Développements et interfaçages de PROSE 3.5

Stéphanie EVEN[1], Ronan KERYELL[2], Nicolas FLIPO[1], Michel POULIN[1]

[1] Centre d'Informatique Géologique, ENSMP, 35 rue Saint-Honoré, 77 305 FONTAINEBLEAU cedex

[2] Laboratoire d'Informatique et Télécommunications, ENST Bretagne, Technopôle Brest Iroise, 29280 PLOUZANE

1. Introduction

Le logiciel PROSE est développé depuis de nombreuses années au CENTRE D'INFORMATIQUE GÉOLOGIQUE (CIG), pour permettre une analyse fine du comportement de systèmes aquatiques en réponse à diverses actions anthropiques. Ce logiciel fait aujourd'hui partie de la boîte à outil du PIREN SEINE.

Le logiciel PROSE est destiné à pouvoir traiter de l'impact de tout type de pollution anthropique sur une large gamme de systèmes. Dans tout type de pollution anthropique, nous entendons des pollutions permanentes, comme par exemple les rejets de station d'épuration, ou transitoires, tels que les rejets urbains de temps de pluie. Nous entendons également pouvoir traiter de pollutions dites ponctuelles ou diffuses.

Pour répondre à ces objectifs, les choix conceptuels ont été réalisés dans le but de pouvoir représenter le plus correctement possible les échelles temporelles et spatiales fines, tout en couvrant correctement le champ d'impact des pollutions étudiées.

Or la qualité des résultats obtenus dépend largement de la qualité des données introduites dans le modèle. L'objectif de travailler à des petites échelles de temps (phénomènes transitoires de temps de pluie) et d'espace (champ proche des rejets) oblige à traiter des données à la précision désirée, à l'échelle de l'heure ou de la journée, depuis la centaine de mètres jusqu'au kilomètre.

Or la quantité d'information à manipuler devient rapidement importante et cela constitue l'étape limitante pour l'utilisation d'un tel logiciel. L'extension du domaine traité aux bassins amont et le désir de traiter les bassins versants en lien avec les rivières, étend également le champ d'investigation, augmentant d'autant la quantité de données. Introduire la notion de données géoréférencées devient nécessaire, ainsi que l'utilisation du potentiel offert par les SIG.

Depuis quelques années, outre les développements conceptuels, les développements informatiques relatifs aux interfaçages (pré-traitement et post-traitement) sont au cœur des projets de développements.

Deux axes ont été poursuivis cette année :

- l'interfaçage des entrées et la poursuite du développement de l'outil GABI ;
- exploration de la problématique du portage sous système d'exploitation WINDOWS.

2. L'interfaçage des entrées : le projet GABI

2.1. Brefs rappels sur le projet GABI

Les interfaces pour gérer les données d'entrées ont pour but de faciliter la création par l'utilisateur des fichiers de données qui sont nécessaires au fonctionnement d'un modèle.

Les fonctions pouvant être dévolues à de telles interfaces sont :

- offrir une vue d'ensemble des données nécessaires en entrée du logiciel ;
- présenter *a priori* à l'utilisateur les formats définis pour ces entrées, l'utilisateur n'ayant plus qu'à renseigner certains champs, soit sous forme de texte (nom de fichier, valeur, etc ...), soit sous forme de listes de choix (choix de variables, etc...) ;
- faciliter la manipulation de formats différents (ascii, XML, ...) ;
- présenter des procédures de contrôle permettant de savoir *a priori* si les données sont au bon format.

Dans l'état actuel des choses, les interfaces sont généralement écrites au cas par cas. Leur conception doit être adaptée dès qu'une modification a lieu dans le logiciel interfacé. Le risque est grand alors, soit de figer complètement les choses et de limiter les perspectives d'évolution d'un logiciel, soit de passer un temps énorme dans la gestion de l'information.

Le projet GABI envisage d'automatiser la conception de l'interface en concevant un outil qui sera capable d'analyser la partie code de lecture des données d'entrée d'un logiciel et de générer le code d'interface automatiquement. Pour préciser les choses, nous dirons que l'outil GABI est un générateur d'éditeur de format pour le logiciel scientifique auquel il est appliqué.

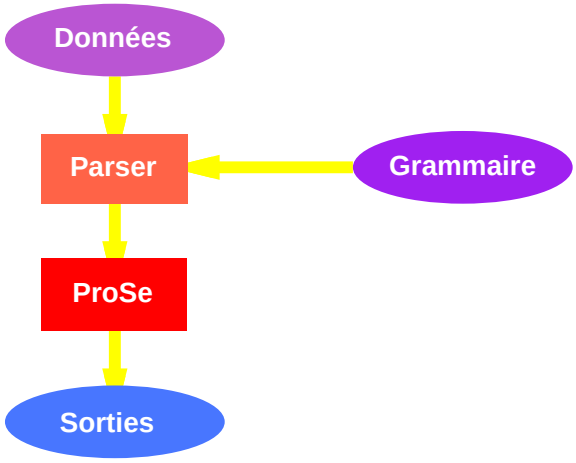

Figure 1: *Représentation schématique des fonctionnalités d'un logiciel de calcul scientifique.*

Cet outil est développé depuis quelques années dans le cadre d'une collaboration entre le Centre d'Informatique Géologique (CIG) à l'École des Mines de Paris et le Laboratoire d'Informatique et Télécommunications (LIT) à l'École des Télécommunication de Bretagne.

2.2. Principes généraux

2.2.1. L'outil GABI générateur d'une interface

L'application est développée en JAVA afin de profiter des facilités des langages objets, des nombreux composants graphiques et de gestion des données fournis en standard. Il présente l'avantage d'être indépendant de la machine.

Un programme consiste à interpréter et structurer des données, calculer et générer des résultats. Un des principes du projet est de séparer la partie du code assurant la lecture et l'interprétation des données de la partie calculatoire du modèle (figure 1).

Le projet est basé sur l'utilisation de parsers :

- outils classiques d'analyse de données en informatique (les compilateurs de langage sont par exemple écrits à l'aide de parsers) ;
- qui permettent une représentation conceptuelle simple des données (sous forme d'un arbre) ;
- qui font appel à des fichiers de définition, indépendants de la partie calculatoire. La grammaire est écrite dans un langage spécifique. Les fichiers de définition de la grammaire sont ensuite compilés pour générer du code en C qui est intégré au code général.

La figure 2 représente une vision idéalisée du couplage entre un programme de calcul et un générateur d'interface.

Un des aspects essentiel pour comprendre le fonctionnement de l'outil GABI est la représentation des données sous la forme d'une arborescence. Chaque information est décrite à partir d'une suite

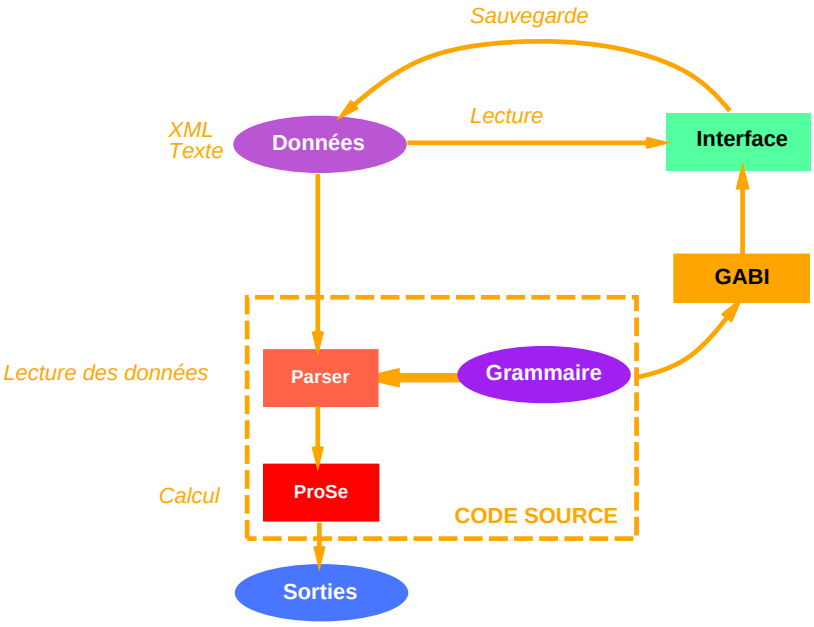

Figure 2: *Représentation schématique du fonctionnement d'un générateur d'interface.*

d'actions (suite de choix) aboutissant à une définition finale. Ainsi la lecture d'un fichier peut se conceptualiser sous la forme d'une action « début » se déclinant elle-même comme la possibilité de lire soit un « en-tête » de fichier, soit des « paramètres », etc. Une fois que l'utilisateur a choisi une des options (un paramètre par exemple), s'affiche alors l'action suivante qui conduira à afficher le format d'entrée d'un paramètre : $\alpha =$ `0,1`. α et $=$ sont des champs parfaitement connus qui peuvent être affichés *a priori*, la valeur 0,1 restant à remplir par l'utilisateur (champ texte à compléter).

La figure 3 illustre schématiquement cette information sous la forme d'un arbre. Tant qu'un choix est possible on parle d'une action correspondant à un nœud de l'arbre. Quand l'ensemble des choix est épuisé on arrive normalement à la définition d'une information représentée à l'aide des feuilles de l'arbre. Les feuilles de l'arbre sont donc des actions connues de trois sortes :

identifiant ou chaîne de caractère connue (signe $=$ par exemple) ;

une liste de plusieurs variables possibles (α peut être présenté dans une liste de choix où on sélectionne le paramètre dont on veut fixer la valeur) ;

une valeur à définir par l'utilisateur : nom, chiffre, etc.

Tout l'enjeu de l'outil GABI est d'être capable de générer la structure des données (l'arbre). Ceci est facilement réalisé avec l'utilisation des parsers. Il doit ensuite produire les outils graphique générique pour chaque type d'objet (nœud, liste, valeur, identifiant) et mettre à jour l'arbre crée par l'utilisateur.

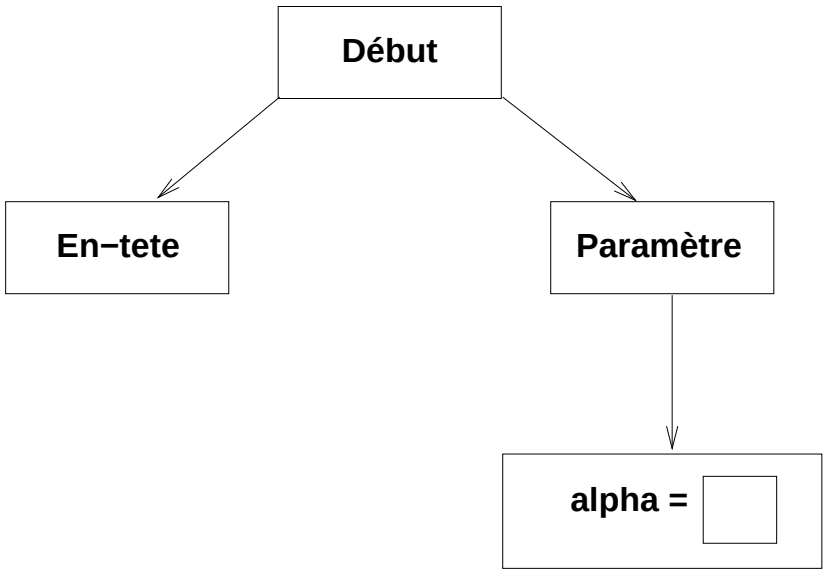

Figure 3: *Représentation schématique d'une suite d'information sous forme d'arbre.*

2.2.2. L'interface générée par GABI

Les fonctionnalités de l'interface générée par l'outil GABI sont les suivantes (figure 2) :

- elle est alimentée par la structure des données, fournie sous forme d'une arborescence ;
- elle doit gérer l'arborescence et permettre des modifications. L'utilisateur vient faire ses choix et remplit au fur et à mesure les champs restés libres (choix dans une liste, valeur à fournir). Ce faisant, l'utilisateur valide un arbre qui correspond aux données qu'il a entrées.
- elle permet la sauvegarde et l'échange en différents formats (texte, arbre, XML, ...).

2.3. Application de GABI au logiciel PROSE

Nous avons crée cette année une interface pour gérer les fonctionnalités de l'outil GABI (figure 4).

Le menu **interface** présente les fonctions de

- génération de l'interface (**générer**). Une boîte de sélection de fichier s'ouvre alors, filtrant les fichiers avec des extensions .l (lexique) puis .y (grammaire yacc) afin de les analyser (figure 5). Une fois les deux fichiers sélectionnés, l'analyse est lancée. Quand elle est finie, une fenêtre anonce la fin du traitement et demande si on veut sauvegarder la nouvelle strucuture (figure 6). Si on répond « oui » une nouvelle boîte de sélection de fichier permet de choisir un fichier pour sauvegarder l'arbre.
- lancement de l'interface (**lancer**). Une fois l'arbre des données crée, on peut lancer l'interface proprement dite (figure 7). Une fois qu'une structure a été sauvegardée par un traitement précédent, l'interface peut être relancée à loisir. On peut donc directement commencer par là, sans traitement préalable. L'interface se présente de la manière suivante (figure 7)

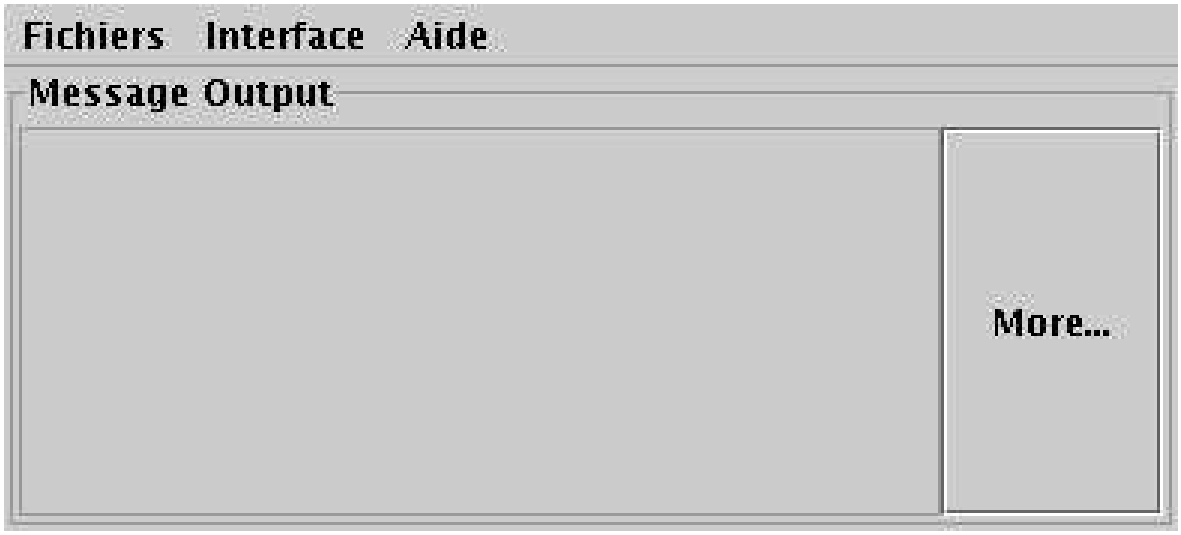

Figure 4: *Fenêtre de gestion de* GABI.

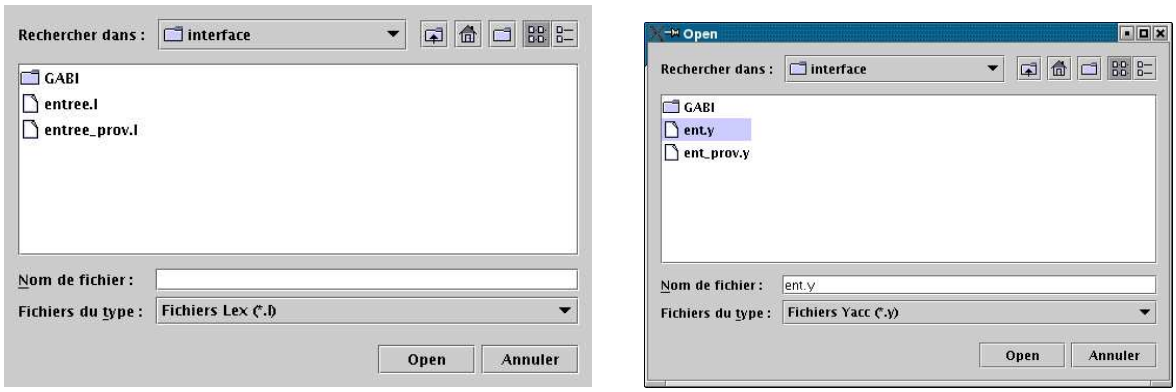

Figure 5: *Boîte des gestion de fichier. Ici, filtre pour sélectionner les fichiers* **.l** *et* **.y**.

Figure 6: *Fenêtre apparaissant à l'issu de l'analyse des parsers.*

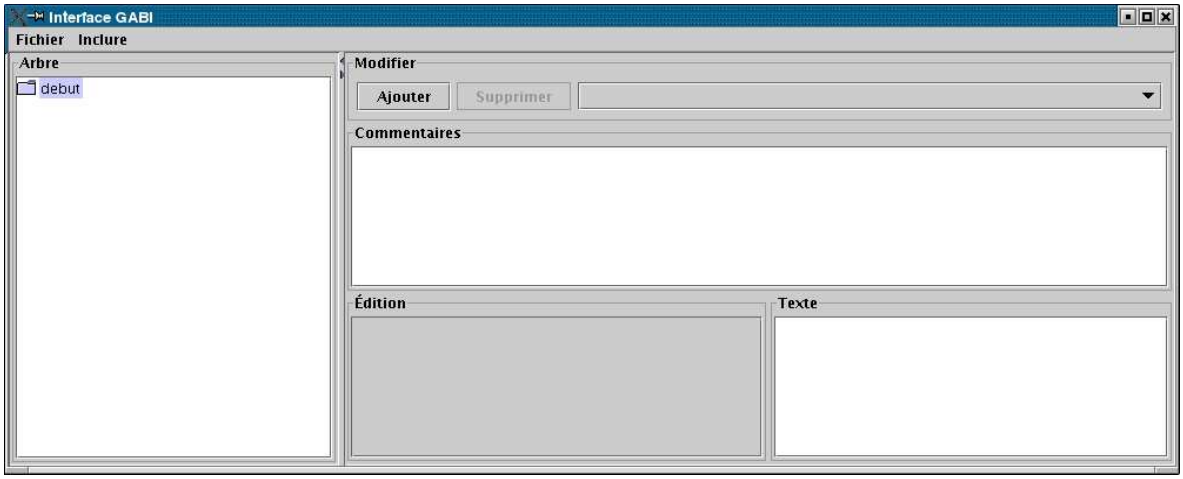

Figure 7: *Fenêtre générale de l'interface généré par* GABI.

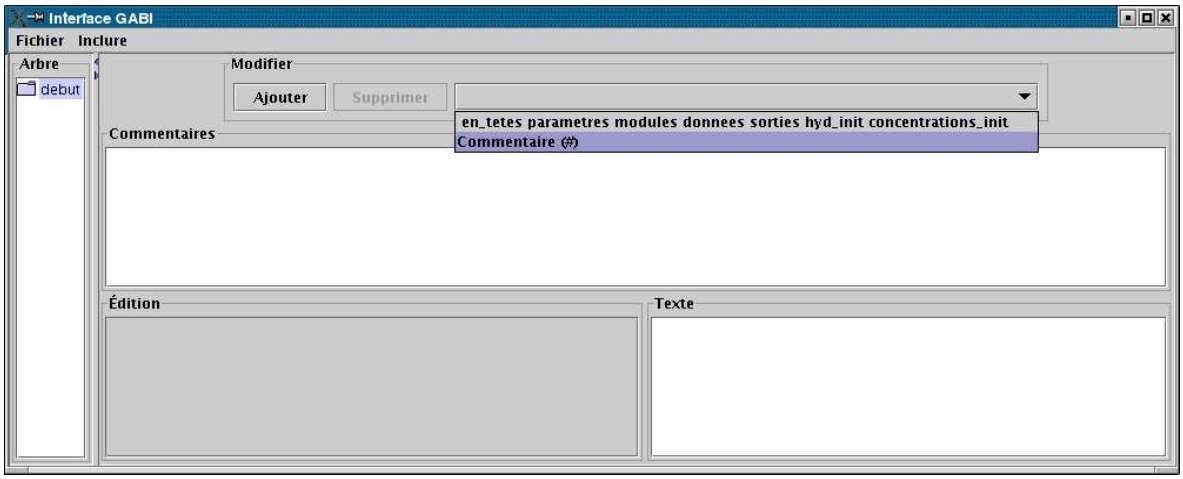

Figure 8: *Affichage d'un menu de sélection d'actions.*

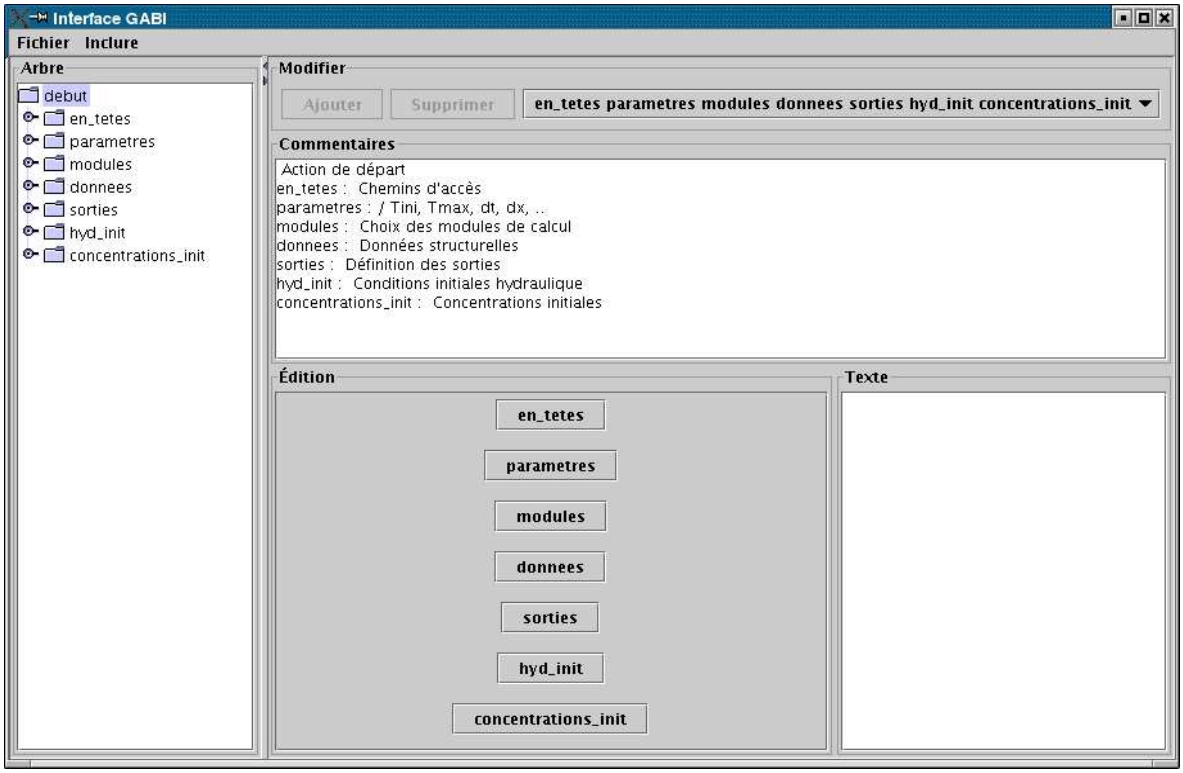

Figure 9: Après ajout d'une action sélectionnée : actualisation de l'arbre, affichage des commentaires pour les éléments de l'action, affichage des composants.

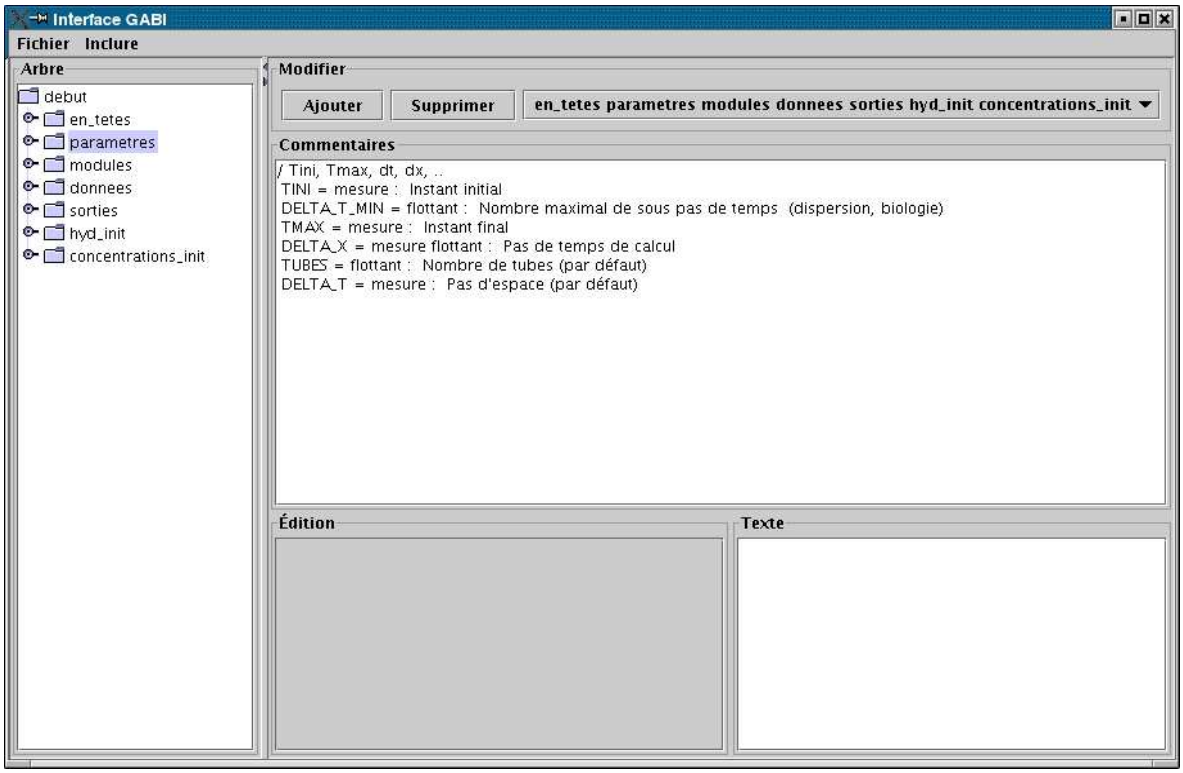

Figure 10: *Changement de sélection dans l'arbre. Affichage des commentaires pour les élemnets de* `paramètres`.

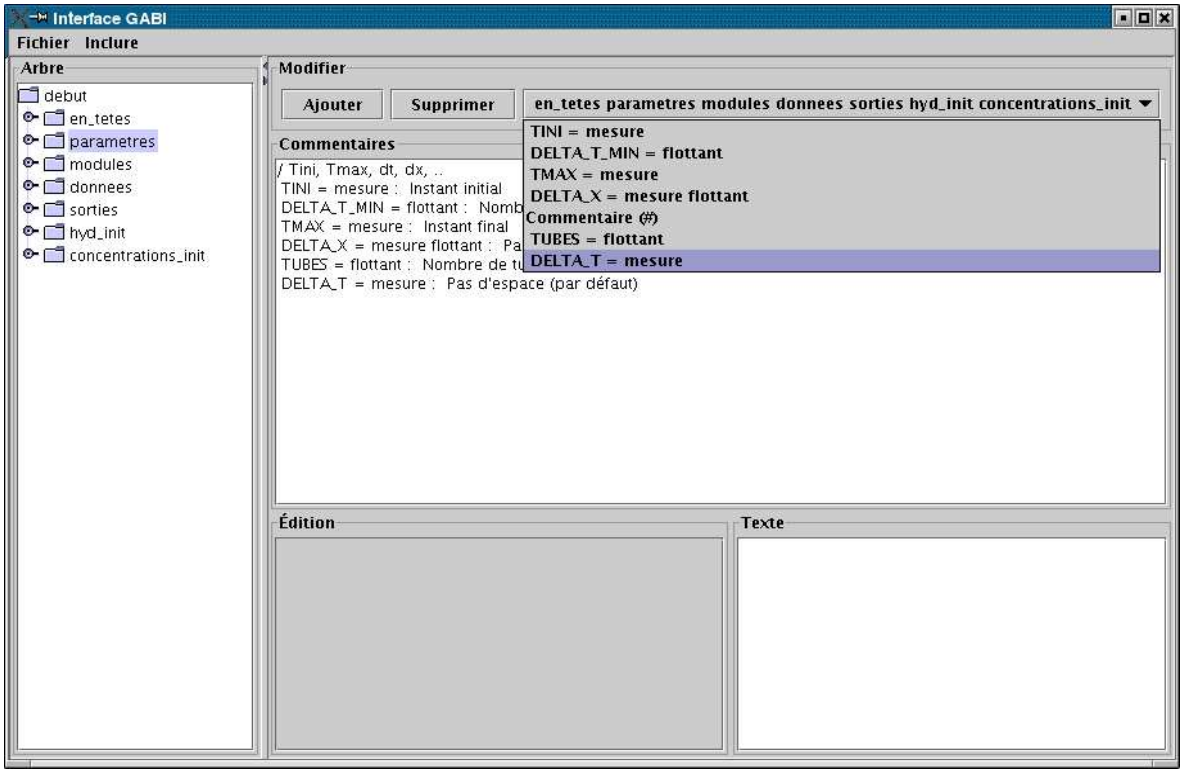

Figure 11: *Sélection d'une action de* `paramètres`.

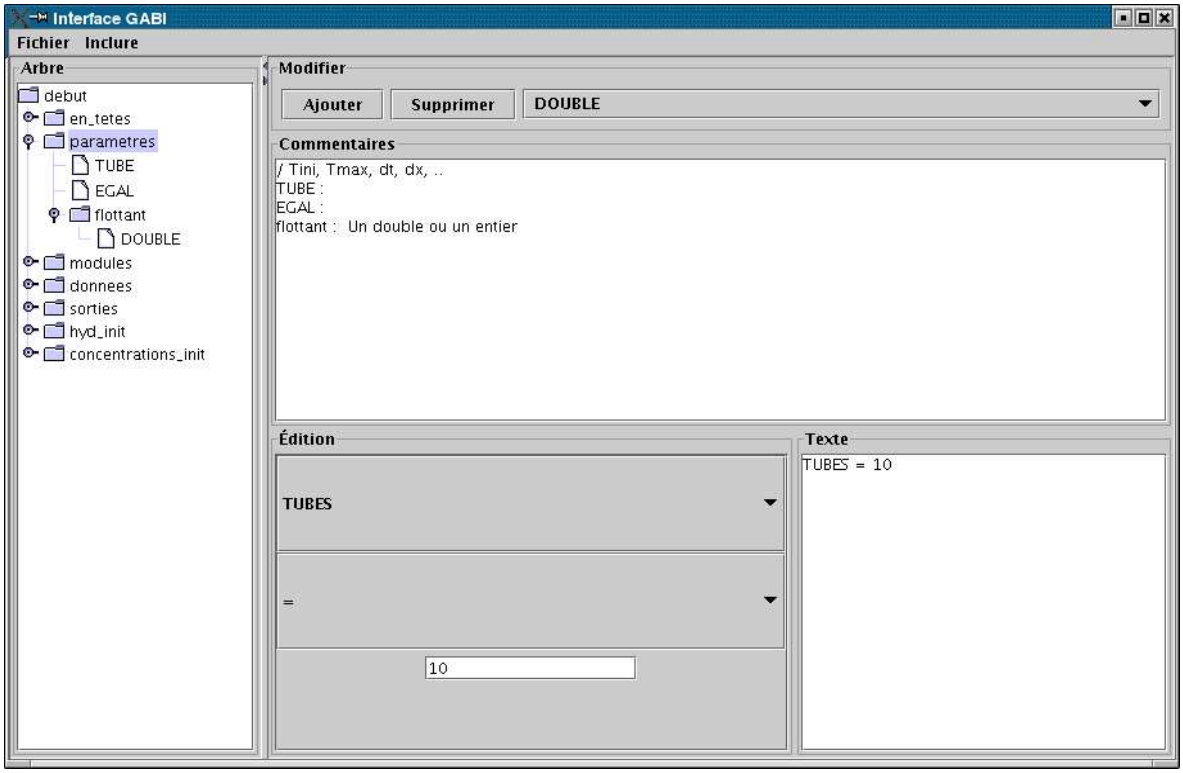

Figure 12: *Entrée de l'information pour un paramètre : affichage automatique des champ identifiants et remplissage du champ valeur par l'utilisateur. Affichage du texte dans la aprtie en bas à droite.*

– la partie gauche présente l'arbre, initialisé à l'action première (figure 7).

– la partie en haut à droite fournit le moyen de créer l'arbre sous la forme d'une liste de choix des actions. Ainsi l'action début se décline comme une suite d'actions *en_tetes parametres modules donnees* ...) (figure 8). La fonction **ajouter** permet d'ajouter une action, **supprimer** permet d'en supprimer. Le bouton **ajouter** ayant été cliqué, la structure de l'arbre est mise à jour dans la partie gauche ;

– des commentaires permettant de décrire chaque type d'information gérée par les différentes actions sont présentés dans la partie centrale à droite. Ces commentaires sont en fait des commentaires du fichier d'origine (.y) qui ont été récupérés automatiquement.

– Dans la partie en bas à droite s'affichent les éléments graphiques sélectionnés (figure 9). Si l'on sélectionne l'action paramètres, la partie commentaires est actualisée permettant de connaître les différents paramètres que l'on peut rentrer par ce biais (figure 10). La sélection d'un des paramètres se fait dans la boîte supérieure (figure 11). Après sélection du paramètre **tubes** permettant de spécifier le nombre de tubes, on voit s'afficher les champ permettant de décrire ce paramètre (figure 12).

– On voit alors s'afficher dans la partie en bas à droite le texte qui a été crée. Cette fenêtre permet de visualiser le texte qui serait sauvegardé.

On peut voir apparaître dans les listes de choix des actions (figures 8 et 11) un champ **commentaire**. La gestion des commentaires est automatique. Il a été identifié dans les fichiers de la grammaire initiale que les commentaires étaient définis et le tag de commentaire (ici #) a été identifié. Les commentaires on été identifiées comme le fait d'ignorer toute une fin de ligne suivant un tag de commentaire. Les commentaires pouvant être placés partout dans le texte, ils apparaissent à chaque action. Sur la figure 13, un commentaire a été crée.

Nous avons également introduit la fonction **inclure** qui est une autre des fonctionnalités des parsers. En effet les parsers permettent de définir simplement un mot clé qui gèrera l'inclusion de fichier. Dans notre exemple, l'analyse de la grammaire initiale a permis d'identifier que cette fonction était bien définie et de trouver le mot clé qui l'initiait (en l'occurance **inclure**). Cette fonctionnalité apparaît dans un menu supérieur.

Il suffit alors de cliquer sur un nœud pour introduire l'inclusion d'un fichier à cet endroit. Nous avons cliqué sur le nœud **en_tetes** puis sélectionné le menu **inclusion d'un fichier** dans le menu principal **inclure**. Une boîte de choix de fichier apparaît alors dans laquelle nous avons sélectionné un fichier *x*. Sur la figure 14 apparaît le résultat de l'opération. Le nœud **en_tetes** est marqué avec le tag d'inclusion. Les champs définissant l'inclusion ont été crées. Si l'on sélectionne à nouveau la racine de l'arbre, le texte écrit en bas à droite fait appraître l'inclusion du fichier **x** qui contiendra les données qui seront contenues dans l'en-tête. Si l'on clique sur le nœud **en_tetes** le contenu du fichier *x* s'affiche. Il est vide a priori (figure 15). Sur la figure 16 s'affiche son contenu après qu'on est spécifié les chemins d'accès aux données (un ou plusieurs noms de répertoire).

Dans le menu **inclure**, la fonction **retour à la racine** permet de revenir à la structure de l'arbre vue depuis la racine générale (figure 17).

Il ne reste alors plus qu'à sélectionner le menu enrgistrer dans **fichier** pour sauvegarder le jeu de données nouvellement crées au format texte.

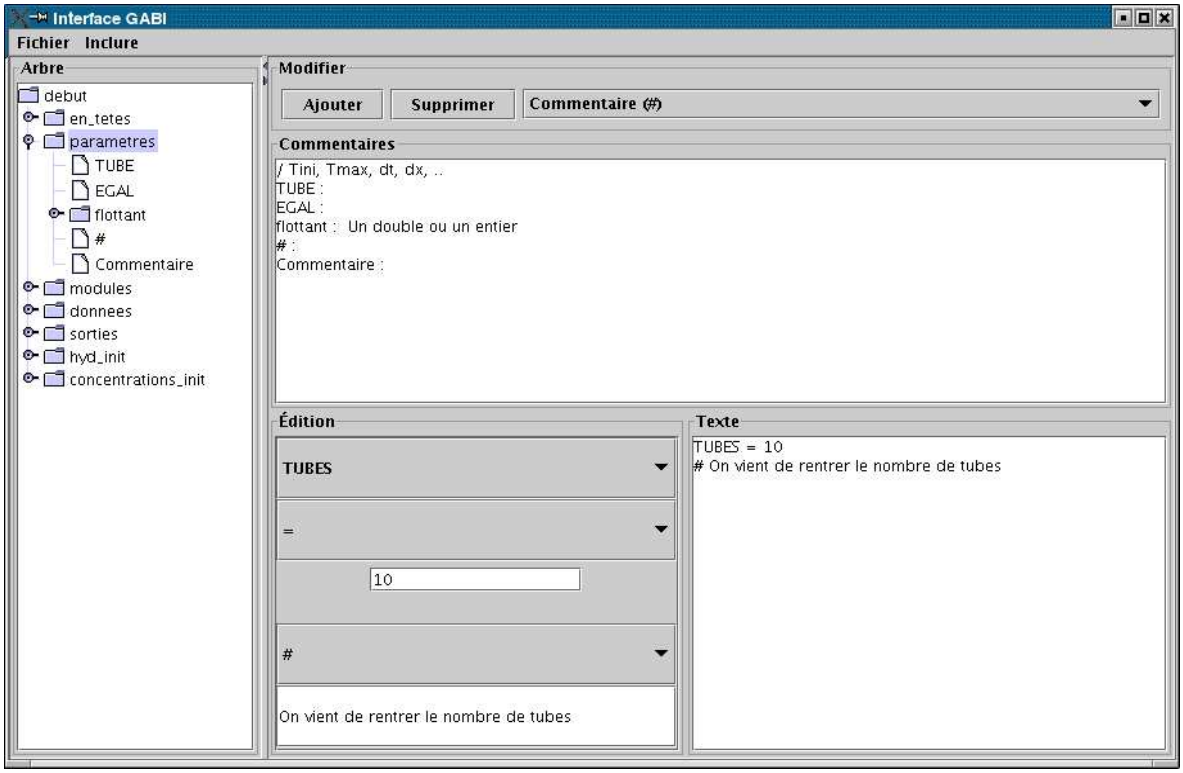

Figure 13: *écriture d'un commentaire.*

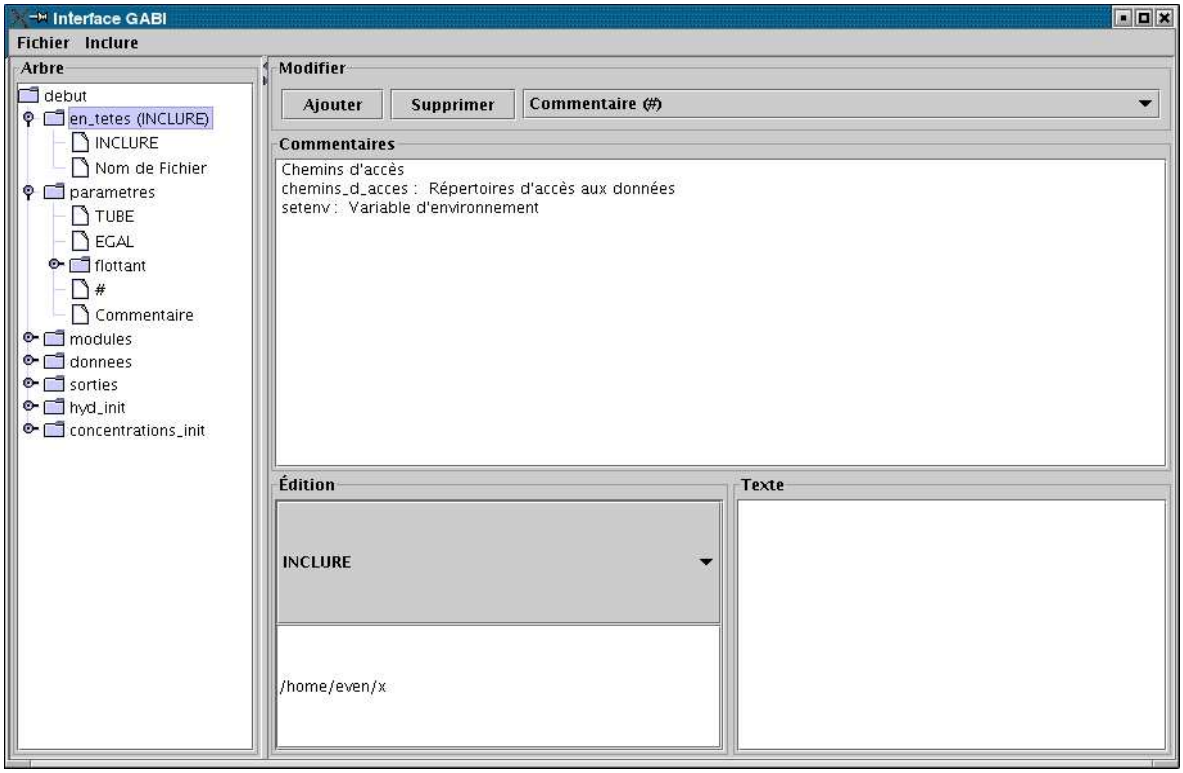

Figure 14: *Utilisation de la fonction* **inclure** *pour créer un fichier* x *où seront écrites les données de l'en-tête du fichier.*

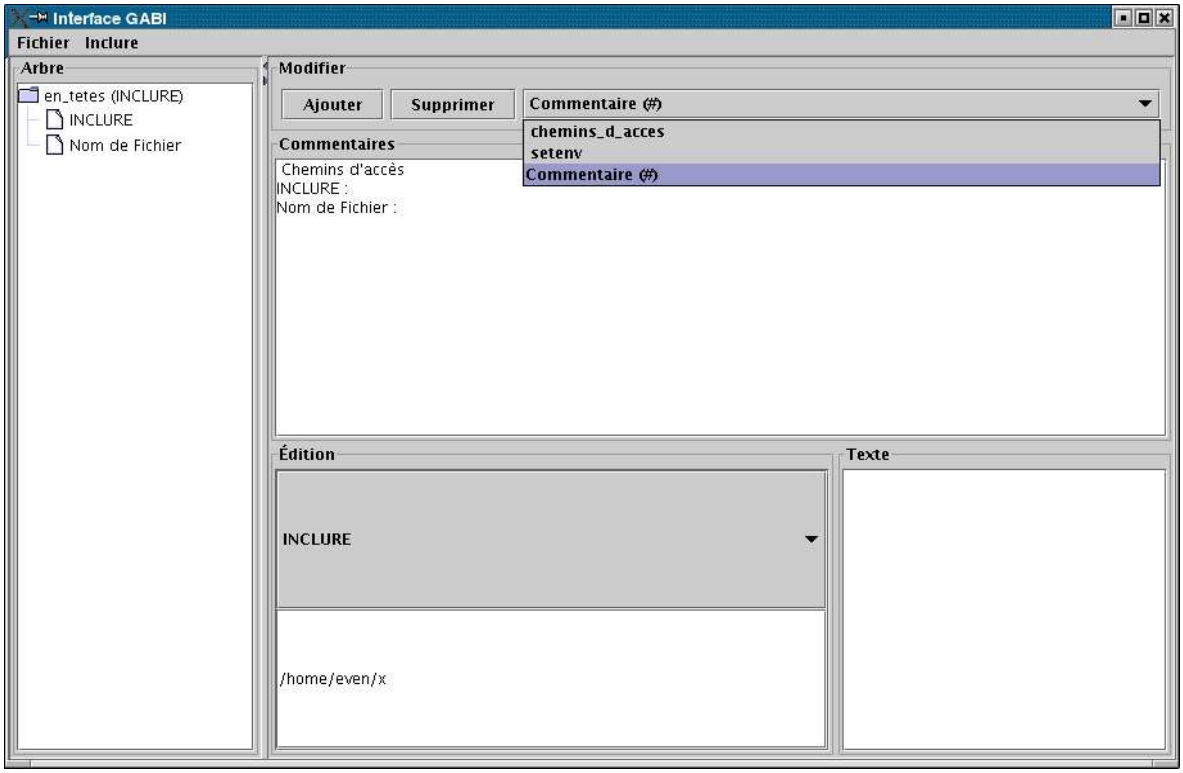

Figure 15: *Visualisation de l'arbre après une inclusion : visualisation d'un sous-arbre dont la racine est le nœud où se situe une inclusion, si c'est ce nœud qui est sélectionné.*

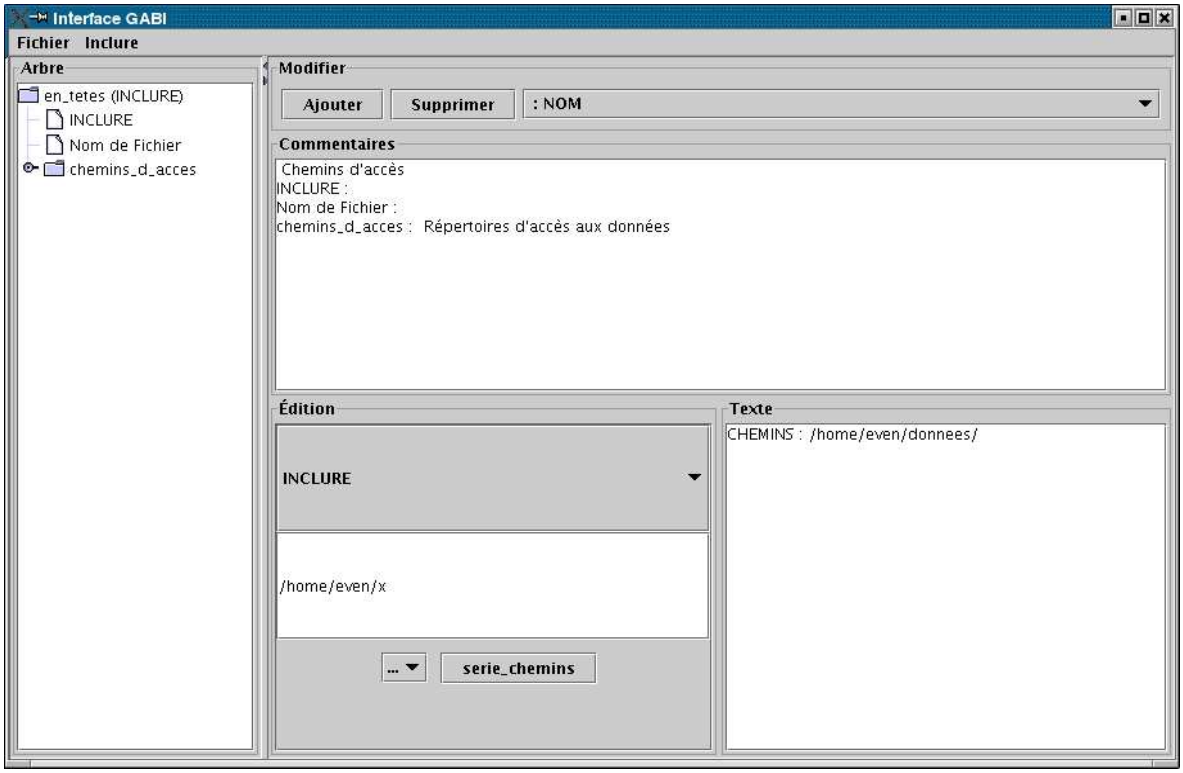

Figure 16: *Entrée d'une information dans le fichier* x.

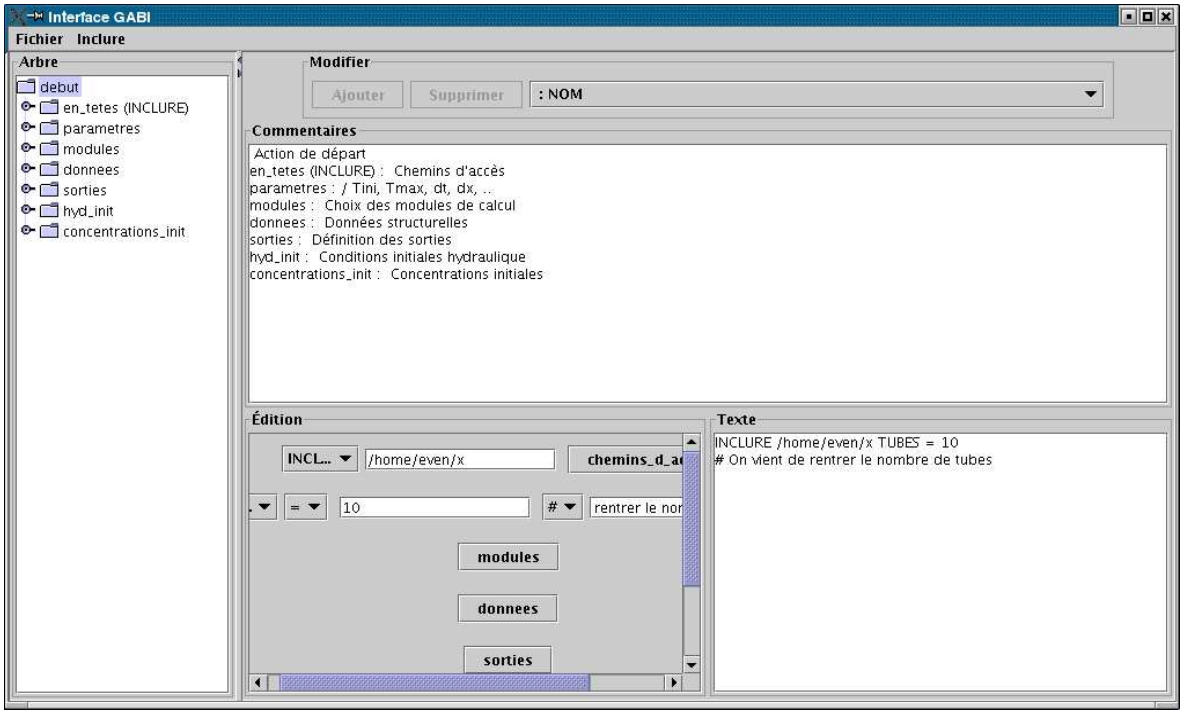

Figure 17: *Retour à la racine générale de l'arbre. Le nœud en tête n'apparaît qu'à travers la ligne* `inclure /home/even/x` *et le contenu du fichier n'est plus affiché.*

3. Le portage sous WINDOWS

Nous avons cette année commencé à explorer le portage du logiciel PROSE sous WINDOWS. En effet ce logiciel est développé sous un système UNIX et a jusqu'à présent été utilisé exclusivement sous des environnements de type d'environnement UNIX (AIX, LINUX, ...).

Une première option envisageable et qui a été testée, est l'installation d'un émulateur d'environnement UNIX sous WINDOWS à savoir CYGWIN. Les avantages de cette installation sont que :

- l'installation du logiciel est rapide et facile à mettre en œuvre ;
- l'utilisateur peut alors bénéficier des outils annexes utilisés avec le logiciel (logiciels graphiques essentiellement) et distribués en free ware.

Le désavantage de cet environnement pour des utilisateurs essentiellement habitués à travailler sous WINDOWS, est de proposer un environnement de travail type UNIX, relativement différent. L'expérience montre que c'est alors un handicap. Le point clé étant que dans sa version actuelle, l'utilisation du logiciel passe essentiellement pas l'écriture de lignes de commandes et l'édition de fichiers textes pour éditer et modifier les fichiers d'entrée.

On en revient à la question centrale, celle d'une interface générale permettant de guider l'utilisateur dans l'utilisation du logiciel et qui permettrait de gérer les entrées, lancer le calcul et visualiser les résultats à partir d'une fenêtre unique qui pourrait être iconifiée.

Nous proposons donc de progresser davantage dans le sens de la création d'une interface intégrée du logiciel PROSE. La compilation sous un environnement WINDOWS serait alors envisagée pour permettre une utilisation plus « habituelle » pour une majorité d'utilisateurs.

4. Conclusion

De nombreuses amélioration ont été apportées à l'outil GABI cette année :

- création d'une interface générale pour gérer ses fonctionnalités (analyser la structure des données à partir de la grammaire, générer et lancer l'interface) ;
- des fonctions d'analyse de l'arbre ont été introduites afin de la simplifier ;
- concernant l'interface lui-même, les améliorations ont consisté dans
  - l'ajout de la fonction de suppression d'une action;
  - l'ajout de commentaires pour décrire les actions ;
  - l'ajout d'une fenêtre faisant apparaître la version texte des données créees.
  - la gestion des commentaires avec identification du tag propre à la grammaire ;
  - la gestion de la fonction inclure.

En plus de toutes ces fonctionnalités un travail important de débogage a été fait, rendant aujourd'hui l'outil quasiment utilisable, avec les fonctions essentielles qui lui étaient *a priori* demandées. Cependant des améliorations importantes restent encore à apporter :

- la gestion de l'affichage d'un nombre important de données. La fonction **inclure**, permettant de sectionner l'information, fournit d'ores et déjà une aide dans ce sens.
- la fonction commentaire devra être gérée au niveau des menus, comme la fonction inclure, afin d'alléger les affichages.
- un système de marquage dans les fichiers de grammaire initiaux est envisagé pour simplifier *a priori* la structure des données selon le désir du concepteur. Ainsi s'il est décidé qu'un certains nombre de données complexes (comme la bathymétrie d'une rivière par exemple) sont fixées *a priori*, une fonction dévolue au logiciel peut-être par exemple de ne permettre de modifier que des apports. Seules ces fonctions doivent apparaître comme modifiables.
- les fonctions de lecture de données au format texte doivent être crées. Les données sont actuellement sauvegardées dans un double format : texte et arbre sérialisé. Seul ce dernier format est actuellemnt lisible en entrée. Des fichiers texte crées par ailleurs ne sont donc pas actuellement lisible par l'interface. La base de cette fonctionnalité existe déjà, puisque des parsers JAVA compatibles avec la grammaire initiale sont générés lors de l'analyse des fichiers. Ce sont ces parsers JAVA, couplés à l'interface, qui doivent permettre de relire les fichiers texte en entrée.
- Ces parsers peuvent également être utilisés pour vérifier la conformité du formatage des fichiers.

L'interface de PROSE crée par l'outil GABI sera intégré à terme dans l'interface général du logiciel pour gérer ses entrées. Nous pensons à l'avenir progresser dans le développement d'une interface intégrée du logiciel PROSE, étape que nous considérons comme nécessaire avant d'envisager son installation sous environnement WINDOWS.