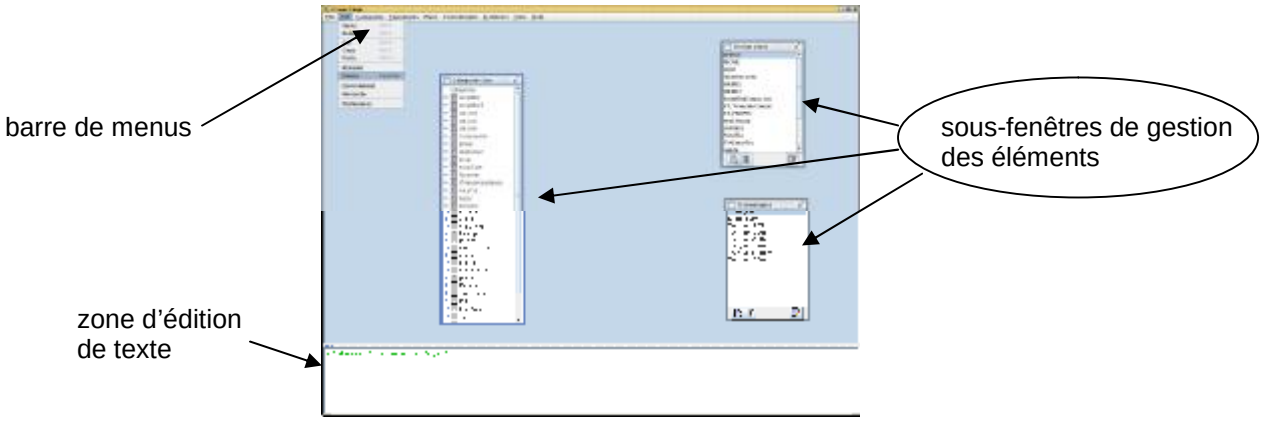
Pour lancer l'application, l'utilisateur doit utiliser soit (sous UNIX) le script run.sh, soit (sous Windows) le script run.bat ou le fichier Rune.jar (tous ces éléments se situent dans le répertoire d'installation de RUNE II, *rune-v2.xx*).

Lorsqu'un utilisateur lance RUNE II, l'application charge l'environnement de travail de cet utilisateur, ainsi que tous les éléments qu'il aura déjà créé au cours de ses utilisations ultérieures (catégories, topologies, plans de conception...).
Cependant, si l'utilisateur ne possède pas de fichier *.runrc.xml* à la racine de son compte utilisateur, l'application le force à configurer son environnement de travail pour pouvoir continuer.

Partie 2 – Aperçu de RUNE II

1. Interface Graphique Utilisateur – GUI

L'interface graphique de RUNE II a l 'aspect suivant :



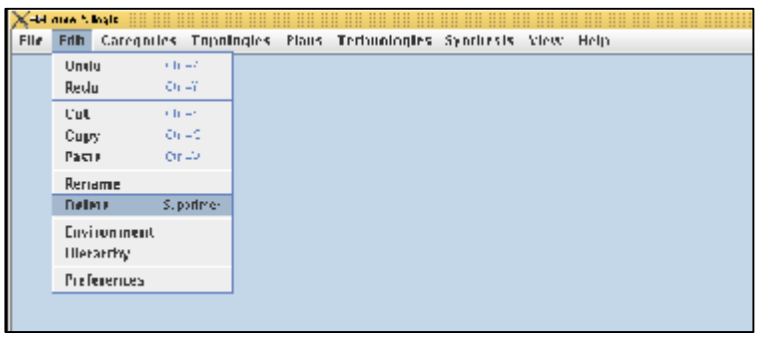
Elle est composée d'une barre de menus principale, d'une zone d'édition de texte, et de trois sous-fenêtres :

- une pour la gestion des catégories et des topologies
- une pour la gestion des plans de conception
- une pour la gestion des technologies

1.1. Barre de menus

1.1.1. Présentation générale

La barre de menus principale permet la gestion de l'ensemble de l'application (sauvegarde, fermeture, couper/copier/coller...), mais également de la création des éléments (catégories, topologies, plans ce conception, technologies), ainsi que la configuration et le lancement d'une synthèse. La barre de menus a l'aspect ci-contre :



Elle se compose alors d'un ensemble de menus :

- File
- Edit
- Categories (permet la gestion des éléments de types *category*)
- Topologies (permet la gestion des éléments de types *topology*)
- Plans (permet la gestion des plans de conception)
- Technologies (permet la gestion des technologies)
- Synthesis (permet de configurer / charger et lancer une synthèse)
- View (permet de rendre visible, ou non, certaine fenêtre de l'application)
- Help

1.1.2. Description des différents menus

- Le menu *File* permet de :
  - sauvegarder toutes les fenêtres d'édition de catégories et topologies ouvertes
  - fermer toutes les fenêtres d'édition de catégories et topologies ouvertes
  - quitter l'application

- Le menu *Edit* permet de :
  - couper / copier / l'élément sélectionné
  - coller le dernier élément coupé / copié
  - renommer l'élément sélectionné
  - effacer l'élément sélectionné
  - configurer l'environnement
  - visualiser la hiérarchie de la topologie sélectionnée
  - configurer les préférences utilisateur
  - indiquer si les fichiers JAVA doivent être créés et compilés à l'importation d'éléments

- Le menu *Categories* permet de :
  - créer une nouvelle catégorie
  - éditer la catégorie sélectionnée
  - exporter la catégorie sélectionnée (avec ou sans topologies associées)
  - importer une topologie sous la catégorie sélectionnée

- Le menu *Topologies* permet de :
  - créer une nouvelle topologie sous la catégorie sélectionnée
  - éditer la topologie sélectionnée
  - exporter la topologie sélectionnée

- Le menu *Plans* permet de :
  - créer un nouveau plan de conception
  - éditer le plan sélectionné
  - supprimer le plan sélectionné

- Le menu *Technologies* permet de :
  - créer une nouvelle technologie
  - éditer la technologie sélectionnée
  - supprimer la technologie sélectionnée

- Le menu *Synthesis* permet de :
  - configurer une synthèse sur les éléments sélectionnés
  - lancer une synthèse
  - charger un fichier de configuration de synthèse
  - sauvegarder la dernière synthèse configurée sur les éléments sélectionnés dans un fichier XML
  - configurer le mode de visualisation des résultats de la synthèse
  - indiquer l'utilisation ou non de la base de données

- Le menu *View* permet de :
  - rendre visible ou non l'arbre des catégories
  - rendre visible ou non la fenêtre des plans de conception
  - rendre visible ou non la fenêtre des technologies
  - rendre visible ou non l'affiche des résultats
  - rendre éditable ou non la zone de texte (situé en-bas de la fenêtre)

- Le menu *Help* permet d'accéder à la rubrique d'aide de RUNE II

1.1.3. Raccourcis clavier

Certaines des actions de la barre de menus sont accessibles via des raccourcis clavier. Le tableau suivant présente l'ensemble de ces raccourcis :

Raccourci	***Action***
Alt-F	déployer le menu *File*
Ctrl-S	sauvegarder toutes les fenêtres ouvertes
Ctrl-W	fermer toutes les fenêtres ouvertes
Ctrl-Q	quitter l'application
Alt-E	déployer le menu *Edit*
Ctrl-Z	annuler la dernière action
Ctrl-Y	refaire la dernière action annulée
Ctrl-X	couper l'élément sélectionné
Ctrl-C	copier l'élément sélectionné
Ctrl-V	coller un élément
Del / Suppr	supprimer l'élément sélectionné
Alt-C	déployer le menu *Categories*
Alt-T	déployer le menu *Topologies*
Alt-S	déployer le menu *Synthesis*
Ctrl-R	lancer une synthèse
Alt-V	déployer le menu *View*
Alt-H	déployer le menu *Help*
F1	lancer l'aide de l'application

1.2. Zone d'édition de texte

La zone d'édition de texte se situe dans la partie inférieure de la fenêtre. Elle permet la saisie de texte. Ce texte pourra ensuite être sauvegardé sous la forme d'un fichier texte (au format *.txt*).
Mais elle permet également d'ouvrir un fichier texte, le contenu du fichier est alors ajouté dans cette zone.

Remarque : la possibilité d'édition de cette zone peut être activée ou non par l'utilisateur (View > Set text area editable).

1.3. Les différentes sous-fenêtres

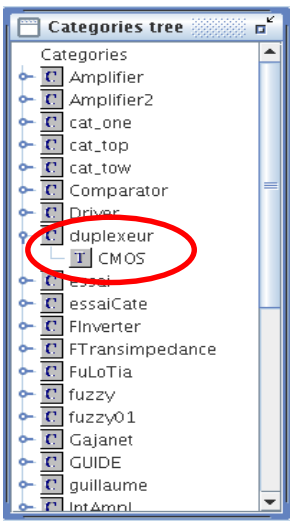
Les trois sous-fenêtres permettent une gestion plus précise des différents éléments (catégories / topologies, plans de conception et technologies), puisqu'elles permettent, en plus de leur création, leur définition, leur modification et leur suppression.

Plus de détails sur la composition chacun de ces éléments seront fournis dans la partie 4 (*Partie 4 – Mise en place de blocs IP*).

1.3.1. Arbre des catégories

La fenêtre des catégories se présente sous la forme d'un arbre. Les noeuds de premier niveau représentent les catégories crées par l'utilisateur, les noeuds de second niveau les topologies associées.

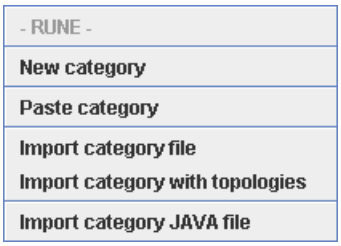
L'image ci-dessous montre un exemple d'arbre des catégories :



Sur cet exemple, ***duplexeur*** est une catégorie créée par l'utilisateur, et ***CMOS*** une topologie associée à la catégorie ***duplexeur***.

Les actions pour chacun des types d'éléments (catégorie / topologie) sont accessibles via un menu contextuel que l'utilisateur peut obtenir par un simple clic droit sur cet élément.

Sur la racine (*Categories*), l'utilisateur peut :



- créer une nouvelle catégorie
- coller une catégorie
- importer une catégorie depuis un fichier XML (avec ou sans topologies associées)
- importer une catégorie depuis un fichier JAVA

Sur un élément de type *Category*, l'utilisateur peut :

- ajouter une nouvelle topologie
- importer une topologie depuis un fichier XML
- éditer la catégorie correspondante
- copier/couper la catégorie
- coller une topologie
- renommer la catégorie
- effacer la catégorie
- importer une topologie depuis un fichier JAVA
- exporter la catégorie dans un fichier XML
- exporter la catégorie dans un fichier JAVA.

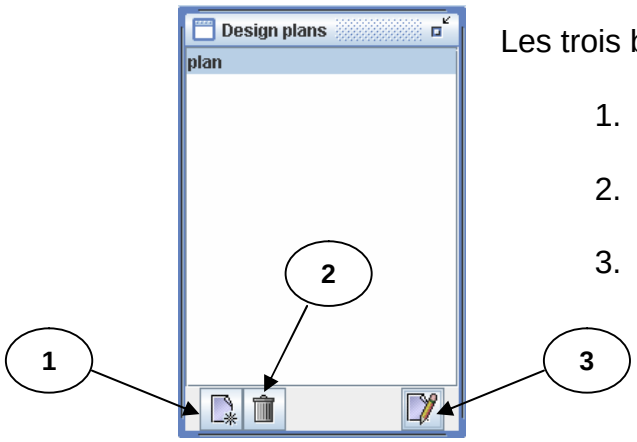
Sur un élément de type *Topology*, l'utilisateur peut :

- éditer la topologie
- copier/couper la topologie,
- renommer la topologie
- effacer la topologie
- exporter la topologie dans un fichier XML
- exporter la topologie dans un fichier JAVA

Finalement, par double-clic sur un élément, l'utilisateur peut avoir accès à la fenêtre d'édition de ce dernier. Les différentes fenêtres d'édition seront présentées plus loin dans le manuel (*Partie 4 – Mise en place de blocs IP*).

1.3.2. Plans de conception & Technologies

La fenêtre de gestion des plans de conception et celle des technologies ont le même aspect. Elles se présentent toutes deux sous la forme d'une liste des éléments déjà existants. La fenêtre de gestion des plans de conception a l'aspect suivant :



Les trois boutons permettent:

1. d'ajouter un nouvel élément
2. de supprimer l'élément sélectionné
3. de définir l'élément sélectionné

L'ajout d'un plan de conception consiste à saisir le nom du nouveau plan, et ainsi ajouter ce nom dans la liste.

⇒ **cliquez sur le bouton *New* (*bouton 1*)**
⇒ **entrez le nom du plan de conception (ou de la technologie), puis validez**
⇒ **double-cliquez sur l'élément correspondant dans la liste, ou sélectionnez l'élément dans la liste et cliquer sur le bouton *Update* (*bouton 3*)**

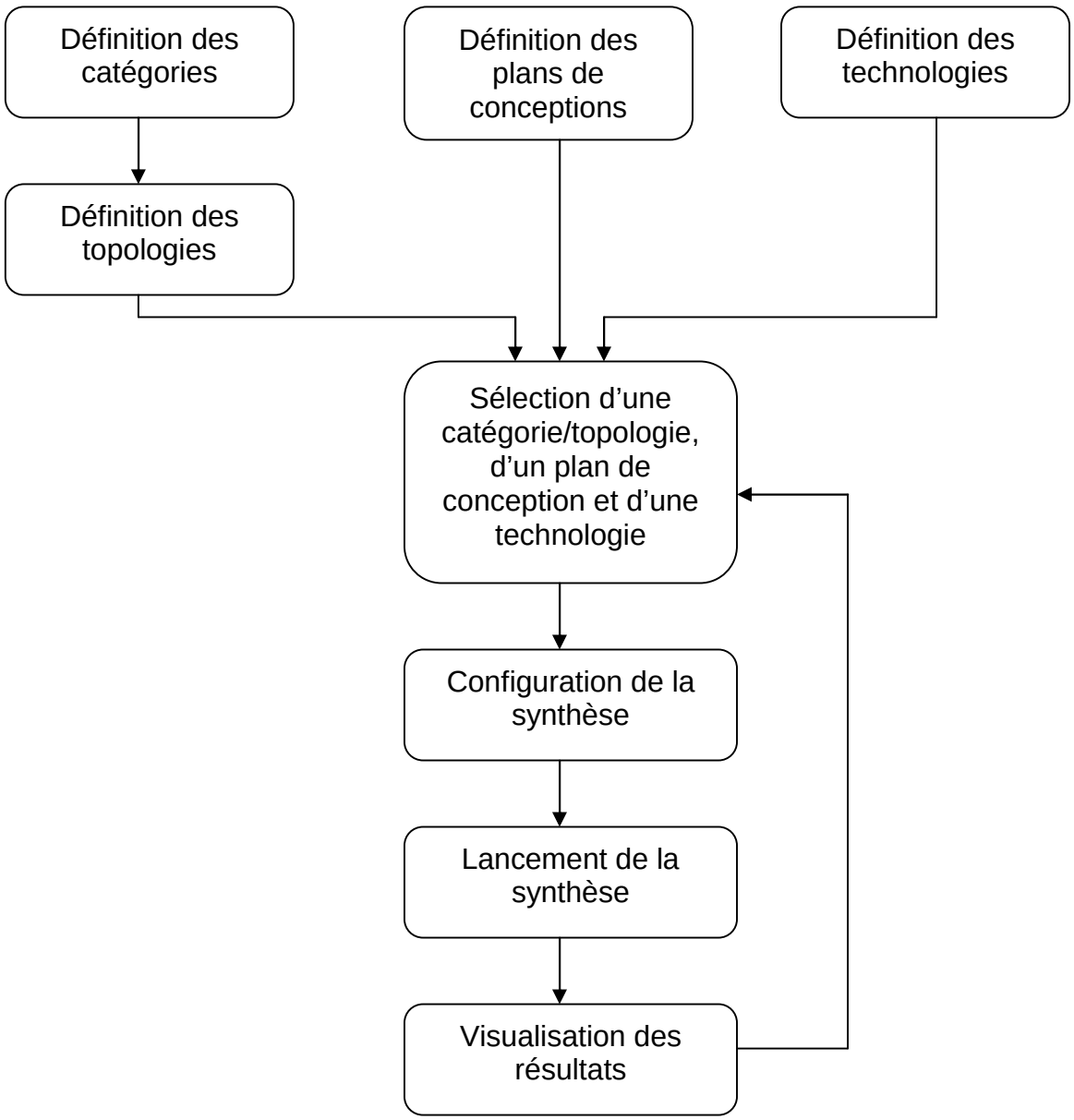
Ensuite, l'utilisateur doit définir ce nouveau plan de conception. Les fenêtres de définitions des plans de conception et des technologies seront présentées en partie 3 et partie 4.

1.4. Phases

Pour pouvoir lancer une synthèse, l'utilisateur doit suivre un ordre particulier dans la création des différents éléments.

1. Dans un premier temps, il doit définir une catégorie.
2. Ensuite, il doit définir l'ensemble des topologies associées (au moins une).
3. Une fois ces éléments définis, il doit choisir un plan de conception, ou en définir un nouveau
4. Quatrièmement, il doit sélectionner la technologie à utiliser.
5. Avant de pouvoir lancer une synthèse, il doit configurer les différentes valeurs des spécifications.

Le schéma ci-dessous représente le processus de lancement d'une synthèse (et plus généralement, le processus d'utilisation de RUNE II).



De plus, certaines options concernant la synthèse peuvent être renseignées par l'utilisateur :

- Vitesse de calcul (*calculation speed*) : temps d'attente entre chaque itération de calcul (plus ce temps est faible, plus la synthèse sera rapide)

- Taux de rafraîchissement (*rate refresh*) : taux de rafraîchissement, c'est à dire le nombre d'itérations au bout duquel l'affichage est rafraîchit (si ce taux est égal à 1, l'affichage est rafraîchit toutes les itérations)

2. XML – eXtensible Markup Language

Afin d'assurer la portabilité de l'application, RUNE II utilise à présent le format de fichier XML pour sauvegarder ces données.

2.1. Import/Export format

RUNE II donne la possibilité d'exporter une catégorie ou une topologie pour ensuite pouvoir l'importer dans un autre environnement de travail.

Remarque : RUNE II utilise la même technique pour réaliser l'import/export que pour le couper/copier/coller.

2.1.1. Export

L'utilisateur de RUNE II peut exporter les catégories et topologies qu'il a créées sous la forme de fichiers au format XML.

Il peut alors choisir d'exporter :

- uniquement une catégorie sous la forme d'un fichier dont il renseignera le nom et le chemin d'accès

- uniquement une topologie sous la forme d'un fichier dont il renseignera le nom et le chemin d'accès

- une catégorie et ses topologies associées, et dans ce cas là, il renseignera le répertoire de stockage (par défaut, ce répertoire aura le nom de la catégorie correspondante)

Les noms de fichiers d'exportation (répertoire ou non) sont à choisir dans une fenêtre de sélection de fichiers. Si toutefois le nom est incorrect, l'exportation ne pourra avoir lieu. Le nom du fichier d'exportation correspondra au nom de l'élément après importation.

*Exemple : si l'utilisateur exporte la catégorie **duplexeur** dans le fichier **duplexeur_exported.xml**, lorsqu'il réimportera cette catégorie, elle prendra automatiquement le nom de **duplexeur_exported**.*

2.1.2. Import

Comme l'export de catégories/topologies sous la forme de fichiers XML est maintenant possible, l'utilisateur peut également importer ces éléments.

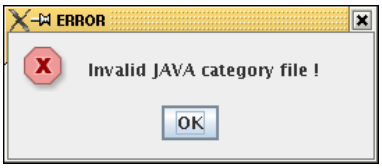
De la même manière que pour l'export, il est possible d'importer :

- une catégorie seule, dont l'utilisateur renseignera le nom de fichier de stockage

- une topologie seule (sous une catégorie existante), dont l'utilisateur renseignera le nom de fichier de stockage

- une catégorie avec ses topologies associées, dont l'utilisateur aura renseigné le répertoire de stockage

Si les fichiers que l'utilisateur tente d'importer sont incorrects, l'application l'en informera par un message d'erreur du type :



et les éléments ne seront pas importés.

De plus, comme nous l'avons vu précédemment, le nom de l 'élément après importation correspond au nom du fichier importé.

2.2. Fichiers générés

Les fichiers utilisés pour l'import/export sont des fichiers au format XML. Ils sont formatés de la même manière que les fichiers de catégories et de topologies.

Un exemple de fichier généré par RUNE II pour une catégorie et une topologie associée est présent en Annexe A.

2.3. Visualisation / Edition de fichiers

Un des principaux avantages de l'utilisation du format de fichier XML, est que ces fichiers sont de simples fichiers textes. Donc, ils sont portables, et éditables.

Ainsi, les utilisateurs peuvent éditer les fichiers catégories ou topologies qu'ils auront créés avec un simple éditeur de texte, et les modifier sans être obliger de passer par le GUI de RUNE II. Ils pourront également éditer et modifier ces fichiers à l'aide d'éditeur XML, comme JAXE (http://jaxe.sourceforge.net).

3. UML - Unified Modelling Language

3.1. Principle

3.2. Linking UML tools to RUNE

Partie 3 - Configuration de l'environnement

1. Préférences utilisateurs

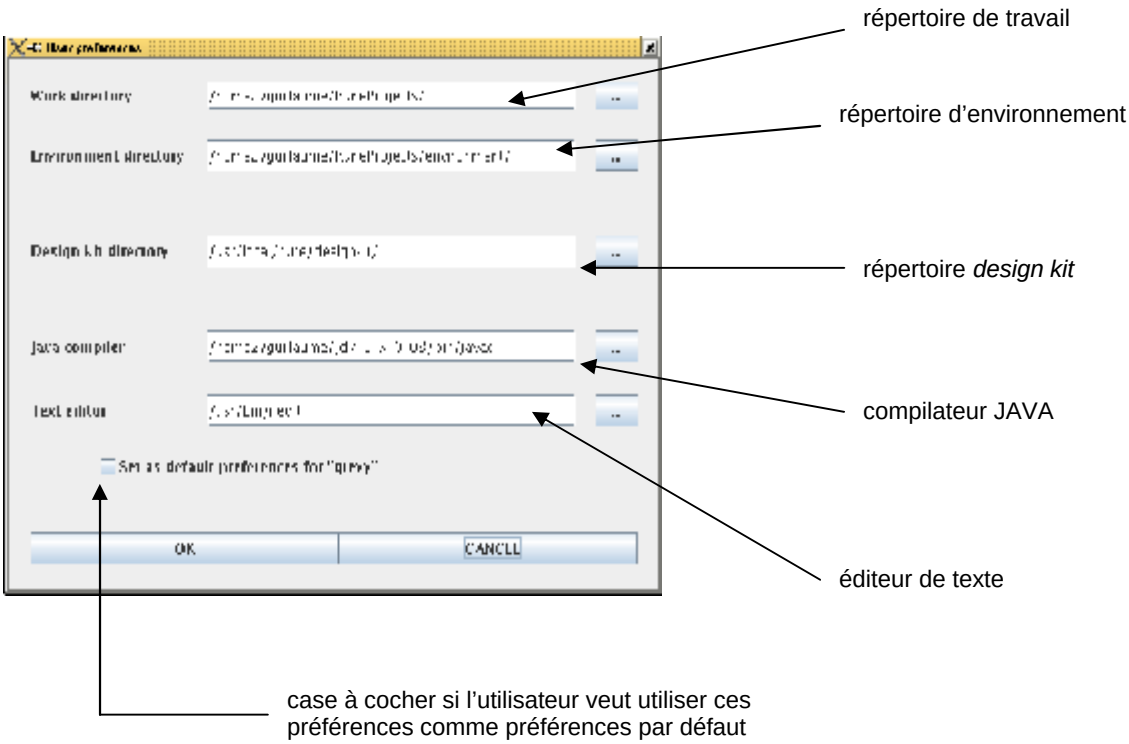
Les préférences utilisateurs sont composées de :

- l'environnement de travail (ensemble de répertoires)
- du compilateur JAVA
- de l'éditeur de texte.

Toutes ses informations sont renseignées par les utilisateurs lors de leur première utilisation de RUNE II, et enregistrées dans un fichier ***.runerc.xml*** à la racine du compte utilisateur. Ce fichier est ensuite chargé chaque fois que l'utilisateur lance RUNE II.

Cependant, ils peuvent modifier ces préférences à tous moments. RUNE II recharge les éléments en fonction de ces nouvelles préférences.

La fenêtre permettant ses modifications a l'aspect suivant :



L'utilisateur peut alors modifier son répertoire de travail (*work directory)*. Dans ce cas, le répertoire d'environnement (*environment directory*), se situant par défaut dans le répertoire de travail, est également modifié.
Il peut également modifier le répertoire "*design kit*", le compilateur et l'éditeur de texte.

Les noms de fichiers peuvent être saisie au clavier ou dans une fenêtre de sélection de fichiers. Leur validité sera vérifiée à la validation (grâce au bouton *"OK"*) par l'utilisateur. Si au moins un élément n'est pas correct (ou n'existe pas), la validation ne peut pas avoir lieu.

Une fois validé, RUNE II recharge complètement l'environnement.

Mais ces modifications sont prises en compte uniquement pendant que RUNE II est ouvert. Une fois fermé, elles ne seront pas sauvegardées. L'utilisateur peut alors choisir d'utiliser ces préférences comme préférences par défaut (et dans ce cas, sélectionnez "*Set as default preferences for **user***"), et elles seront alors sauvegardées dans le fichier *.runerc.xml.*

Par ailleurs, d'autres renseignements sont ajoutés dans le fichier *.runrc.xml :*

- la vitesse de calcul
- le taux de rafraîchissement de l'affichage des résultats
- l'état (visible/non visible) des trois sous-fenêtres
- l'indicateur d'utilisation de la base de données
- l'indicateur de l'affichage des résultats de la synthèse
- l'indicateur de possibilité d'édition de la zone de texte

Exemple de fichier .runerc.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- CONFIGURATION FILE -->
<!-- Automatically generated by RUNE -->
<configuration database="false" designKitDirectory="C:\Documents and
Settings\Guillaume\Mes documents\UTBM\ST50 - ECL\designkit\" editable="false"
editor="C:\WINDOWS\notepad.exe" environmentDirectory="C:\Documents and
Settings\Guillaume\Mes documents\UTBM\ST50 - ECL\designkit\environment\"
form="true" javaCompiler="C:\Program Files\Java\jdk1.5.0\bin\javac.exe" plans="true"
rate="0" speed="0" techno="true" trame="false" tree="true" user="Guillaume"
workDirectory="C:\Documents and Settings\Guillaume\RuneProjects\" />
```

Ainsi, lorsque l'utilisateur utilisera à nouveau RUNE II, l'application sera dans le même état que lorsqu'il l'aura fermé.

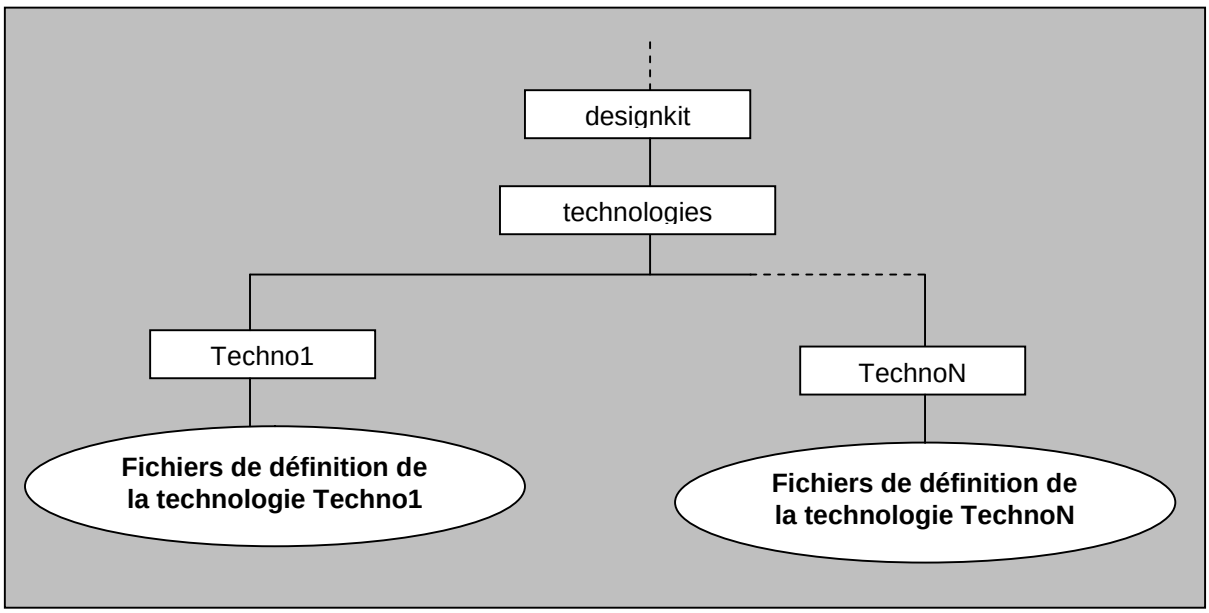
2. Simulateurs

3. Technologies

Les technologies sont également parties intégrantes de l'environnement. Elles représentent les technologies utilisées par les utilisateurs pour réaliser une synthèse. Cette liste de technologies est commune à tous les utilisateurs. Mais chaque utilisateur peut également la modifier, en ajoutant d'autres technologies dont il aurait besoin, ou bien en supprimant des technologies inutiles.

3.1. Stockage des technologies

Une technologie est caractérisée par un ensemble de fichiers. Ces fichiers se trouvent dans un répertoire au nom de la technologie, qui lui se trouve dans le répertoire technologies/ du design kit, comme l'illustre le schéma suivant :



3.2. Création et définition d'une technologie

Pour créer une technologie, après avoir cliqué sur le bouton *New* (1) de la fenêtre des technologies, l'utilisateur doit saisir son nom dans une fenêtre de saisie, qui a la forme ci-contre :

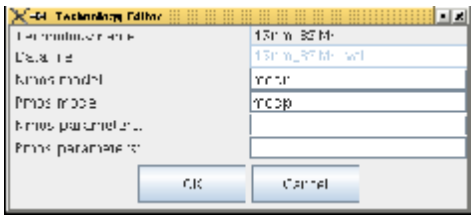
Remarque : la fenêtre de saisie du nom d'une technologie a le même aspect que la fenêtre de saisie du nom des autres types d'éléments.

Lors de la création d'une technologie, un répertoire au nom de cette technologie est créé dans le répertoire *technologies/* du *design kit*.

Une fois créée, la fenêtre de définition s'ouvre automatiquement (cette fenêtre peut également être ouverte par l'utilisateur grâce au bouton *Update* (3)).

Pour définir une technologie, l'utilisateur doit renseigner un certain nombre de paramètres.

La fenêtre permettant cette définition a l'aspect ci-contre :



Paramètres caractérisant une technologie

Comme vous pouvez le voir, une technologie est caractérisée par :

- un nom
- un fichier de données (*Data file*)
- un modèle NMOS correspondent à la technologie (*Nmos model*)
- un modèle PMOS correspondent à la technologie (*Pnmos model*)
- des paramètres à passer au modèle NMOS (*Nmos parameters*)
- des paramètres à passer au modèle PMOS (*Pnmos parameters*)

Le fichier de données doit avoir pour extension *val* (par exemple, *45nm_BSMI4.val*), et doit se situé dans le répertoire de la technologie. Ce fichier peut être vide.

Cependant, le fichier du même nom, mais d'extension *vale* (par exemple, *45nm_BSMI4.vale*), ne doit, lui, pas être vide. En effet, c'est ce fichier qui contient les éléments caractérisant la technologie.

De plus, lorsque l'utilisateur utilise un simulateur (par exemple, *spectre*), il doit également créer un fichier *nomSimulateur_xxx.val* (par exemple *spectre_45nm_BSMI4.val*). Ce fichier, composé d'un ensemble de lignes d'inclusion (compréhensibles par le simulateur), permet, lorsqu'ils sont passés en paramètres du simulateur, d'inclure certains fichiers lors de la simulation.

3.3. Exemple de technologie

Prenons par exemple la technologie 0,35-ams (0,35 µm). Elle est définie de la manière suivante :

- modèle NMOS : modp
- modèle PMOS : modn
- paramètre NMOS : m=1
- paramètre PNOS : m=1

Le fichier de données est ams_035.val. Ce fichier est vide, cependant, le fichier ams_035.vale a le contenu suivant :

```
parameters Kn = 48.788e-6
parameters Kp = 24.067e-6
parameters vtn = 0.4655
parameters vtp = -0.617
parameters an = 50e6
parameters ap = 15e6
parameters Lmin = 0.35e-6
parameters diff = 1.1e-6
parameters Gamma = 0.142
parameters Vea = 6e+7
```

Ce fichier défini la valeur de chacun des paramètres de la technologie.

De plus, le fichier spectre_ ams_035.val doit également être défini par l'utilisateur. Ce fichier peut avoir le contenu suivant :

```
include "/eda/cadence/AMS_3.51_CDS/spectre/c35/mcparams.scs"
include "/eda/cadence/AMS_3.51_CDS/spectre/c35/cmos53.scs" section=cmostm
include "/eda/cadence/AMS_3.51_CDS/spectre/c35/res.scs" section=restm
include "/eda/cadence/AMS_3.51_CDS/spectre/c35/cap.scs" section=captm
include "/eda/cadence/AMS_3.51_CDS/spectre/c35/bip.scs" section=biptm
```

Remarque : pour plus de détails sur le contenu de ce dernier fichier, reportez-vous à la documentation du simulateur.

Partie 4 – Mise en place de blocs IP

La mise en place d'un bloc IP passe par la création d'une catégorie et d'une topologie. Au cours de cette quatrième partie, un exemple présentant la mise en place d'un bloc IP illustrera les explications.

1. Elément *Category*

1.1. Création d'une catégorie

La création d'une catégorie consiste, pour l'utilisateur, à saisir son nom. Une entrée apparaît dans l'arbre des catégories, correspondant à la nouvelle catégorie (cf. Arbre des catégories, p. 9).

⇒ **sélectionnez *Categories* > *Create***
⇒ **entrez le nom de la catégorie (*expl : duplexeur*) et validez**
⇒ **double-cliquez sur le nœud de la catégorie**

Remarque : une nouvelle catégorie ne peut pas avoir le même nom qu'une catégorie existante.

Un répertoire au nom de la catégorie est alors créé dans le répertoire *categories* du répertoire de travail de l'utilisateur (par défaut, *RuneProjects*).

Remarque : un répertoire pour chacune des topologies sera également crée, dans ce répertoire.

1.2. Définition d'une catégorie

Pour un utilisateur, définir une catégorie revient à définir une liste de "performances".

1.2.1. Définition d'une performance

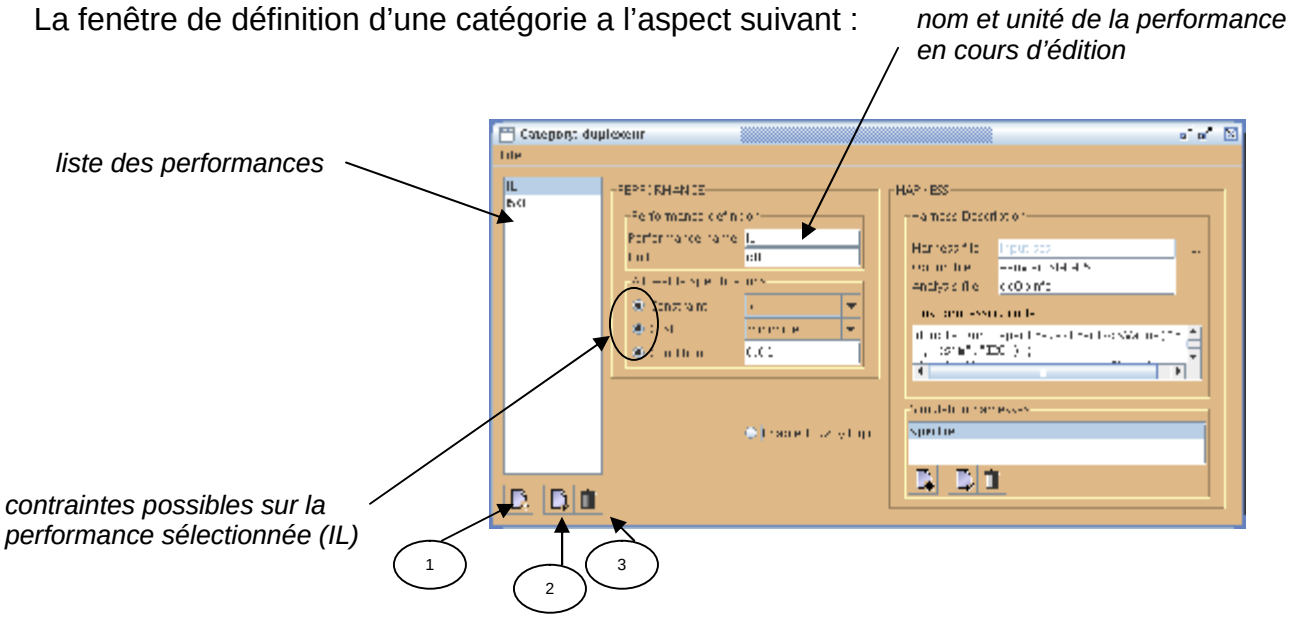
Une performance est définie par :

- un nom
- une unité
- un ensemble de contraintes possibles

Remarque : une contrainte peut-être :
- *inférieure* ou *supérieure*
- *minimisée* ou *maximisée*
- *égale (+/- x, et dans ce cas, on saisira également la valeur de x)*

Par défaut, chaque performance possède la contrainte "égale (+/- 0.0)".

La fenêtre de définition d'une catégorie a l'aspect suivant :



Pour définir une performance, il suffit de remplir les différents champs (nom et unité) et de choisir les contraintes applicables sur celle-ci.

Ensuite, pour sauvegarder la performance, il suffit de presser le bouton *Update* (2), et de confirmer la sauvegarde. La performance est alors ajouté dans la liste.

⇒ **videz le formulaire de saisie des paramètres de la performance (*bouton 1*)**
⇒ **saisissez le nom de la performance : *IL***
⇒ **saisissez son unité : *db***
⇒ **choisissez (sélectionnez) les contraintes possibles sur cette performance**
Constraint : <
Cost : minimize
Condition : saisissez la valeur de la contrainte d'égalité
⇒ **validez la performance (*bouton 2*)**

Remarques :
une fois validez, la performance apparaît dans la liste
vous pouvez créer d'autres performances

Pour mettre à jour une performance, sélectionnez cette performance, modifiez les paramètres et validez.

1.2.2. Définition des *harnais* d'une performance

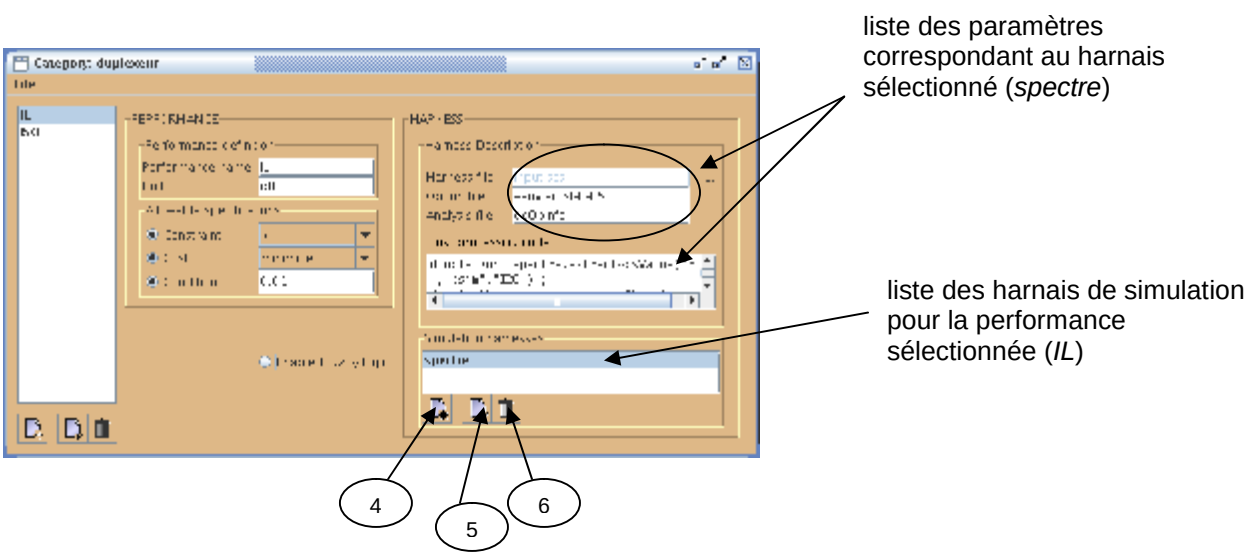
L'utilisateur peut, pour la performance sélectionnée, définir un ensemble de *harnais.*

En fait, une performance peut être évaluée par équation ou par simulation numérique. Dans le cas où une performance est évaluée par simulation, cette liste de harnais représentera la liste des simulateurs utilisables sur la performance.

1.2.2.1. Définition d'un harnais

Un harnais est représenté par :

- un simulateur
- un fichier d'entrée - *harness file* (fichier de paramètres)
- une liste de paramètres à passer au simulateur – *option file* et *analysis file*
- un code *post-processing* (code qui sera exécuté après l'action du simulateur)



Pour définir un harnais, l'utilisateur doit sélectionner les renseignements nécessaires, et valider grâce au bouton *Update* (5). L'utilisateur doit alors sélectionner le simulateur correspondant au harnais qu'il est en train de définir. Ce harnais s'ajoute alors dans la liste des harnais de simulation.

Remarque : pour une performance donnée, un harnais de simulation est identifié par le simulateur qu'il utilise.

Harness file

Ce fichier a pour extension *scs* (par exemple, *input.scs*), et se situe dans le répertoire *netlist* du répertoire de chacune des topologies de la catégorie. Il représente un fichier d'entrée (de paramètres) à passer au simulateur.

Option file

Cette option se présente sous la forme d'une chaîne de caractères et sera passée comme paramètre au simulateur.
Remarque : pour la suite, on considérera cette chaîne de caractères égale à « env-artist 4.4.5 ».

Analysis file

Cette option indique le type d'analyse à utiliser pour la simulation. Pour les simulateurs électriques, comme *Spectre*, l'analyse peut être statique (ac) , temporelle (dc) ou fréquentielle (tran).

Post-processing code

Ce code sera ajouté dans le fichier JAVA correspondant au harnais de la performance. Ce fichier comporte, en fait, une partie permettant la simulation et une partie permettant l'évaluation du code *post-processing.* Ce code sera exécuté après la simulation.

⇒ **videz le formulaire de saisie du harnais de simulation (*bouton 4*)**
⇒ **ouvrez le formulaire de sélection de fichier, puis choisissez un fichier de données (*input.scs*)**
⇒ **saisissez des options à passer au simulateur : *-env artist4.4.5***
⇒ **choisissez le type d'analyse à utiliser pour la simulation**
statique : ac
temporelle : dc
fréquentielle : tran
...
⇒ **remplissez le code post-processing**
⇒ **validez le harnais de simulation (*bouton 5*), et choisissez alors le simulateur correspondant**

Remarques :
le harnais de simulation est ajouté dans la liste
vous pouvez créer d'autre harnais
une fois la liste complète, affectez-la à la performance (bouton 2)

Remarque : en double-cliquant sur la zone de saisie du code post-processing, une

plus grande fenêtre apparaît, permettant une saisie plus aisée du code (il en est de même pour chacune des zones de saisie de code).

1.2.2.2. Affectation des harnais à la performance

Pour affecter cette liste de harnais à la performance sélectionnée, il suffit de mettre à jour la performance.

Parmi ces harnais de simulation, un doit être le harnais par défaut de la performance. Le harnais par défaut de la performance est alors le harnais sélectionné lors de la mise à jour de cette dernière (c'est lui qui sera utilisé pour la simulation).

Lorsque l'utilisateur affecte une liste de harnais à une performance, RUNE II crée, un fichier JAVA correspondant au harnais par défaut, nommée XXSimulationHarness.java (où XX correspond au nom de la performance).

1.2.3. Utilisation de la logique floue

Pour finir, l'utilisateur peut indiquer qu'une synthèse sur un élément de cette catégorie utilisera la logique flou ou non. Pour cela, il doit sélectionner (ou non), le bouton « *Enabled fuzzy logic* »

1.3. Compléments sur la fenêtre de définition des catégories

- Le bouton *1* permet d'effacer complètement tous les champs de saisie de la fenêtre pour pouvoir créer une nouvelle performance.

- Le bouton *3* permet de supprimer, après confirmation, la performance sélectionnée.
- Le bouton *4* permet d'effacer les champs de saisis concernant la zone de définition des harnais de simulation.

- Le bouton *5* permet de supprimer, après confirmation, le harnais de simulation sélectionné.

- Le menu *File* se décompose en trois sous menu :
    - *Save* : sauvegarde la catégorie en cours d'édition dans un fichier XML
    - *Compile all and save* : compile tous les fichiers de harnais, et sauve la catégorie
    - *Generate JAVA file* : génère le fichier JAVA correspondant à la catégorie

2. Elément *Topology*

2.1. Création d'une topologie

De la même manière que pour un catégorie, la création d'une topologie revient à renseigner son nom.

⇒ **sélectionnez *Topologies > Create under selected categories***
⇒ **entrez le nom de la topologie (*expl : CMOS*) et validez**
⇒ **double-cliquez sur le nœud de la topologie**

Une entrée apparaît dans l'arbre des catégories, correspondant à la nouvelle topologie, sous le nœud correspondant à la catégorie à laquelle appartient cette topologie.

Remarque : une topologie ne peut pas avoir le même nom que la catégorie à laquelle elle appartient, ou qu'une topologie de cette catégorie.

2.2. Définition d'une topologie

La définition d'une topologie consiste en :

1. définir un ensemble de dimensions abstraites
2. définir un ensemble de dimensions physiques
3. définir précisément les performances de la topologie
4. définir le code de la topologie
5. définir le code utilisé lors de l'utilisation de la logique floue

2.2.1. Ajout d'une dimension abstraite

L'ajout d'une dimension abstraite se fait à partir de l'onglet "*Dimension*" de la fenêtre.

Les dimensions abstraites représentent les variables de conception qui seront utilisées pour l'optimisation de la fonction *objectif*. Une dimension abstraite est caractérisée par :

- un nom
- une unité
- un type de variation, indiquant comment cette variable varie au cours de la synthèse (linéaire, logarithmique, aléatoire), et à choisir dans le menu déroulant approprié
- un intervalle de variation

⇒ **sélectionnez la liste des dimensions abstraites (la cadre *Abstract* devient rouge)**
⇒ **ouvrez la fenêtre de création des dimensions (*bouton 1*)**
⇒ **saisissez le nom de la dimension (*expl WoL1*) puis valider, le nom apparaît dans la liste**
⇒ **sélectionnez ce nom dans la liste**
⇒ **saisissez l'unité**
⇒ **sélectionnez le type de variation : linéaire (défaut) / logarithmique / aléatoire**
⇒ **saisissez l'intervalle de variation (upper limit / lower limit)**
⇒ **validez la dimension (*bouton 3*)**

2.2.2. Ajout d'une dimension physique

De la même manière que l'ajout de dimensions abstraites, l'ajout de dimensions physiques se fait depuis l'onglet "*Dimension*" de la fenêtre.

Les variables de ce type dépendent des variables abstraites et seront utilisées pour l'évaluation du système.

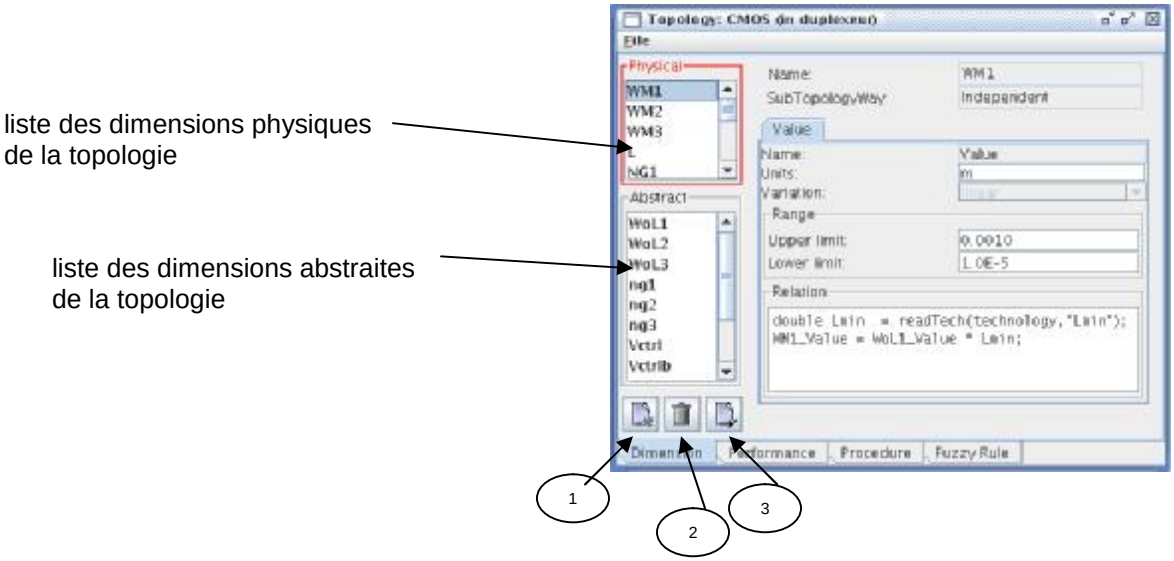
Elles possèdent les mêmes attributs que les variables abstraites. Mais les variables physiques sont également caractérisées par une équation. Cette équation est écrite en JAVA. Elle sera ensuite compilée, et évaluée à chaque évaluation de la topologie.

⇒ **sélectionnez la liste des dimensions physiques (la cadre *Physical* devient rouge)**
⇒ **ouvrez la fenêtre de création des dimensions (*bouton 1*)**
⇒ **saisissez le nom de la dimension (*expl wm1*), choisissez le type (*expl IndependentDouble*) puis valider, le nom apparaît dans la liste**
⇒ **sélectionnez ce nom dans la liste**
⇒ **saisissez l'unité**
⇒ **saisissez l'intervalle de variation (upper limit / lower limit)**
⇒ **remplissez le code de l 'équation correspondant à la dimension**
⇒ **validez la dimension (*bouton 3*)**

Remarque : dans le code JAVA, la valeur d'une dimension peut être obtenu grâce à l'instruction name_Value (où name correspond au nom de la dimension dont on veut connaître la valeur)

L'équation est alors sauvegardée dans un fichier *name_Equation.java*, situé dans le répertoire de la topologie (où *name* représente le nom de la dimension physique). Il est ensuite compilé, et si erreur se produit, l'utilisateur sera prévenu.

La fenêtre de définition d'une variable abstraite ou physique a l'aspect suivant :



De plus, une dimension physique peut être de type numérique, ou bien correspondre à une topologie.

Remarque : dans ce second cas, la dimension physique peut être considérée comme sous-topologie de la topologie à laquelle elle appartient. Il est alors facile de définir une hiérarchie de topologies. Chaque élément de la sous-topologie sera alors défini comme présenté précédemment.

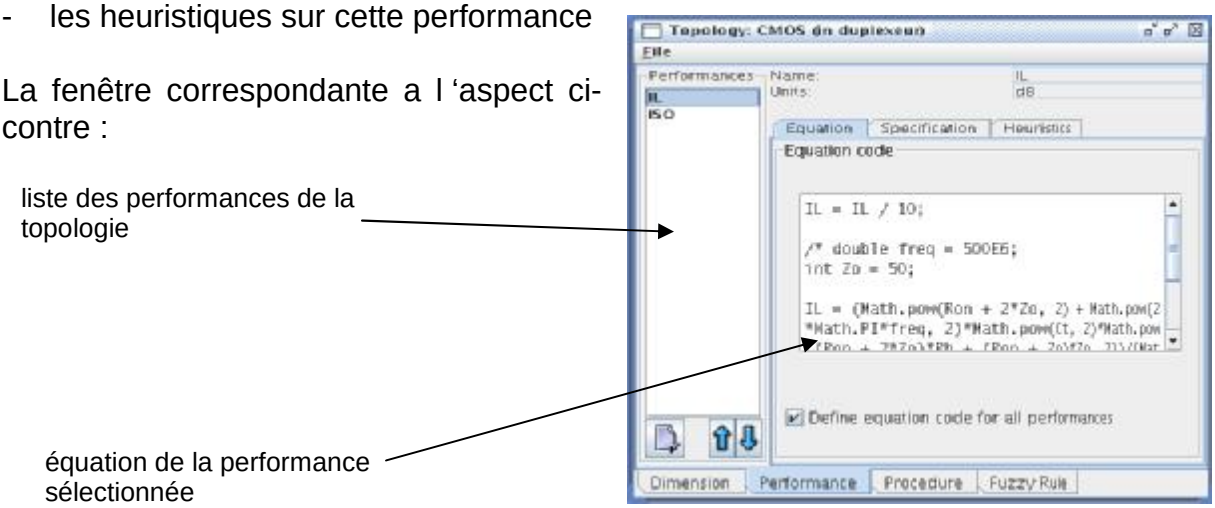
2.2.3. Définition précise des performances

La deuxième étape de la définition d'une topologie consiste à définir chacune de ses performances. Cette étape est réalisable depuis l'onglet "*Performances*" de la fenêtre.
En fait, les performances d'une topologie correspondent aux performances de la catégorie à laquelle elle appartient.

Pour définir plus précisément une performance, il suffit de définir :

- l'équation de cette performance
- la contrainte à utiliser
- les heuristiques sur cette performance

La fenêtre correspondante a l 'aspect ci-contre :



Onglet "Equation"

L'équation d'une performance doit être écrite en code JAVA. Elle sera ensuite compilée et exécutée à chaque évaluation de la topologie.

L'utilisateur peut soit définir un code pour chacune des performances, soit définir un code valable pour l'ensemble des performances (et dans ce cas, sélectionnez "Define equation code for all performances").

⇒ **sélectionnez l'onglet *Equation***
⇒ **sélectionnez une performance dans la liste des performances**
⇒ **remplissez le code correspondant à la performance (ce code doit être fonction des valeurs des dimensions physiques)**
⇒ **validez la performance (*bouton 3*)**

Onglet "Specification"

La contrainte à utiliser sur la performance doit être choisie parmi les contraintes saisies pour cette même performance lors de la création de la catégorie.
De plus, un indicateur, *enabled,* permet d'indiquer si la performance sera prise en compte dans la synthèse, autrement dit, si sa contrainte devra être vérifiée pour que le synthèse se termine avec succès.

⇒ **sélectionnez l'onglet *Specification***
⇒ **sélectionnez une performance dans la liste des performances**
⇒ **choisissez la contrainte à appliquer**
⇒ **indiquez si cette performance devra être prise en compte dans la synthèse (en séleectionnant, ou non, le bouton *enabled*)**
⇒ **validez la performance**

Onglet "Heuristics"

Une heuristique permet de formaliser les connaissances de l'utilisateur concernant le choix du sens de variation d'une dimension abstraite. Une variation peut être croissante ou décroissante ou les deux (cas par défaut).

⇒ **sélectionnez l'onglet *Heuristics***
⇒ **sélectionnez une performance dans la liste des performances**
⇒ **choisissez (dans la liste *Devices*) la dimension sur laquelle vous voulez ajouter une heuristique**

⇒ **choisissez une direction**
  ambigous : croissante / décroissante
  increase : croissante
  decrease : décroissante
⇒ **ajouter l'heuristique**

Remarque : vous pouvez définir d'autres heuristiques

⇒ **indiquez si cette performance utilisera les heuristiques ou non (en sélectionnant le bouton *Use heuristics*)**
⇒ **validez la performance**

Remarque : à chaque validation, le fichier JAVA est mis à jour (ou créé, si il n'existe pas encore) et compilé, et si il contient des erreurs, RUNE II vous l'indiquera.

2.2.4. Ajout d'un code pour la topologie

La définition du code de la topologie se fait depuis l'onglet "*Procedure*" de la fenêtre. De la même manière que l'équation des performances, le code d'une topologie doit être écrit en JAVA. Il sera également compilé puis exécuté lors de la preière évaluation de cette topologie.

⇒ **remplissez le code la procédure (topologie)**
⇒ **sauvegardez la topologie (le fichier JAVA correspondant est alors créé)**

Ce code permet notamment d'initialiser les dimensions abstraites.

2.2.5. Utilisation de la logique floue

Si lors de la création de la catégorie, l'utilisateur a indiqué qu'il désirait utiliser la logique floue (*fuzzy logic*), il doit donc définir les règles de logique floue à utiliser pour la topologie. Cette dernière étape doit être réalisée depuis l'onglet "*Fuzzy Rules*" de la fenêtre.

2.3. Compléments sur le fenêtre de définition des topologies

Le menu *File* se décompose en trois sous menu :

- *Save* : sauvegarde la topologie en cours d'édition dans un fichier XML et compile la procédure

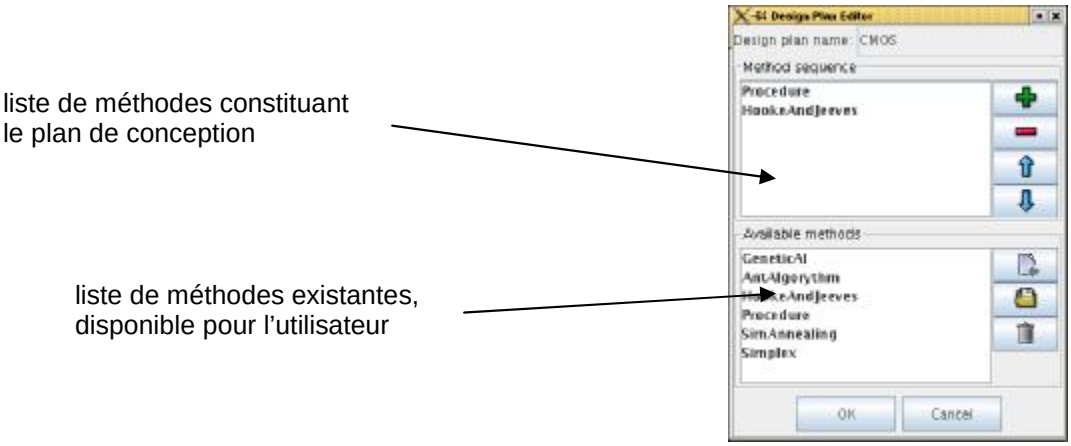
- *Compile all and save* : compile tous les fichiers correspondant aux dimensions physiques, aux performances et à la procédure, puis sauve la topologie

- *Generate JAVA file* : génère le fichier JAVA correspondant à la topologie

3. Plans de conception

Un plan de conception peut être considéré comme une séquence de méthodes. Lors d'une synthèse, chacune des différentes méthodes présentes dans le plan de conception est exécutée, dans l'ordre de leur apparition.

L'utilisateur peut donc créer un plan de conception en saisissant son nom. Ensuite, il doit le définir. La fenêtre de définition d'un plan de conception à l'aspect suivant :



Grâce aux boutons + et -, l'utilisateur peut donc, respectivement, ajouter et supprimer des méthodes existantes dans la plan de conception. De plus, grâce aux boutons « haut » et « bas », il peut réordonner les méthodes au sein d'un même plan de conception (l'ordre étant important, puisque les méthodes seront exécutées dans leur ordre d'apparition dans le plan de conception).

⇒ **double-cliquez sur le plan à définir, ou bien sélectionnez le plan et cliquez sur le bouton *Update* (3)**
⇒ **sélectionnez la première méthode à ajouter dans la liste *Available methods***
⇒ **cliquez sur le bouton +**

Remarque : faites de-même pour les autres méthodes

⇒ **validez le plan de conception (bouton *OK*)**

Compléments :

Une méthode est en fait un algorithme. A l'installation, plusieurs méthodes sont fournies à l'utilisateur (algorithme génétique, simplex, recuit simulé...). Cependant, l'utilisateur peut créer autant de méthodes qu'il le désire.

Une fois la méthode créée, son code JAVA peut être visualiser et modifier dans l'éditeur de texte par défaut. Ensuite, le fichier JAVA la représentant est compilé, pour pouvoir ensuite être utilisé par le moteur.

La méthode de création d'une méthode est présentée en Annexe A.

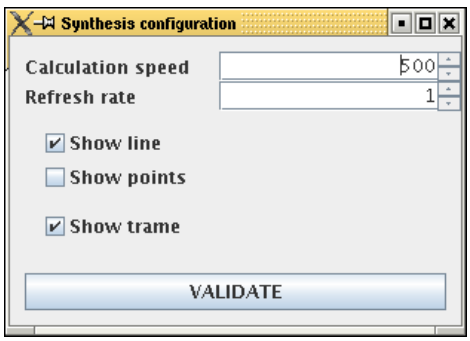
Partie 5 - Synthèse

1. Configuration d'une synthèse

Deux types de configuration peuvent être effectuée pour une synthèse : une configuration générale et une configuration spécifique.

1.1. Configuration générale

La configuration générale consiste à définir un ensemble de paramètres qui pourront être identiques pour chacune des synthèses d'un même utilisateur. La fenêtre de configuration générale a l'aspect suivant :



Cette fenêtre permet donc de renseigner :

- la vitesse de calcul
- le taux de rafraîchissement

Mais elle permet également de définir le mode d'affichage des courbes résultats :

- *Show line* : les courbes résultats seront représentées sous forme de lignes
- *Show points* : les courbes résultats seront représentées sous forme de points, correspondant aux valeurs calculées
- *Show trame* : un quadrillage représentant les échelles de calculs sera également affiché sous la courbe

Remarque :

Sur l'exemple précédent, la vitesse de calcul étant de 500, cela signifie que le moteur attendra 500 ms entre chaque itération de calcul. De plus le taux de rafraîchissement étant de 1, l'affichage sera rafraîchit à chaque itération. Finalement, l'affichage se fera sous la forme de ligne et de points, avec le quadrillage, représentant l'échelle, affiché.

1.2. Configuration spécifique

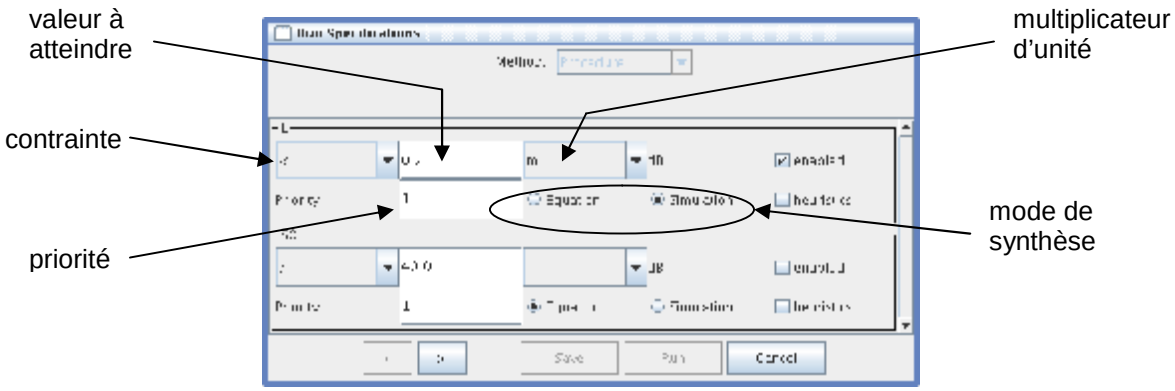
La configuration spécifique, quant à elle, dépend du couple catégorie / topologie, de la technologie et du plan de conception sélectionnés.

Ainsi, l'utilisateur devra, pour chacune des performances de la catégorie, définir :

- la contrainte à appliquer sur cette performance
- la valeur à atteindre
- le multiplicateur d'unité (m,M,G,K…)
- la priorité (0 / 1)
- le mode de synthèse (évaluation/simulation)

De plus, il devra indiquer si cette performance doit être prise en compte dans la synthèse (*enabled*) et si elle utilisera une heuristique ou non (*heuristics*).

La fenêtre suivante permet de faire cette configuration spécifique :



Remarque : une performance est prise en compte dans la synthèse signifie que la contrainte sur cette performance sera vérifiée pour savoir si la synthèse est terminée ou non.

Cette configuration doit être réalisée pour chacune des méthodes du plan de conception. Grâce au flèches < et >, l'utilisateur peut passer d'une méthode à l'autre.

Une fois ces informations renseignées, elles seront sauvegardées dans un fichier de synthèse (au format XML). Ce fichier sera situé dans le répertoire runs du répertoire de travail, et son nom sera constitué du nom de la catégorie, du nom de la topologie et du nom du plan de conception.

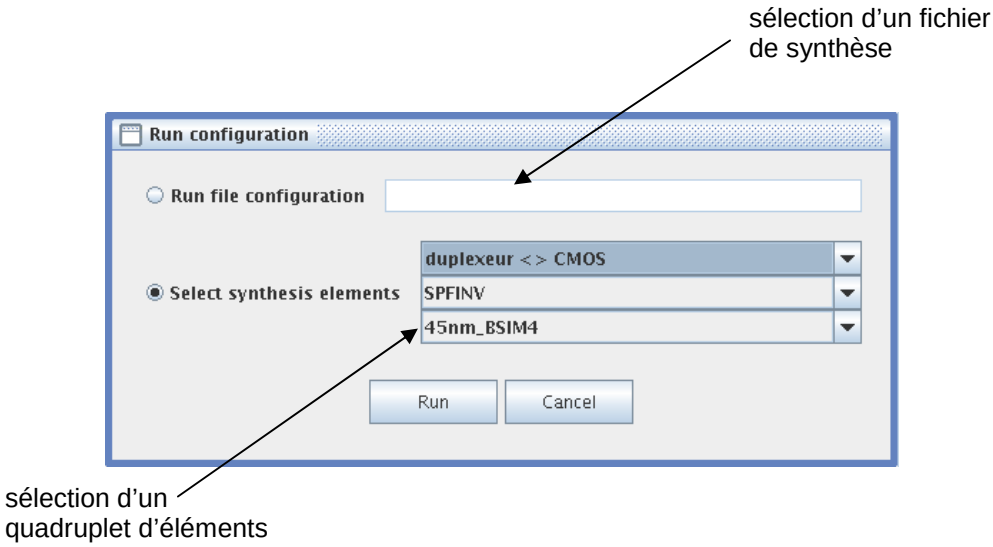
Ce fichier sera chargé au démarrage de la synthèse correspondant à ces éléments.

Exemple : sur la fenêtre précédente, la synthèse sera terminée quand IL era inférieure à 0,7.

2. Lancement d'une synthèse

Pour lancer une synthèse, l'utilisateur doit tout d'abord choisir un couple catégorie/topologie, une technologie et un plan de conception. Mais, il peut également choisir un fichier de synthèse. Ce fichier est en fait un fichier XML dans lequel a été préalablement sauvegardé la configuration d'une synthèse.

La fenêtre suivante permet d'effectuer ce choix :



Lorsque l'utilisateur lance une synthèse sur un quadruplet d'éléments, RUNE II charge le fichier de synthèse correspondant. Si ce dernier n'existe pas, une erreur est générée.

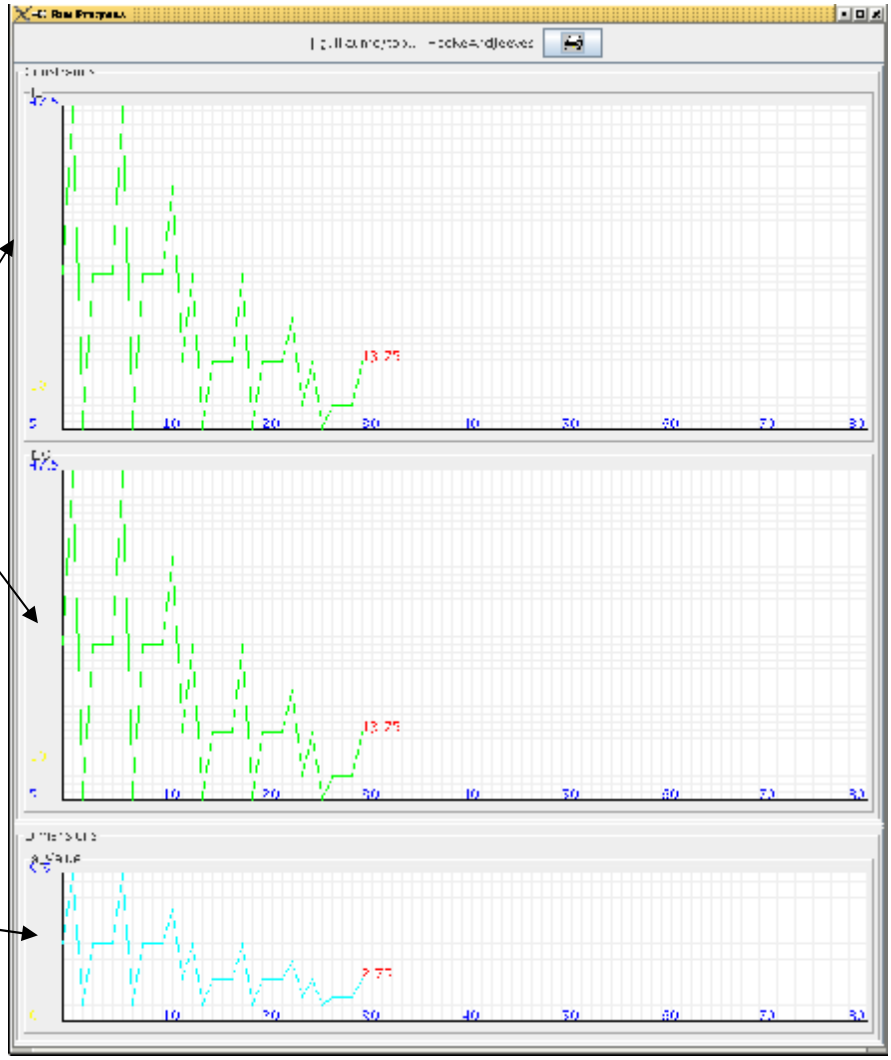
Si l'utilisateur a choisi de visualiser les courbes en même temps que le calcul (*View / Show run progress form* sélectionné dans la barre de menus), RUNE II configure l'affichage. Ensuite, la synthèse proprement dite démarre.

Au démarrage de la synthèse, RUNE II initialise toutes les méthodes du plan de conception. Plus de détails sur le mode d'initialisation d'une méthode et la définition d'une fenêtre d'initialisation sont présentés en Annexe A.

3. Visualisation des résultats

Si l'utilisateur à indiqué qu'il désirait visualiser les resultants en temps reel (indicateur *View > Show run progress form* sélectionné), la fenêtre de visualisation des résultats s'ouvre, avec un affichage pour chacune des performances, et des dimensions abstraites. En fonction de la configuration générale, le calcul se fera plus ou moins vite, l'affichage sera rafraîchit plus ou moins souvent, et les courbes seront présentées sous la forme *de lignes, de poins ou des deux.*

L'affichage pour une performance peut avoir l'aspect suivant :



Partie 6 – Utilisation de la base de données

Pour accélérer la synthèse, RUNE II utilise une base de données. Pour une synthèse donnée, cette base stocke les valeurs calculées à chaque itération.

1er intérêt : si, avant de calculer les valeurs des performances d'une topologie, les valeurs des dimensions abstraites de la topologie se trouvent déjà dans la base (cela signifie que le calcul a déjà été réalisé), dans ce cas, le calcul n'est pas effectué une seconde fois, mais les valeurs des performances sont extraites de la base.

2ème intérêt : si les valeurs que l'on cherche à atteindre pour les performances d'une topologie se trouvent dans la base, l'utilisateur a la possibilité de les extraire directement de cette base. Dans ce cas, uniquement les valeurs résultats lui seront communiquées. Il peut également relancer la synthèse pour réafficher la courbe des résultats.

3ème intérêt : si les valeurs exactes à atteindre des performances n'existent pas dans la base, RUNE II détermine le "point" calculé le plus proche du point recherché. L'utilisateur peut alors décidé de partir de ce point, ou bien de lancer la synthèse normalement.

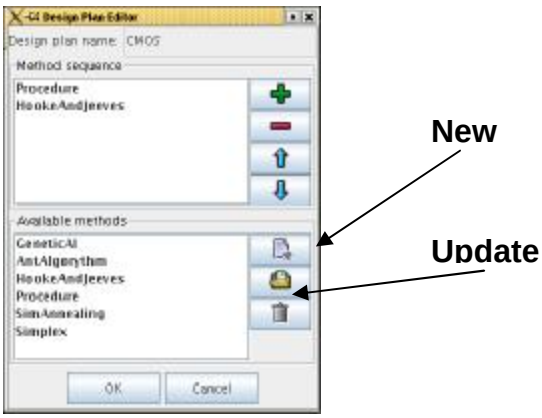
Cette base de données se présente sous la forme de fichiers XML, situés dans le répertoire *database/* du répertoire de travail de l'utilisateur.

Remarque : l'utilisateur peut indiquer si il souhaite utiliser la base de données, grâce à l'indicateur Synthesis > Use database.

Annexe A – Comment créer une méthode ?

1. Création et modification

Pour créer une méthode, il suffit, dans la fenêtre de définition d'un plan de conception cliquez sur le bouton *New*.



Après avoir saisie le nom de la méthode, celui-ci s'ajoute dans la liste des méthodes disponibles et l'éditeur de texte s'ouvre avec le contenu du fichier de méthode automatiquement créer.

L'utilisateur peut alors modifier cette méthode, comme il le souhaite (la méthode entrant directement dans la modification des dimensions est *modifyDimensions*).C'est cette méthode qui contiendra le corps de l'algorithme.

Lorsque l'utilisateur fermera l'éditeur, le fichier sera compilé, et si des erreurs se sont glissées dans le fichier, l'utilisateur en sera averti.

De plus, l'utilisateur peut ajouté une méthode (au sens JAVA) d'initialisation à cet algorithme. Elle doit avoir la forme suivante :

```
public void initialisation()
{
        ...
}
```

Cette méthode sera appelée à chaque début de synthèse. Elle permet d'initialiaser les variables de l'algorithme. De plus, l'utilisateur pourra définir une fenêtre graphique permettant de saisir les valeurs initiales de différentes variables. La création de cette fenêtre sera également appelée dans la méthode *initialisation.*

2. Création d'une fenêtre d'initialisation

Pour définir une fenêtre d'initialisation, l'utilisateur doit créer une classe JAVA. Pour l'exemple, nous appellerons cette XInitializer. Cette classe doit donc se situé dans un fichier JAVA XInitializer.java dans le répertoire *methods* du répertoire d'installation de RUNE II. Le corps due la classe doit avoir la forme suivante :

```
package methods;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class HookeAndJeevesInitializer extends MethodInitializer
{
        public XInitializer(X method)
        {
                super(method,title);
        }

        public void createElements(){
                ...
        }

        public void saveElements(){
                ...
        }
}
```

2.1. Méthode createElements()

Cette méthode permet de créer l'interface graphique de la fenêtre. Elle est composée d'un ensemble de lignes. Chaque ligne est constituée d'un label et d'une zone de saisie de texte. Le texte saisi peut être un entier ou un réel.

Pour ajouter une ligne de saisie d'un entier (avec une valeur initiale) :

```
JFormattedTextField zone;
addLineElements(label,(zone = getIntegerTextField()));

zone.setValue(entier);
```

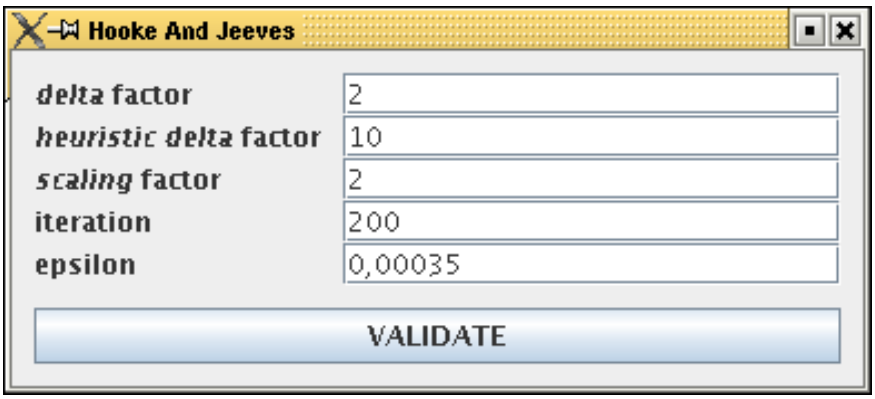
Pour ajouter une ligne de saisie d'un réel (avec une valeur initiale):

```
JFormattedTextField zone;
addLineElements(label,(zone = getDoubleTextField(nbChiffreAprèsLaVirgule)));

zone.setValue(réel);
```

L'exemple ci-dessous permet de créer la fenêtre d'initialisation de la méthode HookAndJeeves.

```
public void createElements()
{
        addLineElements("delta factor",(delta = getIntegerTextField()));
        addLineElements("heuristic delta factor",(heuristic = getIntegerTextField()));
        addLineElements("scaling factor",(scaling = getIntegerTextField()));
        addLineElements("iteration",(maxIt = getIntegerTextField()));
        addLineElements("epsilon",(epsilon = getDoubleTextField(10)));

        try{
            delta.setValue(new Integer(((HookeAndJeeves)method).getDeltaFactor()));
            heuristic.setValue(new Integer(((HookeAndJeeves)method).getHeuristicDeltaFactor()));
            scaling.setValue(new Integer(((HookeAndJeeves)method).getScalingFactor()));
            maxIt.setValue(new Integer(((HookeAndJeeves)method).getMaxIterations()));
            epsilon.setValue(new Double(((HookeAndJeeves)method).getEpsilon()));
        }catch(Exception e){}
}
```

La fenêtre ainsi obtenue a l'aspect suivant :



2.2. Méthode saveElements

Cette méthode est appelée quand l'utilisateur valide la fenêtre (*VALIDATE*). Elle permet d'affecter les valeurs des différentes zones de saisie aux variables de l'algorithme.

L'exemple ci-dessous est la méthode de sauvegarde de la fenêtre d'initialisation de la méthode *HookAndJeeves*.

```
public void saveElements()
{
        try{
            ((HookeAndJeeves)method).setDeltaFactor(Integer.parseInt(delta.getValue().toString()));

((HookeAndJeeves)method).setHeuristicDeltaFactor(Integer.parseInt(heuristic.getValue().toString()));
            ((HookeAndJeeves)method).setScalingFactor(Integer.parseInt(scaling.getValue().toString()));
            ((HookeAndJeeves)method).setMaxIterations(Integer.parseInt(maxIt.getValue().toString()));
            ((HookeAndJeeves)method).setEpsilon(Double.parseDouble(epsilon.getValue().toString()));
        }catch(Exception ex){}
}
```

Annexe B – GNU/General Public Licence