

# Atelier B 4

## Manuel de l'Utilisateur



Document établi par ClearSy.

Cette documentation est placée sous License CREATIVE COMMONS - PATERNITÉ (CC-BY).



Tous les noms des produits cités sont des marques déposées par leurs auteurs respectifs.

ClearSy System Engineering  
Parc de la Duranne - 320 av. Archimède  
Les Pléïades III Bat A  
13857 AIX EN PROVENCE CEDEX 3  
FRANCE

Tel : 33 (0)4 42 37 12 70  
Fax : 33 (0)4 42 37 12 71  
email : [contact@clearsy.com](mailto:contact@clearsy.com)

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	À propos . . . . .	6
1.2	Objets manipulés par l'Atelier B . . . . .	6
1.3	Modes d'utilisation de l'Atelier B . . . . .	7
1.4	Organisation du document . . . . .	8
1.5	Conventions typographiques . . . . .	9
<b>2</b>	<b>Installation</b>	<b>10</b>
2.1	Préparation de l'installation . . . . .	10
2.1.1	Introduction . . . . .	10
2.1.2	Ressources nécessaires pour l'installation . . . . .	11
2.1.3	Espace mémoire . . . . .	11
2.1.4	Espace disque . . . . .	11
2.1.5	Outils connexes . . . . .	11
2.1.6	Utilisation en réseau hétérogène . . . . .	12
2.1.7	Création du gérant de l'Atelier B . . . . .	12
2.1.8	Choix du répertoire d'installation . . . . .	13
2.1.9	Si vous avez une version antérieure . . . . .	13
2.1.10	Installation sur une partition en lecture seule . . . . .	14
2.2	Installation des fichiers . . . . .	15
2.2.1	Windows . . . . .	15
2.2.2	Mac OS X . . . . .	17
2.2.3	GNU/Linux et Solaris . . . . .	18
2.3	Versions antérieures de l'Atelier B . . . . .	20
<b>3</b>	<b>Présentation générale</b>	<b>21</b>
3.1	Interfaces . . . . .	21

3.2	IHM principale . . . . .	22
3.3	Éditeur intégré . . . . .	23
3.4	Interface en ligne de commande . . . . .	25
<b>4</b>	<b>Pour démarrer</b>	<b>28</b>
4.1	Gestion des projets . . . . .	30
4.1.1	Liste des projets . . . . .	31
4.1.2	Création d'un projet . . . . .	32
4.1.3	Suppression d'un projet . . . . .	34
4.1.4	Ouverture et fermeture d'un projet . . . . .	36
4.2	Projets : Gestion avancée . . . . .	39
4.2.1	Gestion des utilisateurs . . . . .	39
4.2.2	Gestion des bibliothèques . . . . .	40
4.2.3	Gestion des répertoires de fichiers de définitions . . . . .	41
4.2.4	Archivage . . . . .	42
4.2.5	Restauration . . . . .	44
4.2.6	Lire des informations sur un projet . . . . .	46
4.3	Gestion des composants . . . . .	48
4.3.1	Ajout de composants . . . . .	48
4.3.2	Exclusion de composants . . . . .	51
4.3.3	Affichage des composants . . . . .	51
4.3.4	Propriétés des composants . . . . .	53
4.3.5	Édition d'un composant . . . . .	54
4.3.6	Désarchiver un composant . . . . .	56
<b>5</b>	<b>Appliquer la méthode B</b>	<b>58</b>
5.1	Présentation . . . . .	58
5.2	Gestion des espaces de travail . . . . .	59

5.2.1	Création d'un espace de travail . . . . .	59
5.2.2	Modification des espaces de travail . . . . .	61
5.2.3	Suppression d'un espace de travail . . . . .	63
5.2.4	Ouverture et fermeture d'un espace de travail . . . . .	63
5.3	Analyse syntaxique et contrôle de type . . . . .	64
5.4	Raffinement automatique . . . . .	66
5.5	Génération des obligations de preuve . . . . .	67
5.6	Affichage des obligations de preuve . . . . .	71
5.7	Démonstration automatique . . . . .	72
5.8	Démonstration interactive . . . . .	75
5.9	Annulation des démonstrations . . . . .	76
5.10	Contrôle du langage d'implantation . . . . .	76
5.11	Contrôle global d'un projet . . . . .	79
5.12	Traduction . . . . .	80
5.13	Appliquer un outil à l'ensemble des composants d'un projet . . . . .	81
5.14	Mise à jour d'un projet . . . . .	82
5.15	Interruption des outils . . . . .	83
5.16	Gestion des dépendances . . . . .	83
<b>6</b>	<b>Analyse des développements B</b>	<b>85</b>
6.1	Présentation . . . . .	85
6.2	État d'un projet . . . . .	86
6.3	État d'un composant . . . . .	87
6.4	Graphes de dépendances . . . . .	89
6.5	Graphes d'appel d'opération . . . . .	92
6.6	Références croisées . . . . .	93
6.7	Extraction de métriques . . . . .	95

<b>7</b>	<b>Documentation des projets B</b>	<b>100</b>
7.1	Présentation . . . . .	100
7.2	Afficher un source B . . . . .	100
<b>8</b>	<b>Modification des paramètres de l'Atelier B</b>	<b>103</b>
8.1	Introduction . . . . .	103
8.2	Présentation du système de paramétrage . . . . .	103
8.3	Création d'un fichier de ressources . . . . .	104
8.4	Affichage des valeurs des ressources et de la version de l'Atelier B . . . . .	105
8.5	Modification des outils connexes . . . . .	106
8.6	Modification de la place mémoire du Logic Solver . . . . .	107
8.7	Modification de la place mémoire du Parser K . . . . .	108
8.8	Paramétrage du script d'impression . . . . .	109
	<b>Annexes</b>	<b>111</b>
	Limitations des outils de documentation projet . . . . .	111
	Fichiers créés par l'Atelier B . . . . .	112

# 1 Introduction

## 1.1 À propos

Ce document a pour but de présenter les différentes fonctionnalités offertes par l'Atelier B.

L'Atelier B est un ensemble d'outils logiciels permettant le développement d'applications suivant la méthode B.

L'Atelier B assiste le concepteur dans la formalisation de son application :

- En exécutant automatiquement sur les spécifications et leurs raffinements, un certain nombre d'actions dictées par la méthode,
- En proposant des services “annexes” à la méthode, mais néanmoins indispensables à des développements industriels, comme la gestion, l'analyse et la documentation d'un projet.

## 1.2 Objets manipulés par l'Atelier B

Nous introduisons dans ce paragraphe les principaux objets manipulés par les fonctions de l'Atelier B.

Composant : Fichier contenant un source écrit en langage B. Il constitue la base d'un développement suivant la méthode B. Un composant est un terme générique et représente :

- soit une spécification B (machine abstraite),
- soit un raffinement de cette spécification,
- soit son implémentation (dernier niveau de raffinement).

La saisie des composants est réalisée grâce à l'éditeur de texte de la station de développement.

Projet : Ensemble de fichiers (composants, fichiers annexes sources C, C++, HIA, ou ADA, makefiles) utilisés ou produits durant le développement d'une application suivant la méthode B, complété des informations nécessaires à la gestion de ces fichiers sous l'Atelier (voir BDP).

Utilisateur : Il est nécessaire de définir une liste d'utilisateurs autorisés à accéder et modifier le projet. Notez que tous les utilisateurs ont les mêmes pouvoirs, et qu'il est nécessaire qu'au moins un utilisateur soit renseigné.

Base de Données Projet (BDP) : Toutes les informations internes nécessaires à la bonne exécution des outils de l'Atelier B sont stockés dans un répertoire nommé BDP. Ce

répertoire contient aussi des fichiers produits par les outils de documentation de l'Atelier B.

### 1.3 Modes d'utilisation de l'Atelier B

Dans ce document, nous appelons **Outils B** les outils liés à l'application de la méthode B, ainsi qu'à l'analyse, la mise au point et la documentation de logiciels écrits en B. L'environnement B présenté dans ce manuel, propose deux modes d'utilisation des outils B :

- un **mode interactif**, utilisant une interface graphique à base de fenêtres et de boutons de commande ; dans la suite du document nous appellerons ce mode *interface utilisateur graphique*,
- un **mode programmé**, reposant sur un langage appelé **langage de commande** ; dans la suite de ce document, nous appellerons cette interface *interface utilisateur mode commande*.



## 1.4 Organisation du document

Ce document est orienté autour de 3 axes, permettant d'initier progressivement le lecteur à l'utilisation de l'Atelier B et ses différentes interfaces Homme / Machine, dans le cadre d'un développement B.

Il a été écrit dans le but d'initier l'utilisateur novice, mais aussi afin que les utilisateurs des anciennes versions puissent retrouver facilement leurs repères dans cette version.

- La première partie s'attarde sur l'installation de l'Atelier B, les prérequis de configuration nécessaire selon les différents systèmes d'exploitation supportés, et sur la présentation générale des deux interfaces principales.
- La deuxième partie se veut être une sorte de guide de démarrage, présentant les différentes fonctionnalités disponibles au travers des interfaces de l'Atelier B.
- La troisième partie aborde l'utilisation pratique de l'Atelier B, dans le cadre d'un développement B.

## 1.5 Conventions typographiques

Chaque fonctionnalité qu'offre l'Atelier B est en général présentée de la façon suivante :

- Un paragraphe *Description* présente les caractéristiques de la commande,
- Un paragraphe *Interface utilisateur graphique* qui présente la procédure à suivre pour utiliser la commande dans l'interface graphique de l'Atelier B,
- Un paragraphe *Interface utilisateur mode commande* qui présente l'utilisation de la commande via l'invite du mode Commande,
- Un paragraphe *paramètres utilisables* éventuel, qui présente les ressources permettant de paramétrer le fonctionnement de la commande.

Les paramètres sont présentés de la façon suivante :

Exemple	Commentaire
ATB*POG*Generate_Obvious_PO	nom de la ressource
Configuré à l'installation de l'Atelier B	valeur par défaut
Générer ou non les obligations de preuve triviales.	Description de la ressource

Dans les descriptions de l'interface utilisateur les noms des boutons sont toujours écrits en italique. Par exemple, le bouton *"Help"*.

## 2 Installation

Cette partie décrit les procédures à suivre pour :

- Installer l’Atelier B,
- Configurer et paramétrer l’Atelier B.

Il est conseillé de lire l’ensemble du chapitre avant de procéder à l’installation de l’Atelier B.

L’installation peut nécessiter les compétences de votre administrateur système. En effet, certaines opérations peuvent potentiellement n’être réalisées qu’en étant “super-utilisateur”.

La première partie donne toutes les informations vous permettant de choisir la machine et le disque sur lesquels vous allez installer l’Atelier B. Il contient également des explications en vue d’une utilisation en réseau de l’Atelier B.

La seconde partie explique en détail les procédures à suivre pour lire les fichiers de l’Atelier B du support d’installation.

Après avoir réalisé les opérations précédentes, vous devez potentiellement configurer l’Atelier B et les comptes des utilisateurs. Ces opérations sont décrites dans le paragraphe 2.1.7.

La partie 8 contient des informations qui vous permettent de paramétrer l’Atelier B.

La dernière partie de cette section explique les procédures à suivre pour récupérer facilement des projets B développés avec une version antérieure de l’Atelier B.

### 2.1 Préparation de l’installation

#### 2.1.1 Introduction

Cette partie s’intéresse à :

- Choisir la machine d’installation,
- Utiliser l’Atelier B en réseau,
- Créer l’utilisateur gérant l’Atelier B,
- Choisir le disque où seront installés les fichiers de l’Atelier B.

Notez que toutes ces étapes ne sont pas nécessairement utiles selon les systèmes sur lesquels vous envisagez d’utiliser l’Atelier B. En effet, sous système Windows, il est inutile de tenir compte des remarques sur la configuration d’un utilisateur spécifique.

### 2.1.2 Ressources nécessaires pour l'installation

En terme de systèmes d'exploitation, l'Atelier B a été développé en particulier pour les systèmes suivants :

- Linux (avec glibc version 2 ou supérieure),
- Solaris 2.6 (SunOS 5.6) ou compatibles,
- Microsoft Windows (2000 ou supérieur),
- MacOS (versions 10.4 ou supérieure).

### 2.1.3 Espace mémoire

Il est conseillé de disposer d'au moins 512 Mo de mémoire RAM. Toutefois, notez que l'espace mémoire réellement nécessaire est fortement dépendant des autres applications fonctionnant sur votre station et de la taille des développements effectués avec l'Atelier B.

### 2.1.4 Espace disque

Pour installer l'Atelier B, vous avez besoin d'environ 100 Mo d'espace disque.

L'espace disque occupé par un projet développé avec l'Atelier B est dépendant de la taille des fichiers sources B. L'espace disque occupé par les fichiers générés automatiquement par l'Atelier B est égal à environ 25 fois la taille de l'espace disque de tous les fichiers sources B.

### 2.1.5 Outils connexes

L'Atelier B produit également des fichiers exploitables par les outils suivants :

Outil	Version	Nature	Fournisseur
L <sup>A</sup> T <sub>E</sub> X	2E	Traitement de texte	Domaine public
PDFLaTeX	3	Traitement de texte	Domaine public
Word	7 ou supérieure	Traitement de texte	Microsoft

Par ailleurs, afin de profiter pleinement de certaines fonctionnalités de l'Atelier B, il peut être intéressant d'installer des outils annexes, librement disponibles sur Internet pour les différentes plateformes ; parmi eux, citons :

- Dot, disponibles dans les outils GraphViz <sup>1</sup>

---

<sup>1</sup>Visitez <http://www.graphviz.org> pour de plus amples informations

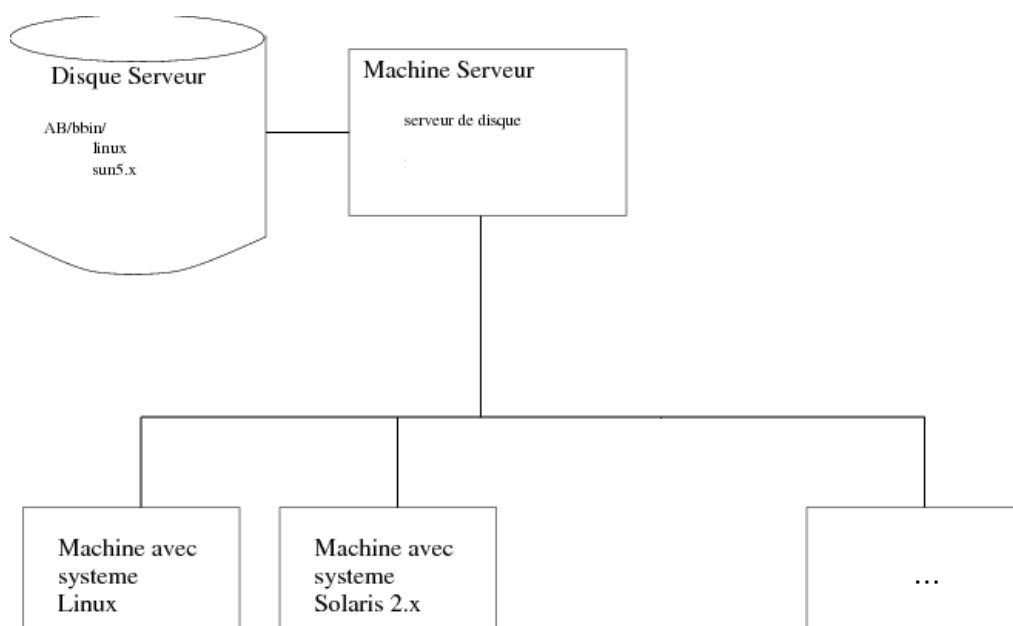
- Un client SSH quelconque (OpenSSH, ou la distribution de Putty pour les utilisateurs sous Windows<sup>2</sup>.)

Vous pouvez utiliser l'Atelier B sans ces outils, toutefois les fonctionnalités reposant sur ces outils externes ne seront alors pas disponibles.

### 2.1.6 Utilisation en réseau hétérogène

Vous pouvez installer l'Atelier B sur un réseau de machines qui utilisent un serveur de fichiers commun. Tous les utilisateurs sur les systèmes supportés par l'Atelier peuvent partager un répertoire d'installation. Ce manuel comporte des instructions pour installer l'Atelier B sur un réseau multi-plateformes.

La figure suivante donne un exemple d'utilisation en réseau.



Dans cet exemple, l'Atelier B est installé sur une machine serveur.

Plusieurs machines, ayant des systèmes différents utilisent l'Atelier B. Le routage en fonction du type de système est fait automatiquement par les scripts de lancement de l'Atelier B.

### 2.1.7 Création du gérant de l'Atelier B

Cette partie ne concerne que les systèmes d'exploitation compatibles UNIX, dans le cas d'une utilisation multi-utilisateur.

<sup>2</sup>Voir <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Le partage de projets par plusieurs utilisateurs entraîne un partage de fichiers UNIX entre ces utilisateurs et donc les problèmes classiques de droit d'accès entre ces utilisateurs.

Afin de ne pas compromettre la sécurité du système, l'Atelier B utilise les droits d'un utilisateur et d'un groupe UNIX nommés **atelierb**.

Avant d'installer l'Atelier B, vous devez donc créer un compte **atelierb** et un groupe du même nom.

Cet utilisateur et ce groupe doivent être définis sur toutes les machines susceptibles d'utiliser l'Atelier B.

En revanche, les fichiers sources des utilisateurs (sources B, fichiers de règles pour la preuve) posséderont les droits donnés par les utilisateurs. Ils peuvent donc être protégés avec les fonctions classiques d'UNIX.

Nous vous recommandons d'utiliser les procédures définies par les constructeurs pour la création de ce compte et de ce groupe :

- Utilisez `admintool` pour les systèmes SUN,
- Utilisez `adduser` ou `linuxconf` pour les systèmes LINUX.

Nous vous recommandons d'utiliser le répertoire choisi au chapitre ci-dessous comme répertoire d'accueil de ce compte.

### 2.1.8 Choix du répertoire d'installation

Si vous installez l'Atelier B pour plusieurs utilisateurs, vous devez choisir un répertoire pour lequel les utilisateurs auront le droit de lecture et d'exécution. Ce répertoire doit aussi être visible de toutes les machines susceptibles d'utiliser l'Atelier B.

Pour installer les fichiers de l'Atelier B vous devez avoir le droit d'écrire dans ce répertoire.

Nous vous conseillons d'installer les fichiers de l'Atelier B dans un répertoire nommé `atelierb` sur la partition disque où vous avez l'habitude d'installer des applications.

Pour installer l'Atelier B, vous avez besoin d'environ 100 Mo. Ces besoins varient toutefois selon le type de système d'exploitation utilisé.

### 2.1.9 Si vous avez une version antérieure

Si vous possédez une version antérieure de l'Atelier B, vous devez installer la nouvelle version dans un répertoire différent ou un sous-répertoire du répertoire précédent. Lisez les instructions de la partie 2.3 de ce document.

Par ailleurs, si vous avez installé une version bêta, il peut être nécessaire afin d'éviter les conflits, de désinstaller préalablement la version installée.

#### 2.1.10 Installation sur une partition en lecture seule

Si pour plus de sécurité, vous choisissez une partition disque, qui est “montée” sur les autres machines en lecture seule vous devrez prendre certaines précautions lors de l'installation de l'Atelier B.

En effet, par défaut l'Atelier B utilise un répertoire où les utilisateurs doivent pouvoir écrire des fichiers. Ces fichiers contiennent des informations sur les projets créés par les utilisateurs.

Ce répertoire nommé **bdb** représente la *base de données Atelier*. Il est par défaut situé dans le répertoire d'installation de l'Atelier B. Si la partition d'installation est en lecture seule vous devrez utiliser une base de données Atelier sur une autre partition disque.

## 2.2 Installation des fichiers

Cette section décrit la procédure d'installation de l'Atelier B et de ses options sur une station de travail ou un serveur.

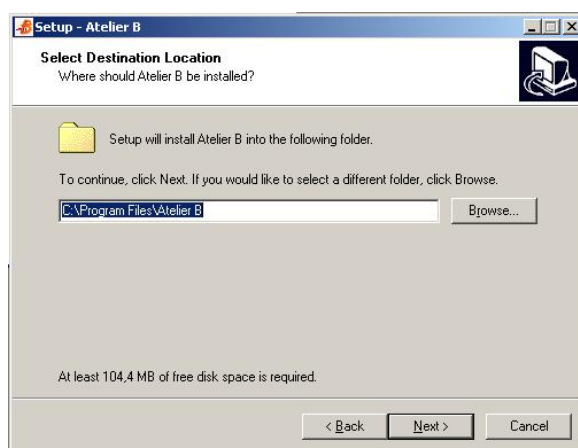
### 2.2.1 Windows

Pour ce système d'exploitation, il existe un assistant permettant d'automatiser l'installation.



Il suffit ainsi de se laisser guider par l'assistant afin de parvenir à installer l'Atelier B.

Il vous sera demandé quelques préférences d'installation, par exemple sur la figure suivante, où le chemin d'accès de l'installation vous est demandé :



Enfin, si l'installation s'est correctement déroulée, vous devriez voir apparaître l'écran suivant :





### **2.2.2 Mac OS X**

Sous ce système d'exploitation, l'Atelier B est aussi fourni sous la forme d'un installeur, possédant l'extension “.dmg”. Il suffit de lancer l'installeur et de se laisser guider par les différentes étapes.

### 2.2.3 GNU/Linux et Solaris

L'installation de l'Atelier B sous GNU/Linux ou Solaris peut nécessiter la création d'un compte utilisateur "atelierb". Cela n'est pas obligatoire, mais comme vu précédemment, dans le cas d'une utilisation multi-utilisateur, cela permet de résoudre un grand nombre de problèmes liés aux droits d'accès sur les fichiers. Veuillez vous reporter à la partie précédente pour de plus amples informations sur la création d'un utilisateur "atelierb" spécifique.

Bien entendu, l'ajout d'un utilisateur peut dépendre de la configuration et de la politique de sécurité en place sur votre machine et/ou votre réseau informatique ; n'hésitez pas à prendre contact avec votre administrateur système en cas de doutes.

L'installation sur ces types de système s'effectue via un script shell. Après avoir récupéré l'archive de l'Atelier B, il suffit de la décompresser et de lancer le script "install\_atelierb".

```
user@host:~$ su atelierb
Mot de passe :
atelierb@host:/home/user$

[...]

atelierb@host:~$ tar xvzf atelierb-4.0-linux.tgz

atelierb@host:~$ cd atelierb-4.0/

atelierb@host:~/atelierb-4.0$ ./install_atelierb
```

Suivez alors les instructions données par le script ; ci-après, les différentes étapes proposées par l'installateur :

-----

#### ATELIERB Installation Procedure

Machine : host

Operating System : Linux 2.6.24-23-generic

Most of the questions that you will be asked have a default answer

enclosed in brackets, for instance:

Do you understand [yes]?

To use the default answer, just type "return".

-----  
Would you like to continue [yes]?

Installation procedure steps :

- 1 - Extract installation files
- 2 - Configure and parameter Atelier B products
- q - Quit installation procedure

Go to step number : [1]?

Do you want to generate obvious proof obligations as a default [no]?

Enter the complete path to your text editor [/usr/local/bin/xemacs]?

Is Latex installed on your system [yes]?

Enter the directory containing Latex binaries [/usr/bin]?

Enter the name of the Latex viewer [xdvi]?

Enter the name of the Latex to PostScript generator [dvips]?

Is there a HTML viewer installed on your system [yes]?

Enter the complete path to your HTML viewer [/usr/bin/firefox]?

Installation of Atelier B succeeded [OK]?

## 2.3 Versions antérieures de l'Atelier B

Cette partie décrit les procédures à suivre pour récupérer les projets développés avec la version précédente de l'Atelier B.

La procédure à suivre est la suivante :

1. Archivage des projets : Avec la version précédente de l'Atelier B, vous devez archiver les projets que vous souhaitez conserver. Pour cela utiliser la fonctionnalité “*Archiver un projet*”, en sélectionnant l'option d'archivage des fichiers sources B et des fichiers de preuve uniquement,
2. Installation de la nouvelle version de l'Atelier,
3. Désarchivage des projets : Avec la nouvelle version de l'Atelier B, désarchivez les projets précédemment archivés en utilisant la fonctionnalité de restauration décrite ci-après à la section 4.2.5.

## 3 Présentation générale

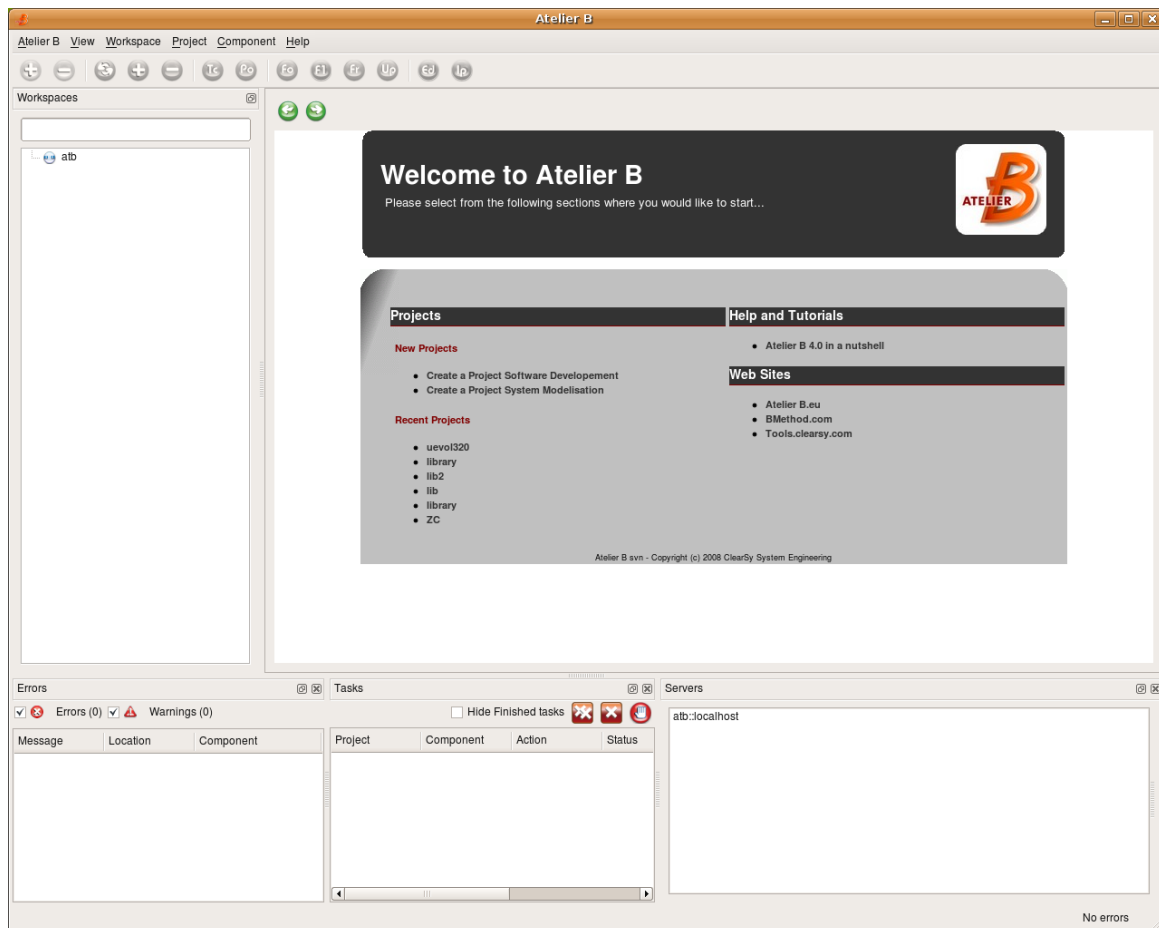
### 3.1 Interfaces

L'interface graphique de l'Atelier B se décompose en deux parties majeures : l'une est dédiée à la gestion des projets B, et l'autre permet une édition aisée via un éditeur spécialement conçu pour assister l'utilisateur dans son développement B.

Nous aborderons par la suite les détails de l'utilisation de ces dernières.

## 3.2 IHM principale

L'interface principale se présente de la façon suivante :



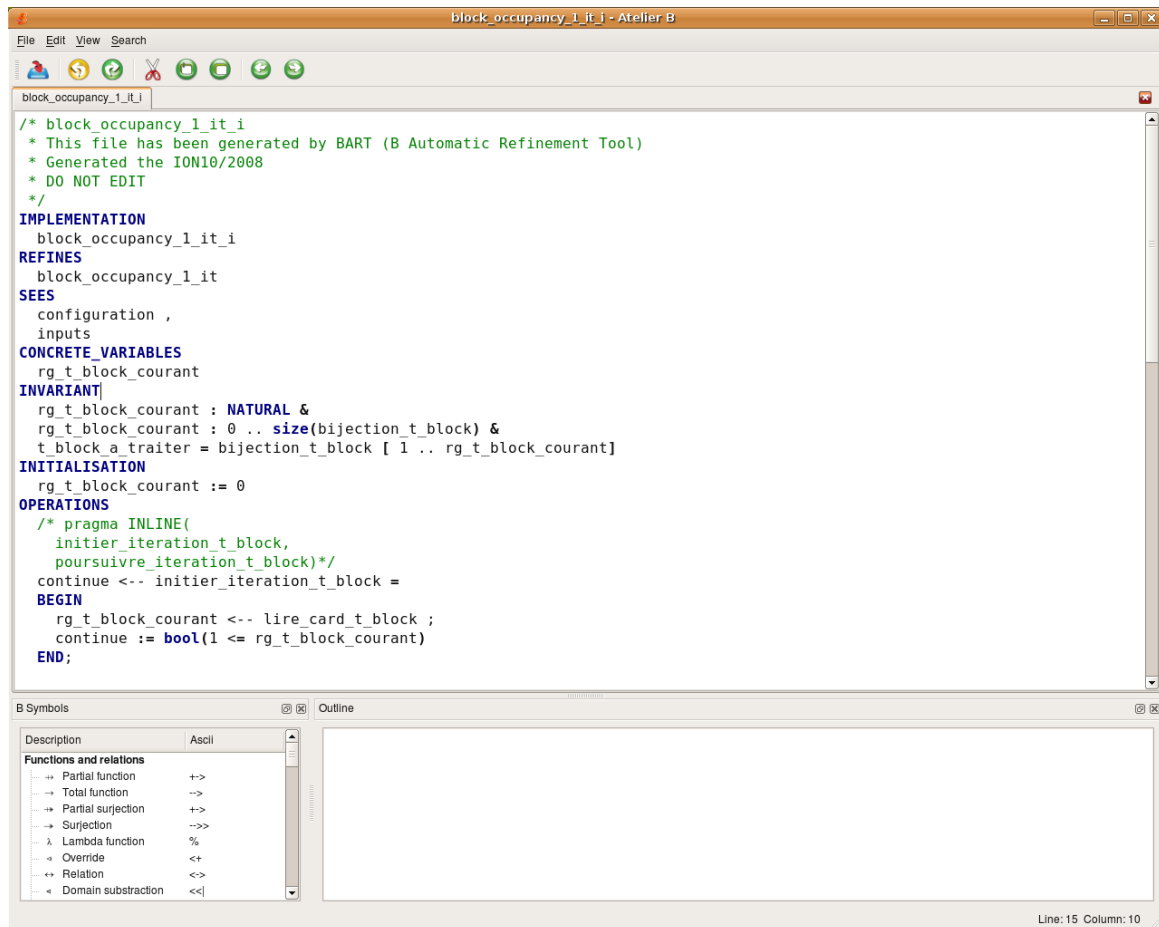
Comme expliqué sur la capture précédente, nous voyons que depuis l'interface principale, l'utilisateur a accès facilement et efficacement à la gestion de ses projets B.

C'est à partir de cette interface que la majeure partie de la gestion des projets B s'effectue. Nous reviendrons dans la suite de ce document plus en détail sur l'utilisation de cette dernière.

### 3.3 Éditeur intégré

L'Atelier B est livré avec un éditeur spécifiquement adapté à l'édition de composants B. Celui-ci propose des fonctionnalités comme l'auto-complétion de mots-clé, le soulignement en cas d'erreur, etc.

Un aperçu de la fenêtre est donné ci-dessous :



Cette application offre les fonctionnalités suivantes :

- L'édition des composants,
- L'édition des fichiers PMM associés aux composants,
- L'édition des fichiers RMF associés aux composants,
- L'édition des fichiers PatchProver et PatchRaffiner,
- La coloration syntaxique,
- L'impression,



- L'autocomplétion et l'indentation automatique,
- La vérification syntaxique du parenthésage,
- Une navigation par onglets entre les fichiers.

Il inclut en outre un tableau permettant l'insertion aisée des symboles B.

Ce dernier s'utilise comme un éditeur de texte classique et a été conçu de façon à rendre son utilisation intuitive.

### 3.4 Interface en ligne de commande

Cette interface utilisateur permet l'utilisation de l'Atelier B depuis une console ainsi qu'une utilisation en mode semi-automatique par fichiers de commande.

Cette interface se présente comme un interpréteur de commandes, possédant en grande majorité les mêmes fonctionnalités que l'interface graphique de l'atelier B.

Le lancement de l'interface de l'Atelier B en ligne de commande s'effectue de la manière suivante : Une fois dans le répertoire contenant les binaires de l'Atelier B, lancez l'exécutable :

```
$ ./lanceBB
```

Sous environnement Windows, un raccourci est créé par défaut dans le menu démarrer (Raccourci nommé "AtelierB Batch interface").

Pour quitter l'interface utilisateur mode commande, il faut taper **quit** ou **q**.

Vous pouvez utiliser cette interface de deux manières :

1. De manière interactive, ou
2. Avec un fichier de commandes

#### Utilisation d'un fichier de commandes

Un fichier de commandes peut contenir :

- Des commentaires : lignes commençant par "#",
- Des commandes Atelier B : forme longue ou abrégée,

Exemple de fichier de commandes :

```
#-----  
#Ceci est un commentaire  
#-----  
  
# liste des projets :  
show_projects_list  
  
# utilisateurs du projet library  
spul library  
  
#fin du fichier de commandes
```

Pour exécuter un fichier de commandes (sous UNIX uniquement), il faut taper l'une des commandes suivantes :

```
lanceBB -i=nom_fichier
ou
lanceBB < nom_fichier
ou lanceBB << END
contenu du fichier de commandes
END
```

### Utilisation de l'interpréteur

Certaines commandes ont des paramètres par défaut. Pour les commandes de gestion des projets, l'interpréteur mémorise le dernier nom de projet tapé. De la même manière, l'interpréteur mémorise le dernier nom de composant tapé. Pour utiliser ce paramètre par défaut, il faut juste taper **<return>**.

Exemple :

```
bbatch 3> typecheck MM_1
....
bbatch 4> pogenerate
pogenerate MM_1 1 ? (yes=return)
...
```

### Utilisation de l'aide interactive

La commande *help* affiche la liste des commandes disponibles. La liste est affichée de la façon suivante :

General commands :

```
(cd ) change_directory
...
(v  ) version_print
```

Project level commands :

```
(add ) add_definitions_directory
(apl ) add_project_lib
...
(spl ) show_projects_list
```

Machine level commands (available after open\_project) :

```
(b2c ) ComenCtrans  
(af  ) add_file  
...  
(us  ) unproved_status  
(vr  ) verify_rule
```

Les commandes sont présentées dans l'ordre suivant :

1. Commandes générales,
2. Commandes de gestion des projets,
3. Commandes applicables à des composants : il faut ouvrir un projet pour pouvoir utiliser ces commandes.

Les raccourcis des commandes sont indiqués entre parenthèses juste avant le nom de la commande.

Vous pouvez également obtenir de l'aide sur une commande particulière en tapant la commande :

```
help nom_commande  
ou  
h nom_commande
```

Par exemple :

```
bbatch 9> help help  
help [command] get help on commands
```

## 4 Pour démarrer

Les chapitres suivants de ce document décrivent toutes les fonctions fournies par l'Atelier B.

Ce chapitre décrit plus particulièrement la démarche générale permettant d'utiliser les fonctions de l'Atelier B.

En résumé, afin de démarrer un développement avec l'Atelier B, vous devez :

1. Créer un projet (c.f. section 4.1.2),
2. Ouvrir ce projet (c.f. section 4.1.4),
3. Ajouter des composants à ce projet (c.f. section 4.3.1).

Ou plus simplement, si vous disposez d'une archive de projet, vous pouvez restaurer cette dernière (c.f. section 4.2.5) et commencer à travailler.

Une fois ces étapes accomplies vous pouvez commencer à appliquer la méthode B sur vos composants ; il est possible de :

1. Effectuer l'analyse syntaxique et le contrôle de types (c.f. section 5.3),
2. Générer les obligations de preuve (c.f. section 5.5),
3. Démontrer automatiquement une partie de ces obligations de preuve (c.f. section 5.7),
4. Afficher les obligations de preuve (c.f. section 5.6),
5. Utiliser le prouveur interactif pour démontrer les obligations de preuve restantes (c.f. section 5.8).

Après avoir créé les implémentations de votre projet, vous serez en mesure de :

1. Contrôler le langage des implémentations (c.f. section 5.10),
2. Traduire le projet en langage cible en utilisant un traducteur (c.f. section 5.12).

Lors de ces phases du développement, utilisez les fonctions d'analyse de l'Atelier B, pour :

1. Afficher l'état d'avancement du projet ou d'un composant (c.f. sections 6.2 et 6.3),
2. Afficher des graphes de dépendances entre les composants (c.f. section 6.4),

Vous pouvez également utiliser des fonctions de documentation pour produire automatiquement des documentations au format des traitements de texte L<sup>A</sup>T<sub>E</sub>X, PDF, ou RTF (c.f. section 7.2).

Lorsque vos projets atteignent une taille importante, vous pouvez :

1. Les archiver pour en faire des sauvegardes (c.f.section 4.2.4),
2. découper vos gros projets en plusieurs petits projets en utilisant la notion de bibliothèques (c.f. section 4.2.2).

## 4.1 Gestion des projets

Un projet géré par l'Atelier B est défini par :

- Son nom,
- Un répertoire Base de Données Projet où seront rangés tous les fichiers créés par l'AtelierB (BDP),
- Un répertoire où seront rangées les traductions des composants vers les différents langages, en fonction des traducteurs disponibles,
- Un ensemble de fichiers sources B; ces fichiers peuvent être répartis dans plusieurs répertoires.

La création de ces répertoires et des fichiers sources B est à la charge de l'utilisateur en mode commande. L'interface graphique crée les répertoires si besoin.

L'Atelier B a été conçu de sorte qu'il puisse gérer des projets multi-utilisateurs. Plusieurs utilisateurs peuvent ainsi travailler simultanément sur le même projet.

L'Atelier B utilise dans son installation idéale, les droits du groupe UNIX `atelierb` pour résoudre les problèmes de droits UNIX entre ces utilisateurs (c.f. paragraphe 2.1.7).

Il vous est toutefois possible de l'utiliser en session "mono-utilisateur", mais vous risquez de perdre en souplesse d'utilisation.

Les projets gérés par l'Atelier B peuvent également être liés entre eux. L'utilisation de bibliothèques permet un découpage des projets de taille industrielle en plusieurs petits projets.

L'Atelier B offre également des fonctions permettant d'archiver des projets. Ces fonctions peuvent être utilisées pour faire des sauvegardes, ou des copies de projets.

Pour des raisons d'archivage et de portabilité, nous encourageons les utilisateurs de l'Atelier B à adopter une architecture de projet respectant les règles suivantes :

- Il est préférable que tous les utilisateurs d'un même projet appartiennent au même groupe UNIX,
- Les répertoires du projet doivent dans l'idéal avoir une racine commune. Ainsi, si `$rep_projet` est le répertoire d'un projet, il est préférable que la BDP et le répertoire de traduction soient situés dans des sous-répertoires de `$rep_projet`,
- Les fichiers sources B doivent préférentiellement être situés dans la même arborescence que `$rep_projet`. Cela simplifie l'archivage des projets, qui, en cas de sauvegarde de sources disséminés, ramènera tout au sein de l'archive dans un répertoire commun.

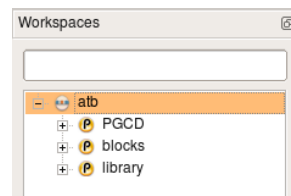
### 4.1.1 Liste des projets

#### Description

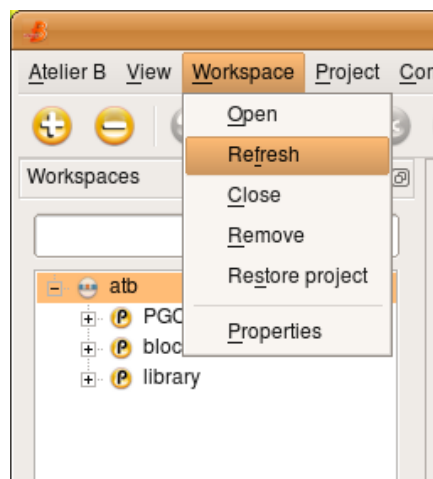
Cette fonction permet l’affichage de la liste des projets visibles par l’utilisateur de l’Atelier B. Notez que celle-ci est visible en permanence dans l’interface graphique sous réserve que l’espace de travail soit ouvert, comme le montrent les captures d’écran précédentes.

#### Interface utilisateur graphique

Une fois l’espace de travail ouvert, la liste des projets est directement accessible depuis la fenêtre principale de l’application.



Il peut être utile de mettre à jour la liste des projets, cela s’effectue en cliquant sur le menu “Workspace -> refresh”.



#### Interface mode Commande

On suppose l’interface utilisateur déjà lancée. Pour obtenir la liste des projets, il suffit de taper la commande suivante :

```
show_projects_list
```

ou

```
spl
```



La liste est alors affichée de la façon suivante :

```
Printing Projects list ...
```

```
    project1
    project2
```

```
End of Projects list
```

### 4.1.2 Création d'un projet

#### Description

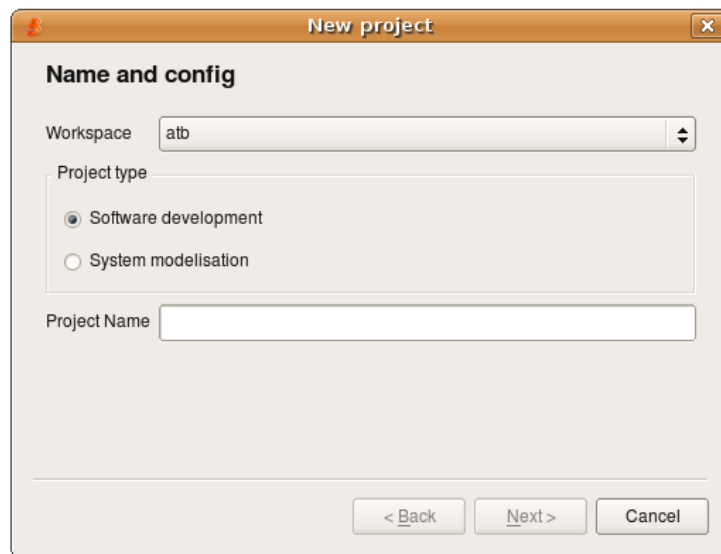
La fonction *Attach project* permet à tout utilisateur de déclarer auprès de l'Atelier B un nouveau projet. Les informations liées à cette déclaration de projet sont :

- Le nom du nouveau projet,
- L'endroit où conserver les informations qui permettront la gestion par l'Atelier de ce projet (BDP),
- L'endroit où ranger les sources (et makefiles s'ils existent) générés lors de la traduction des implantations en langage informatique.
- Le type de projet (facultatif) : SOFTWARE pour les projets B logiciels, SYSTEM pour les projets en B-Événementiel. Par défaut, un projet est considéré logiciel (SOFTWARE).

Pour référencer un projet donné, seuls les deux répertoires BDP et traduction sont nécessaires à l'Atelier. La création de ces deux répertoires est à la charge de l'utilisateur. Les composants du projet peuvent, eux, être éparpillés dans le système de fichiers, leur chemin d'accès et leur nom sont enregistrés dans un fichier présent dans la BDP et nommé `nom_projet.db`.

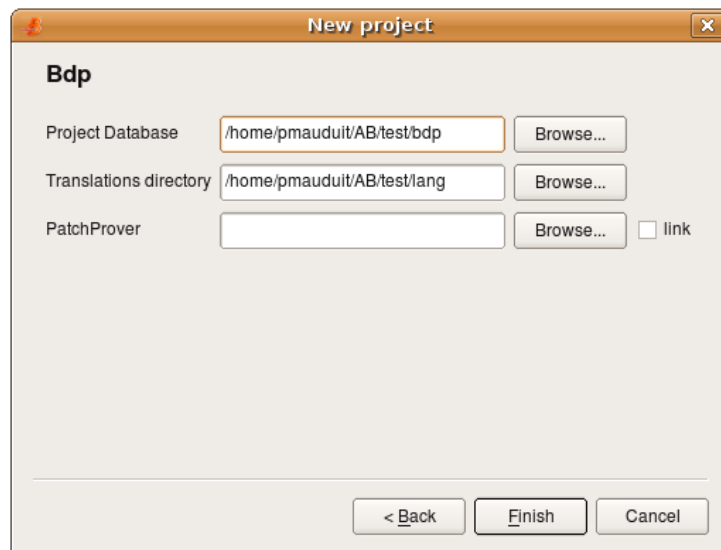
#### Interface utilisateur graphique

Afin de créer un projet depuis l'interface, il suffit de cliquer sur le menu "AtelierB -> New Project", ou d'utiliser soit le raccourci "CTRL+P", soit le click droit sur l'espace de travail voulu, puis le menu "New -> Project". L'assistant de création de projet apparaît alors :



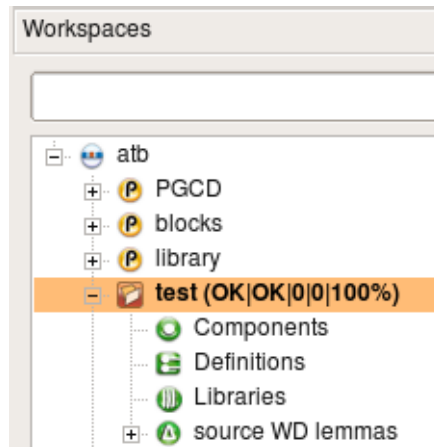
Il vous faudra choisir entre B-logiciel et B-Événementiel.

Le fait de saisir un nom permet à l'interface de déterminer des répertoires adéquat pour la localisation du projet, ce qui sera tout de même personnalisable sur la deuxième page de l'assistant :



Vous pouvez bien sûr utiliser tout de même les boutons “Browse” afin de faciliter le choix des répertoires.

Une fois la configuration faite et l'assistant terminé, le projet nouvellement créé devrait apparaître dans la vue des espaces de travail, et être automatiquement ouvert, comme le montre la capture d'écran suivante :



### Interface mode Commande

Afin de créer un nouveau projet, il faut effectuer les opérations suivantes :

- Choisissez et créez un répertoire pour la Base de données du projet.
- Choisissez et créez un répertoire pour les traductions.
- Lancez l'interface en ligne de commande
- Créez le projet B en tapant la commande suivante :  

```
create_project nom_projet chemin_bdp chemin_lang SOFTWARE
```

ou  

```
crp nom_projet chemin_bdp chemin_lang SOFTWARE
```
- Vérifiez que le projet a bien été créé en tapant la commande :  

```
show_projects_list
```

ou  

```
spl
```

**Remarque :** Vous n'êtes pas obligé de taper le chemin complet des répertoires ; vous pouvez donner un chemin relatif par rapport au répertoire depuis lequel vous avez lancé l'Atelier B. La commande précédente permet de créer un projet B Logiciel. Si vous souhaitez faire du B-Événementiel, remplacez "SOFTWARE" par "SYSTEM".

#### 4.1.3 Suppression d'un projet

##### Description

La fonction *Detach project* permet à tout utilisateur de supprimer un projet existant. Les fichiers intermédiaires produits par l'Atelier B dans le répertoire de la Base de données du projet sont alors détruits.

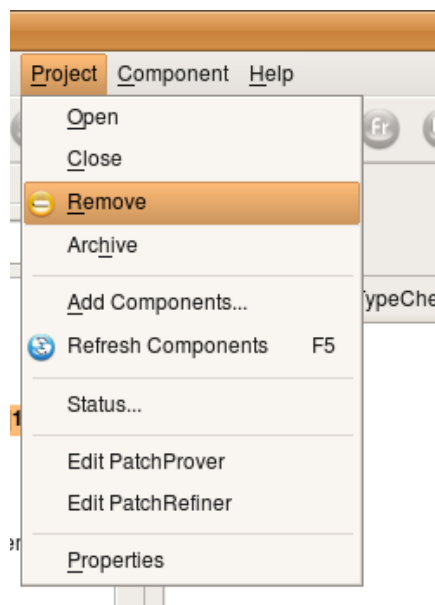
Les informations qui ne sont pas supprimées sont :

- Les fichiers sources B,
- Les fichiers de règles utilisateur (\*.pmm),
- Les documentations de projet générées automatiquement,
- Les traductions,
- Le répertoire de base de données du projet,
- Le répertoire des traductions.

Afin d'effectuer un nettoyage complet du projet vous devez, après avoir supprimé le projet avec l'Atelier B, détruire manuellement ces fichiers et répertoires.

### Interface utilisateur graphique

Cette action est accessible depuis le menu *"Project -> remove"*, ou via un click droit dans la vue de l'espace de travail :



Le comportement de cette commande reste similaire toutefois aux anciennes versions ; les fichiers créés par l'Atelier B sont détruits, mais les informations suivantes sont conservées :

- Les fichiers sources B,
- Les fichiers de règles utilisateur (\*.pmm),
- Les documentations de projet générées automatiquement,

- Les traductions,
- Le répertoire de base de données du projet,
- Le répertoire des traductions.

Afin de réaliser un nettoyage complet du projet, vous devrez vous-même, après avoir supprimé le projet dans l'interface, détruire manuellement les fichiers et répertoires restants.

### **Interface mode commande**

L'interface est supposée lancée ; Pour supprimer un projet nommé *proj*, il suffit de taper la commande suivante :

```
remove_project proj  
ou  
rp proj
```

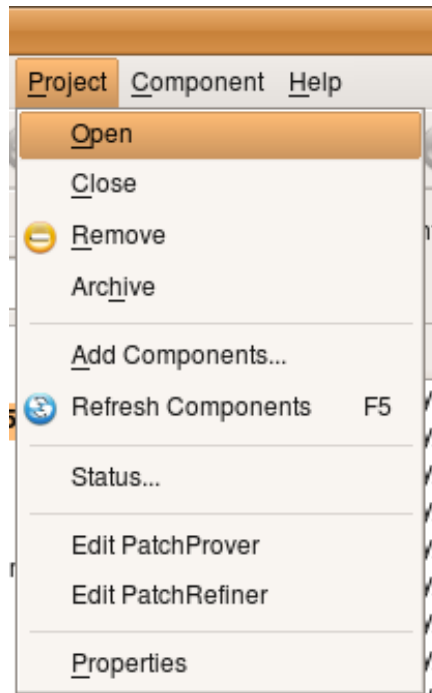
#### **4.1.4 Ouverture et fermeture d'un projet**

##### **Description**

L'ouverture permet l'accès aux composants d'un projet. La fermeture permet à l'inverse de fermer le projet courant.

##### **Interface utilisateur graphique**

Pour ouvrir un projet dans l'interface de l'Atelier B 4, il suffit, par exemple de passer par le menu "Project -> Open", ou bien d'effectuer un click droit dans la vue de l'espace de travail sur le projet voulu et de cliquer sur "Open".



Une fois le projet ouvert, la liste des composants s'affiche :

Workspaces

alib

- PGCD
- blocks (-|191|65|65%)**
  - Components
  - Definitions
  - Libraries
  - source WD lemmas
  - library
  - test

blocks: Components

☐ Hierarchical view

Filter:

Component	TypeChecked	POs Generated	Proof Obligations	Proved	Unproved	B0 Checked
block_occupancy	-	-	-	-	-	-
block_occupancy_1	OK	OK	7	7	0	OK
block_occupancy_1_i	OK	OK	33	21	12	OK
block_occupancy_1_it	OK	OK	6	6	0	OK
block_occupancy_1_it_i	OK	OK	19	9	10	-
block_occupancy_2	OK	OK	5	5	0	OK
block_occupancy_2_i	OK	OK	31	18	13	OK
block_occupancy_3	OK	OK	4	4	0	OK
block_occupancy_3_i	OK	OK	12	5	7	OK
block_occupancy_4	OK	OK	4	4	0	OK
block_occupancy_4_i	OK	OK	2	0	2	OK
block_occupancy_5	OK	OK	4	4	0	OK
block_occupancy_5_i	OK	OK	35	24	11	OK
block_occupancy_i	-	-	-	-	-	-
block_occupancy_it	OK	OK	6	6	0	OK
block_occupancy_it_i	OK	OK	19	9	10	-
configuration	OK	OK	0	0	0	OK
inputs	OK	OK	4	4	0	OK

Cette vue est évidemment paramétrable, et il est possible de trier selon une colonne donnée en cliquant sur l'en-tête de celle-ci.

Notez que cette version vous permet d'ouvrir plusieurs projets B simultanément.

À l'inverse, la fermeture d'un projet se fait en utilisant les menus "Project -> Close" de l'interface. Cela a pour effet notamment de vider la liste des composants.

**Interface mode commande**

Afin d'ouvrir un projet `proj` il suffit de taper la commande suivante :

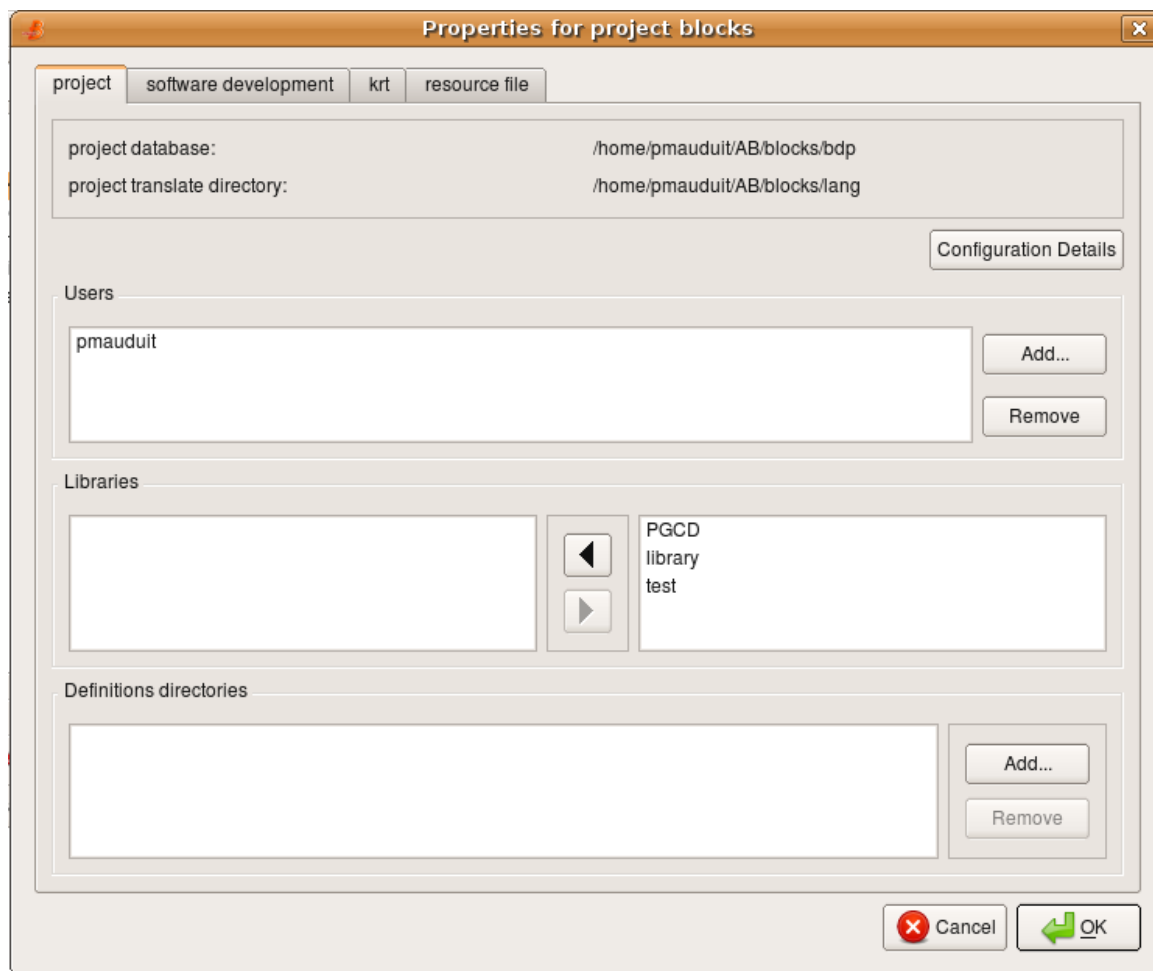
```
open_project proj  
ou  
op proj
```

Une fois un projet ouvert, il est possible de le refermer via la commande suivante :

```
close_project  
ou  
clp
```

## 4.2 Projets : Gestion avancée

Cette section s'attarde sur la gestion avancée des projets. En ce qui concerne l'interface graphique de l'Atelier B, les principales fonctionnalités décrites ci-après sont accessibles via le menu "Project -> Properties", donnant accès à l'interface suivante :



Les équivalents en ligne de commande pour l'interface en mode commande seront donnés par la suite.

### 4.2.1 Gestion des utilisateurs

#### Description

Afin de permettre un contrôle d'accès aux objets gérés par l'Atelier B, il est possible d'ajouter et/ou supprimer des utilisateurs ; par ailleurs, tous les utilisateurs définis sur un projet ont les mêmes droits d'accès en lecture / écriture sur ce dernier. Il est tout de même indispensable d'avoir au moins un utilisateur défini dans la liste.



### Interface utilisateur graphique

Cliquez simplement sur les utilisateurs que vous voulez supprimer, puis cliquez sur le bouton *remove*.

Appuyez sur le bouton *Add ...* pour faire apparaître une boîte de dialogue permettant la saisie d'un nom d'utilisateur. Si vous souhaitez donner un accès à tout utilisateur du système, vous pouvez spécifier un caractère joker en utilisant le caractère '\*'.

#### 4.2.2 Gestion des bibliothèques

##### Description

Dans le cadre du développement de projets de taille importante, il est essentiel de pouvoir :

- Utiliser des bibliothèques de composants prédéfinis
- Structurer un projet important en plusieurs sous-projets

S'ils le souhaitent, les utilisateurs peuvent lier leurs projets à d'autres projets gérés par l'Atelier. Pour cela, ils disposent de la fonction *Add library*.

Tout projet qui est accessible peut devenir bibliothèque du projet.

Lorsqu'une bibliothèque est liée à un projet, l'utilisateur du projet peut faire des liens (SEES, IMPORTS, ...) vers les composants de cette bibliothèque.

La fonction *Add library* vérifie que la bibliothèque à ajouter n'est pas déjà déclarée dans le projet.

Si un composant est défini dans plusieurs bibliothèques du projet, alors le composant qui sera pris en compte est celui défini dans la bibliothèque qui a été ajoutée en premier. En cas de doute sur les composants qui sont pris en compte, il peut être utile d'afficher le graphe de dépendance du projet.

### Interface utilisateur graphique

Comme le montre la capture d'écran précédente, la liste des bibliothèques potentielles apparaît dans le cadre *Libraries*, dans la boîte de droite. Utilisez la souris pour sélectionner le projet à rajouter en tant que bibliothèque puis cliquez sur la flèche dirigée vers la boîte de gauche pour ajouter la bibliothèque sélectionnée au projet.



À l'inverse, sélectionnez la bibliothèque à exclure du projet, puis cliquez sur la flèche dirigée vers la droite, pour exclure une bibliothèque du projet.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée. Pour ajouter la librairie `nom_lib` au projet `proj`, il faut taper la commande suivante :

```
add_project_lib nom_lib  
ou  
apl nom_lib
```

Le listing des bibliothèques d'un projet s'obtient en exécutant la commande suivante :

```
show_project_libs_list nom_proj  
ou  
spll nom_proj
```

Enfin, la suppression d'une bibliothèque d'un projet s'effectue de la façon suivante :

```
remove_project_lib nom_proj nom_lib  
ou  
rpl nom_proj nom_lib
```

### 4.2.3 Gestion des répertoires de fichiers de définitions

#### Description

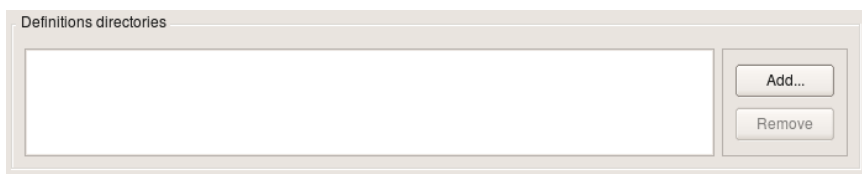
Les fichiers de définitions permettent le regroupement de définitions communes pour plusieurs composants.

Leur description est donnée au chapitre 2.3 du manuel de référence du langage B.

Cette partie décrit les procédures à suivre pour ajouter et supprimer d'un projet B un nouveau répertoire pouvant contenir des fichiers de définitions utilisés par les composants de ce projet.

#### Interface utilisateur graphique

Toujours sur la page de propriétés des projets (c.f. capture d'écran précédente), dans le cadre "Definitions directories", cliquez sur le bouton "Add" pour afficher une boîte de dialogue vous permettant la sélection d'un répertoire.



Inversement, pour supprimer un répertoire de définitions, cliquez sur ce dernier dans la liste prévue à cet effet, et cliquez sur *Remove*.

### Interface utilisateur en mode commande

L'interface est déjà lancée. Afin d'ajouter un répertoire **rep** au projet **proj**, il suffit de taper la commande suivante :

```
add_definitions_directory proj rep
ou
add proj rep
```

Attention, il est nécessaire de donner un chemin complet (absolu) du répertoire.

À l'inverse, afin de supprimer un répertoire de définitions, il faut utiliser la fonction *Remove definitions directories*. Attention toutefois à s'assurer qu'aucun composant du projet ne dépend d'un fichier de définitions présent dans le répertoire. La commande à utiliser est la suivante :

```
remove_definition_directory nom_proj rep
ou
rdd nom_proj rep
```

La encore il est nécessaire de donner le chemin complet - absolu - du répertoire.

#### 4.2.4 Archivage

##### Description

Cette fonction permet d'archiver l'ensemble des fichiers constituant un projet géré par l'Atelier B.

L'archive créée est un fichier compressé par la bibliothèque zlib, au format tar, ayant pour suffixe .arc.

Cette fonction peut être utilisée pour :

- Faire la sauvegarde d'un projet,
- Faire une copie d'un projet (pour le transférer sur une autre machine, par exemple)

Il existe trois options pour l'archivage :

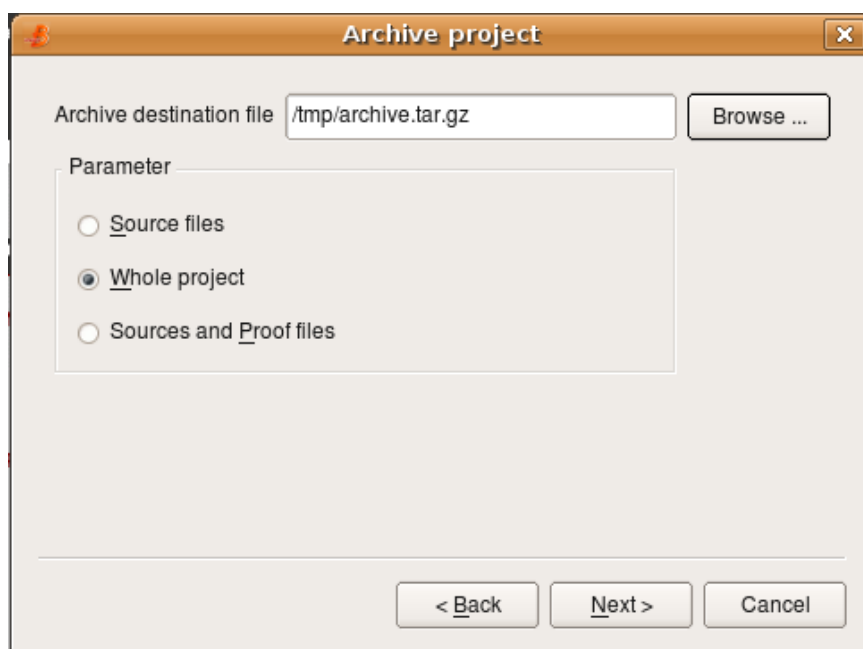
- Sauvegarde de l'ensemble des fichiers sources B (mch, ref et imp).
- Sauvegarde de l'ensemble des fichiers sources B et des fichiers de preuve. Avec cette option l'Atelier B enregistre aussi tous les fichiers nécessaires à la preuve, présents dans la base de données du projet

- Fichiers po : obligations de preuve
- Fichiers pmi : démonstrations,
- Fichiers pmm : règles utilisateur,
- Fichiers stc : état du composant lors de la génération
- Fichier PatchProver : tactiques utilisateur
- Sauvegarde de tout le projet :
  - Fichiers sources B
  - Fichiers présents dans le répertoire de base de données du projet
  - Fichiers présents dans le répertoire des traductions,

Lorsque l'on archive tout le projet, toutes les informations sont enregistrées. Par conséquent, lorsque le projet sera désarchivé (Chapitre 4.2.5) l'utilisateur le retrouvera dans le même état ; il ne sera donc pas obligé de refaire le contrôle de types, la preuve ...

### Interface utilisateur

Afin d'archiver un projet au moyen de l'interface graphique, il suffit, après avoir ouvert l'espace de travail contenant le projet à archiver, de cliquer sur ce dernier, et de sélectionner dans le menu l'option archive. Cela a pour effet d'ouvrir un assistant d'archivage se présentant sous la forme suivante :



Sélectionnez le fichier archive de destination, ainsi que le paramètre d'archivage puis cliquez sur le

bouton Suivant (*Next*).

### Interface mode commande

Pour archiver un projet à l'aide de l'interface en ligne de commande, il suffit de faire appel à la commande `archive`, ou `arc`. Cette commande prend en entrée 3 paramètres :

- Le nom du projet,
- Le chemin d'accès au fichiers archive,
- Le type d'archive :
  - 0 archivage des sources B,
  - 1 archivage de tout le projet,
  - 2 archivage des sources B et des fichiers de preuve.

#### 4.2.5 Restauration

##### Description

Cette fonction permet la restauration d'un projet géré par l'Atelier B à partir des informations contenues dans une archive créée avec la fonction décrite au paragraphe précédent. Trois options de restauration sont possibles :

- Restauration des fichiers sources B,
- Restauration des fichiers sources B et des fichiers de preuve,
- Restauration de tout le projet.

Une restauration crée toujours un nouveau projet.

Lors de la restauration les informations suivantes sont perdues :

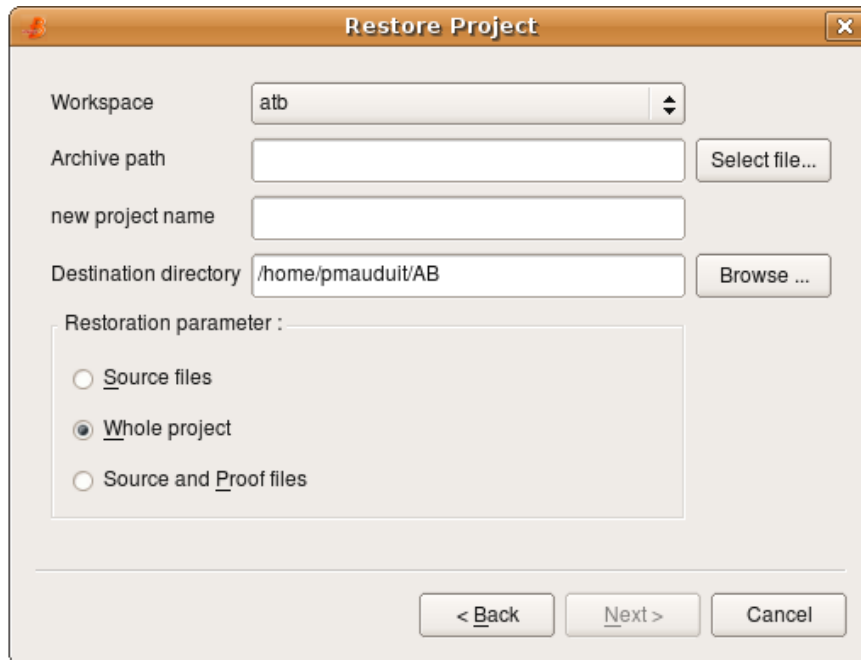
- La liste des bibliothèques du projet,
- La liste des utilisateurs du projet.

Remarque : Vous pouvez restaurer les sources et les fichiers de preuves d'un projet créé avec une version précédente de l'Atelier B.

Vous ne pouvez pas effectuer une restauration des sources et des fichiers de preuves si votre archive a été élaborée pour ne contenir que les sources B. L'Atelier B vous avertira que la restauration est impossible.

### Interface utilisateur

Un assistant de restauration vous est proposé dans la nouvelle interface afin de vous aider dans cette tâche :



Choisissez tout d'abord un espace de travail dans lequel vous souhaitez restaurer votre projet, puis le chemin d'accès vers l'archive ; ensuite, donnez un nom au nouveau projet, et sélectionnez un répertoire dans lequel restaurer votre projet. Enfin n'oubliez pas de sélectionner le paramètre de restauration voulu avant de cliquer sur "Suivant" ("Next").

Si la restauration s'est correctement déroulée, le nouveau projet doit apparaître dans l'espace de travail. Les composants sont automatiquement attachés au projet, ainsi que les répertoires de définition.

### Interface mode commande

L'interface utilisateur est considérée comme lancée ; pour restaurer un projet, il faut utiliser la commande `restore` ou `res`.

Cette commande prend en argument 4 paramètres :

- Le chemin d'accès au fichier archive,
- Le type de restauration :
  - 0 restauration des sources B,
  - 1 restauration de tout le projet,
  - 2 Restauration des sources B et des fichiers de preuve.
- Le nom du projet restauré ; vous devez donner le nom d'un nouveau projet.

- Le chemin d'accès du projet, si vous ne souhaitez pas que les répertoires soient créés dans le répertoire courant.

#### 4.2.6 Lire des informations sur un projet

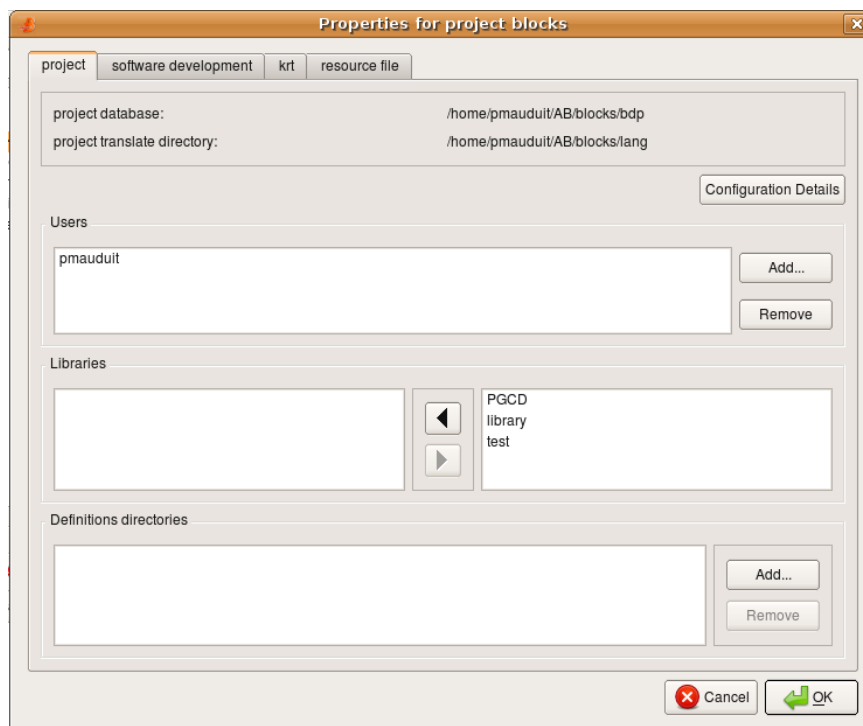
##### Description

Cette fonction affiche, pour un projet, les informations suivantes :

- Le chemin d'accès de la base de données du projet,
- Le chemin d'accès du répertoire des traductions,
- Le nom du gérant du projet,
- La liste des bibliothèques du projet.

##### Interface utilisateur graphique

L'interface est déjà lancée, l'espace de travail est connecté. Pour obtenir des informations sur un projet, il suffit de demander l'affichage de la fenêtre de propriétés du projet :



##### Interface mode commande

L'interface utilisateur est déjà lancée. Pour avoir les informations sur le projet *proj*, il faut taper la commande suivante :

```
infos_project proj  
ou  
ip proj
```

Les informations sont affichées de la façon suivante :

```
bbatch 2> ip TEST  
Name : TEST  
Database path : /home/pmaudit/AB/TEST/bdp  
Translation path : /home/pmaudit/AB/TEST/lang  
Manager : user
```



## 4.3 Gestion des composants

### 4.3.1 Ajout de composants

#### Description

Un projet B est constitué par une liste de composants B situés dans des fichiers texte. Ces composants sont soit directement rattachés au projet, soit attachés par l'intermédiaire de bibliothèques.

Les fichiers sources B, sont des fichiers textes contenant un ou plusieurs composants. Une fois qu'un fichier source est attaché à un projet, le gestionnaire de projet analyse quels sont les composants contenus dans le fichier et permet la gestion de ces composants.

L'Atelier B n'accepte comme fichier de sources B, que des fichiers dont le nom se termine par l'un des quatre suffixes suivants : .mch, .ref, .imp et .mod. Le contenu de chaque type de fichiers est le suivant :

- un fichier de nom Ident.mch doit contenir un et un seul composant : une machine abstraite nommée Ident.
- un fichier de nom Ident.ref doit contenir un et un seul composant : un raffinement nommé Ident.
- Un fichier de nom Ident.imp doit contenir un et un seul composant : une implémentation nommée Ident.
- Un fichier de nom Ident.mod doit contenir un composant nommé Ident. Il peut éhgalement contenir l'ensemble des raffinements de ce composant, ainsi que des modules (une machine abstraite et tous ses raffinements) importés par l'implantation du composant. enfin, pour chaque module présent dans le fichier, des modules importés par ce module peuvent également être présents. Un fichier de suffixe .mod permet ainsi de regrouper en un seul fichier un module B, ou une sous-partie du projet comprenant un module et des modules importés par ce module.

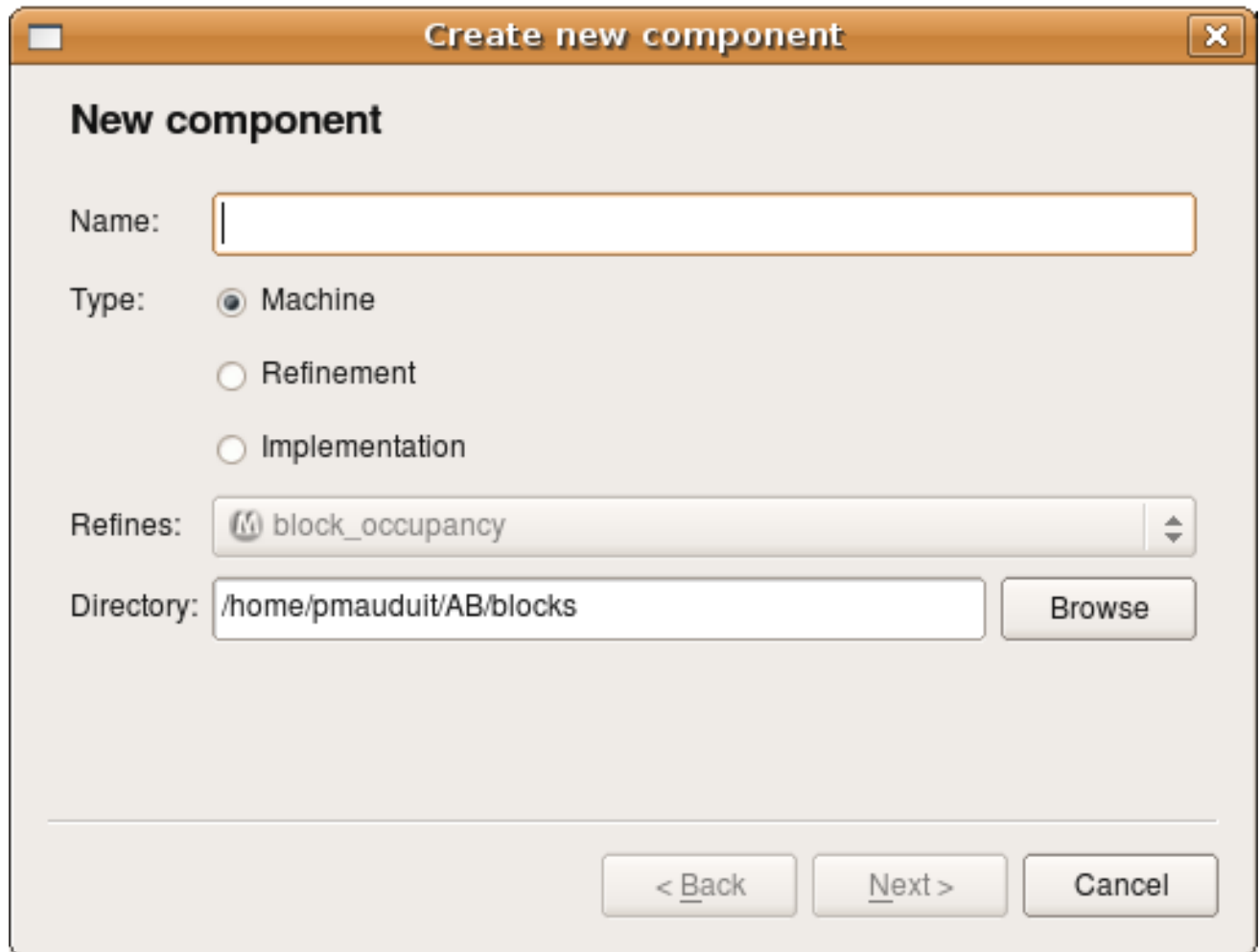
Si le contenu des fichiers ne respecte pas ces règles le fichier sera quand même attaché au projet, mais un message d'erreur sera affiché dans la fenêtre de lancement. Aucune autre opération ne pourra être effectuée sur le composant avant sa correction.

#### Interface utilisateur

Supposons l'interface déjà lancée, et le projet déjà ouvert. Pour ajouter des composants depuis des fichiers existants, vous pouvez utiliser le menu "Project -> Add components ...". Ce menu aura pour effet d'ouvrir une fenêtre de sélection de fichiers (la sélection pouvant être multiple). Une fois les fichiers désirés sélectionnés, confirmez en cliquant sur le bouton "Open".

Une autre solution consiste à créer un nouveau composant, si celui-ci n'existe pas encore. Pour cela, sélectionnez dans le menu contextuel "New -> Components ...", ou cliquez dans la barre d'outils sur

l'icône représentant un “+” bleu. S’affiche alors l’assistant suivant :

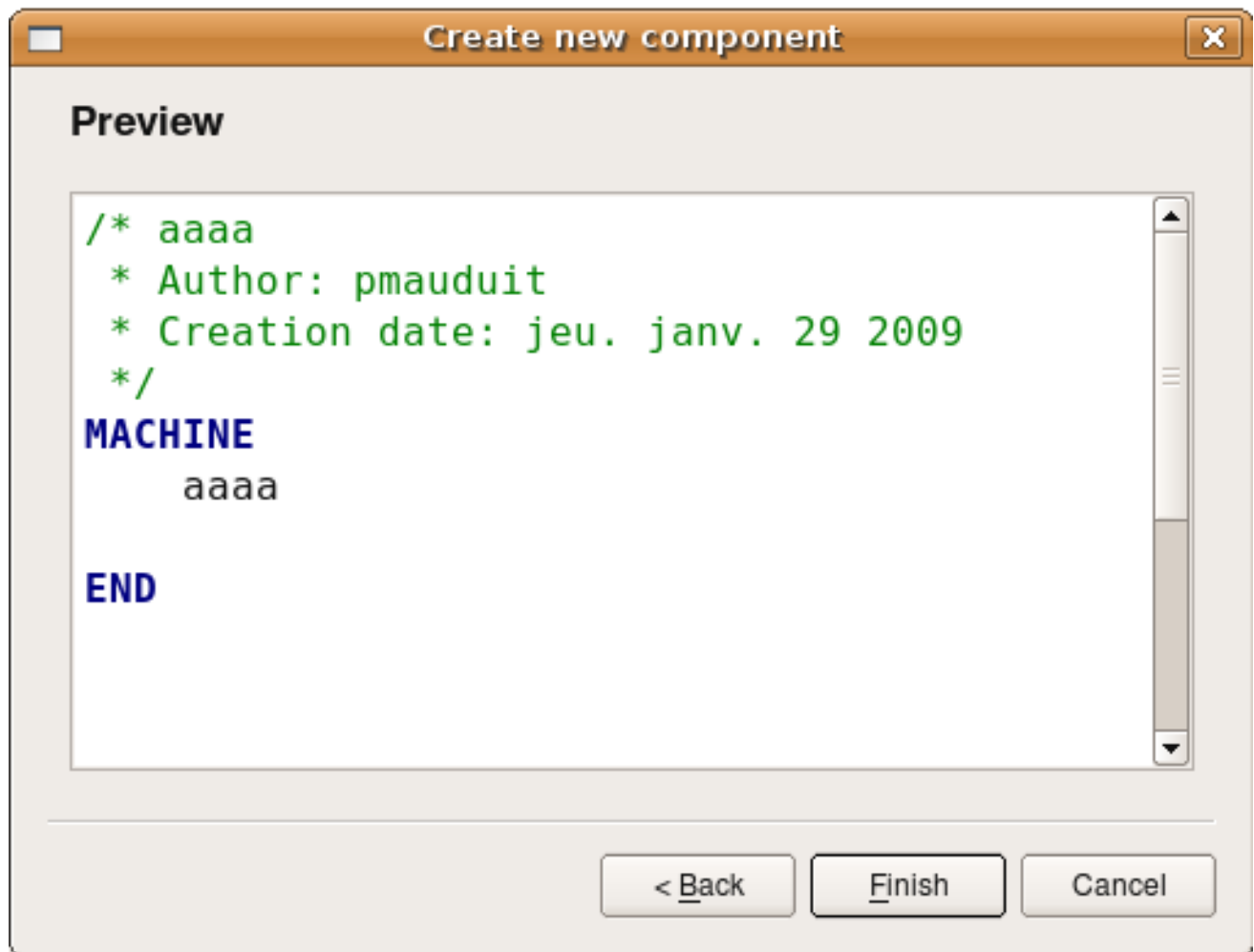


The image shows a graphical user interface window titled "Create new component". Inside the window, there is a section titled "New component". Below this title, there are several input fields and options:

- Name:** A text input field that is currently empty.
- Type:** A group of three radio buttons. The first, labeled "Machine", is selected. The other two are labeled "Refinement" and "Implementation".
- Refines:** A dropdown menu showing "block\_occupancy" with a small icon to its left.
- Directory:** A text input field containing the path "/home/pmauduit/AB/blocks". To the right of this field is a "Browse" button.

At the bottom of the window, there are three buttons: "< Back", "Next >", and "Cancel".

Une fois le nom du nouveau composant rentré, la sélection du type le composant raffiné éventuel et le chemin d'accès de destination, cliquez sur “suivant” (ou “Next”). L’assistant vous proposera alors un aperçu :



### Interface en ligne de commande

Supposons L'interface utilisateur déjà lancée, et que vous ayez déjà ouvert un projet. Pour ajouter des composants à un projet, il suffit d'effectuer les opérations suivantes :

1. Allez dans le répertoire où se trouve les fichier source B en utilisant la commande `cd` ou `change_directory`.
2. Afficher la liste des sources B présents dans ce répertoire en utilisant la commande `lsb` ou `list_sources_b`.  
Cette commande affiche la liste des fichiers ayant pour suffixe `mch`, `ref`, `imp` ou `mod` présents dans le répertoire courant.
3. Attachez le composant, en utilisant la commande suivante :  
af `nom_fichier`, ou `add_file nom_fichier`.

### 4.3.2 Exclusion de composants

Cette fonction permet d'exclure du projet courant un ou plusieurs composants.

Les fichiers contenant les composants ne sont pas détruits, ils sont seulement retirés de la liste des composants faisant partie du projet.

Lorsqu'un composant est exclu, les informations relatives à ce composant sont effacées de la BDP, à l'exception des informations suivantes :

- Fichiers sources B,
- Fichiers de règles utilisateur (pmm),
- Documentations de projet générées automatiquement,
- traductions

### Interface utilisateur graphique

L'interface graphique est considérée comme lancée, et vous avez ouvert un projet. Sélectionnez les composants que vous souhaitez enlever du projet, et sélectionnez dans le menu "component -> remove ...". (ou bien cliquez sur l'icône représentant un "-" bleu de la barre d'outils).

### Interface mode commande

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet. Pour détacher un composant du projet, il faut taper la commande suivante :

```
bbatch > remove_component nom_comp
```

ou

```
bbatch > rc nom_comp
```

Si `nom_comp` correspond à un composant inclus dans un fichier `.mod`, alors tous les composants présents dans le fichier `.mod` seront supprimés.

### 4.3.3 Affichage des composants

Cette fonction affiche la liste des composants d'un projet, groupés par fichier source B. Les composants sont affichés dans l'ordre alphabétique par rapport aux noms de fichiers sources B.

Si les composants BB, BB\_1 et AA sont présents dans le même fichier source B (fichier `.mod`), alors les composants seront affichés de la manière suivante :

BB AA  
 BB  
 BB\_1

Sur des projets de taille industrielle le nombre de composants est souvent très important. Cette fonction offre plusieurs filtres permettant de voir un nombre réduit de composants et de faire une recherche lexicographique.

Les filtres disponibles sont :

- Filtre en fonction de l'utilisateur gérant du composant,
- Filtre en fonction du type de composant : machine, raffinement, implémentation.
- Filtre en fonction du nom des composants.

### Interface utilisateur graphique

L'interface utilisateur est déjà lancée, et un projet est ouvert. La liste des composants est affichée en permanence.

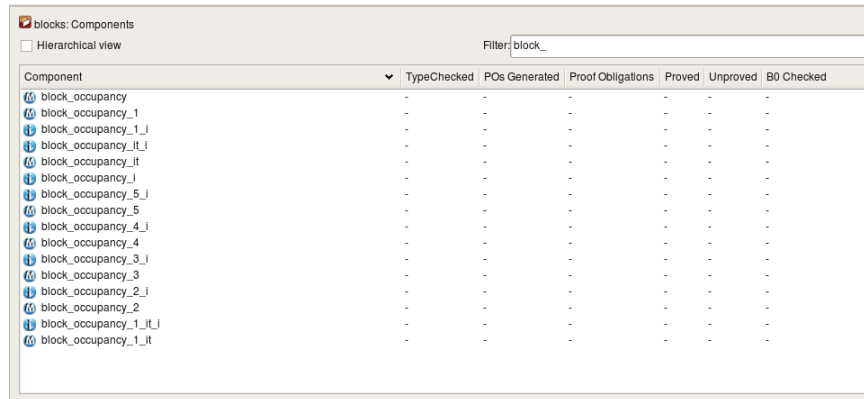
Les filtres disponibles au sein de l'interface sont un peu différents par rapport à l'interface en ligne de commande ; il est possible de :

- Filtrer sur le nom des composants (en donnant un filtre de recherche dans la zone de texte située au dessus de la vue des composants)
- D'activer la vue hiérarchique, qui classe les composants selon une arborescence de raffinements.
- Trier selon une colonne spécifique.

Ci-dessous un exemple d'affichage hiérarchique :

Component	TypeChecked	POs Generated	Proof Obligations	Proved	Unproved	B0 Checked
block_occupancy	-	-	-	-	-	-
block_occupancy_it	-	-	-	-	-	-
block_occupancy_1	-	-	-	-	-	-
block_occupancy_1_it	-	-	-	-	-	-
block_occupancy_1_it_it	-	-	-	-	-	-
block_occupancy_1_it_it_it	-	-	-	-	-	-
block_occupancy_2	-	-	-	-	-	-
block_occupancy_2_it	-	-	-	-	-	-
block_occupancy_3	-	-	-	-	-	-
block_occupancy_3_it	-	-	-	-	-	-
block_occupancy_4	-	-	-	-	-	-
block_occupancy_4_it	-	-	-	-	-	-
block_occupancy_5	-	-	-	-	-	-
block_occupancy_5_it	-	-	-	-	-	-
block_occupancy_it	-	-	-	-	-	-
block_occupancy_it_it	-	-	-	-	-	-
configuration	OK	-	-	-	-	-
inputs	OK	-	-	-	-	-

Ci-après un exemple d'utilisation de la zone de texte permettant le filtrage sur le nom :



### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet. Pour afficher la liste des composants du projet vous devez utiliser la commande :

```
show_machines_list
```

ou

```
sml
```

Cette fonction prend 5 paramètres optionnels :

- own : Si ce paramètre est égal à 1 seuls les composants dont vous êtes gérant sont affichés.
- mch : Si ce paramètre est égal à 0 les machines ne sont pas affichées
- ref : Si ce paramètre est égal à 0 les raffinements ne sont pas affichés.
- imp : Si ce paramètre est égal à 0 les implémentations ne sont pas affichées.
- nom : Ce paramètre permet de filtrer la liste en fonction des noms des composants. Il faut utiliser le caractère \*. Par exemple, pour avoir tous les composants dont le nom commence par la lettre S il faut donner la valeur 'S\*'.

#### 4.3.4 Propriétés des composants

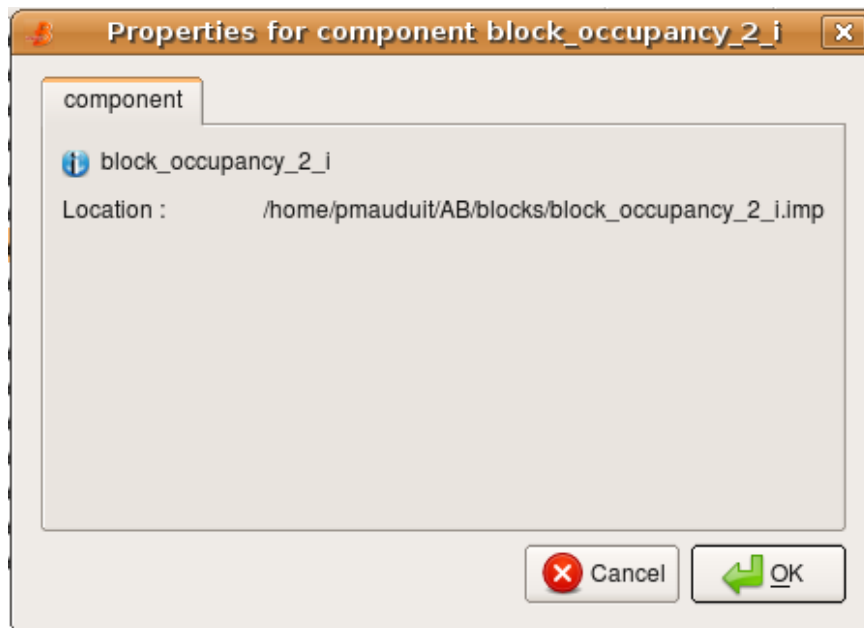
Cette fonction affiche, pour des composants du projet, les informations suivantes :

- Le chemin d'accès complet à la source,

- Le type de composant : machine, raffinement, ou implémentation.

### Interface utilisateur graphique

Afin d'obtenir les informations disponibles sur un composant, il suffit de le sélectionner et de cliquer sur le menu "Component -> Properties". Une fenêtre comme celle présentée ci-après apparaît alors :



Notez que la distinction sur le type de composant se fait au niveau de l'icône bleue utilisée sur l'interface.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet.

Pour obtenir des informations sur un composant `nom_comp`, il suffit de taper la commande suivante :

```
infos_composant nom_comp  
ou  
ic nom_comp
```

#### 4.3.5 Édition d'un composant

##### Description

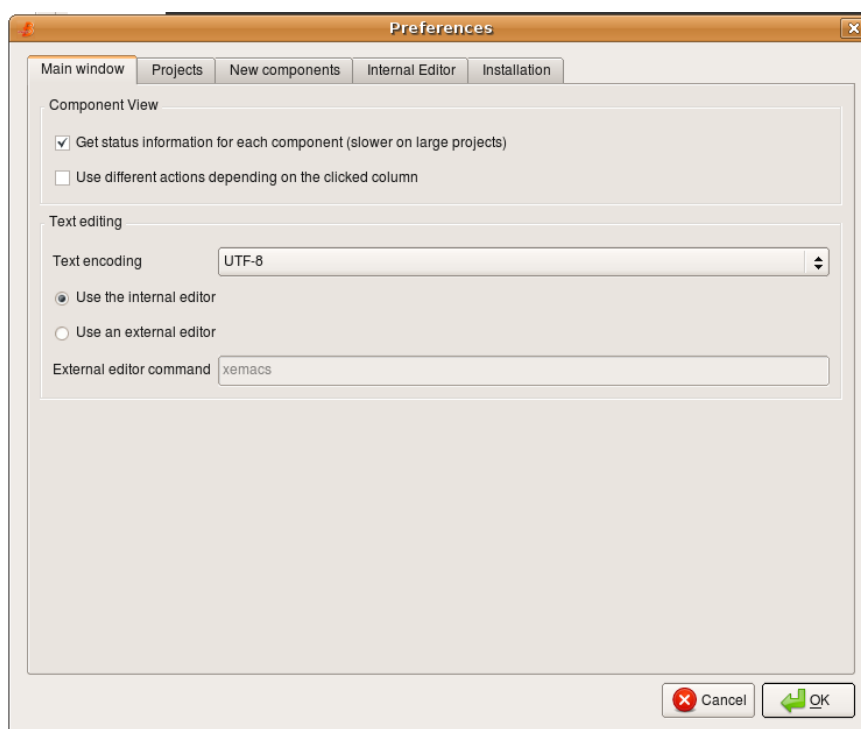
Cette fonction vous permet d'éditer le fichier source correspondant à un composant.

L'éditeur utilisé est par défaut l'éditeur interne, mais ceci reste configurable afin de vous laisser la possibilité de choisir l'éditeur de votre choix. Voyez le paragraphe "Paramètre utilisables" pour comprendre comment personnaliser l'éditeur.

### Interface utilisateur graphique

Pour éditer un composant dans l'interface graphique, en lançant l'éditeur par défaut, il suffit de double-cliquer dessus. L'éditeur se lance alors. Vous pouvez aussi passer par le menu "Component -> Edit", ou le raccourci clavier "Ctrl+E".

Vous pouvez personnaliser la configuration de votre éditeur dans la configuration de l'Atelier B. Pour ce faire, il suffit de cliquer sur le menu "Atelier B -> Preferences", et de configurer son éditeur (interne ou externe, ce dernier nécessitant la configuration du chemin d'accès) :



### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, un projet est déjà ouvert. Pour éditer un composant `nom_comp`, tapez la commande suivante :

```
edit nom_comp  
ou  
e nom_comp
```

### Paramètres utilisables



ATB*OPT_TOOLS_<SYSTEM>*Editor_Path
Par défaut utilise l'éditeur interne
Chemin d'accès à l'éditeur de texte.

### 4.3.6 Désarchiver un composant

#### Description

Cette fonction permet le désarchivage de fichiers sources B ou de fichiers de définition à partir d'une archive créée avec la fonction décrite au paragraphe 4.2.5.

La restauration est effectuée dans le projet courant.

Si le composant restauré n'est pas dans le projet courant, la fonction le rajoute automatiquement. Si le composant est déjà présent dans le projet, il est remplacé par le composant restauré.

#### Interface utilisateur graphique

La commande de restauration des sources n'est pas disponible via l'interface graphique de l'Atelier B à proprement parler. Toutefois cette dernière possède des fonctionnalités qu'il paraît intéressant de signaler ici ; il est en effet possible d'ajouter en masse des composants. Pour ce faire,

1. Décompressez l'archive contenant les sources à restaurer,
2. Ouvrez dans l'interface graphique de l'Atelier B le projet considéré,
3. Cliquez dans le menu "Project -> Add Components", et sélectionnez les fichiers récemment décompressés que vous souhaitez ajouter au projet.

#### Interface utilisateur mode commande

On considère l'interface utilisateur déjà lancée, et qu'un projet est déjà ouvert.

Pour désarchiver un composant, il faut utiliser la commande *get\_list\_from\_archive* puis la commande *restore\_source*.

La commande *get\_list\_from\_archive* affiche la liste des fichiers présents dans une archive de projet :

```
bbatch 4> get_list_from_archive /tmp/blocks.arc.gz
MANIFEST [0/437530]
```

```
Printing Components in archive file
/tmp/blocks.arc.gz ...
```

```
block_occupancy.mch
block_occupancy_1.mch
```

```
block_occupancy_1_i.imp  
block_occupancy_1_it.mch  
block_occupancy_1_it_i.imp  
block_occupancy_2.mch  
block_occupancy_2_i.imp  
block_occupancy_3.mch  
block_occupancy_3_i.imp  
block_occupancy_4.mch  
block_occupancy_4_i.imp  
block_occupancy_5.mch  
block_occupancy_5_i.imp  
block_occupancy_i.imp  
block_occupancy_it.mch  
block_occupancy_it_i.imp  
configuration.mch  
inputs.mch
```

End of List

La commande *restore\_source* réalise le désarchivage. Par exemple, pour désarchiver le composant *inputs.mch*, il vous faudra taper les commandes suivantes :

## 5 Appliquer la méthode B

### 5.1 Présentation

Pour développer des logiciels suivant la méthode B, l'Atelier B propose un ensemble de commandes permettant :

- l'analyse syntaxique et le contrôle de type des machines d'un projet,
- La génération automatique des obligations de preuve (OP),
- La démonstration automatique de ces obligations,
- La démonstration interactive des OP non démontrées automatiquement,
- Le contrôle du langage d'implantation,
- La traduction du dernier niveau de raffinement des spécifications vers un langage informatique courant (C via le traducteur livré en standard ComenC).

La présentation de ces commandes suppose que le lecteur soit initié à la méthode B. Ce manuel ne traite donc que des conditions de mise en oeuvre des fonctions listées ci-avant, et non pas de leur but vis-à-vis de la méthode.

Dans la version de base de l'atelier, la traduction vers des langages informatiques courant n'est pas comprise, à l'exception du C. Des traducteurs optionnels peuvent toutefois être installés séparément.

## 5.2 Gestion des espaces de travail

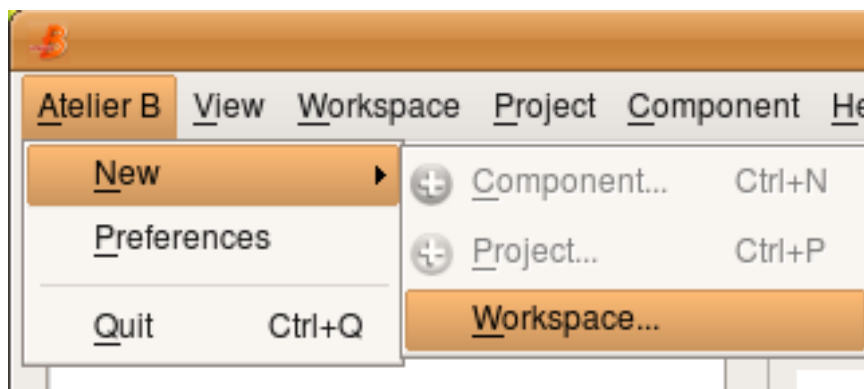
Notez que cette section n'a de sens que dans le cas de l'utilisation de l'interface graphique.

L'utilisation de l'interface en ligne de commande nécessite une configuration préalable. C'est pourquoi la notion d'espace de travail (ou workspace) existe dans l'interface graphique. Cela correspond donc à une configuration spécifique de l'interface en ligne de commande.

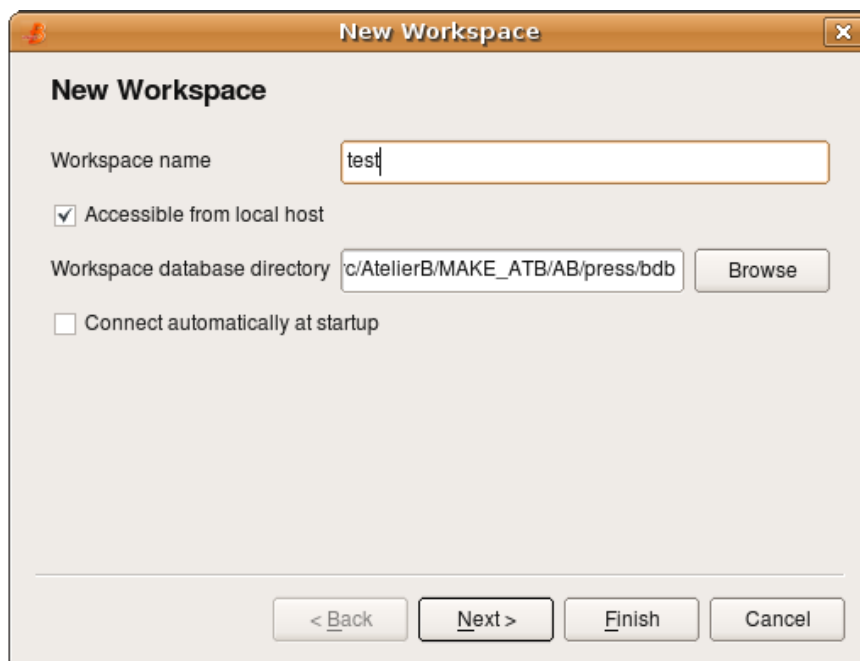
Lors de l'installation de l'Atelier B, un espace de travail par défaut est créé et vous permet de commencer à développer. Toutefois, il est possible de personnaliser son installation et sa configuration en créant des espaces séparés.

### 5.2.1 Création d'un espace de travail

La création d'un espace de travail depuis l'interface graphique se fait via le menu suivant :

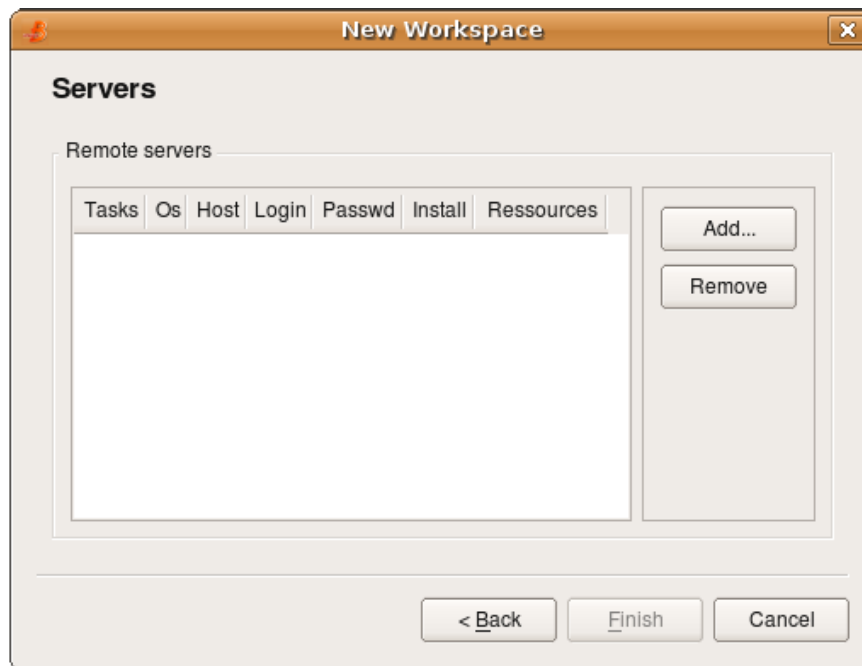


Ce menu aura pour conséquence d'appeler un assistant de création d'espace de travail, vous demandant dans un premier temps de rentrer un nom, et un répertoire de travail :



N’oubliez pas de cocher les options “Accessible from local host”, si l’espace de travail est locale (i.e. si le répertoire contenant l’espace de travail est disponible localement sur la machine), et si vous souhaitez ouvrir automatiquement cet espace au lancement de l’application (option “Connect automatically at startup”).

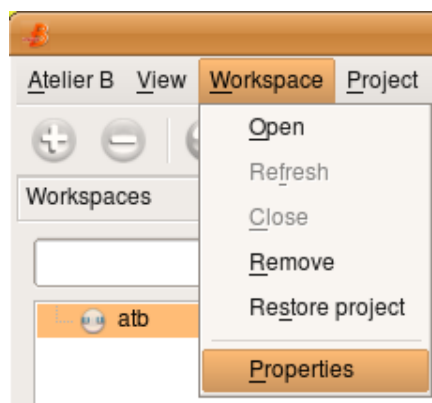
Si vous travaillez sous systèmes compatibles UNIX, L’Atelier B intègre des possibilités de parallélisation des tâches sur des machines distantes, via l’utilisation de SSH notamment. Il vous est donc demandé sur la deuxième page de l’assistant, si vous souhaitez spécifier des machines distantes. Cette étape est facultative si l’espace de travail que vous êtes en train de créer est local, mais il est obligatoire de spécifier au moins un serveur dans le cas d’un espace dit distant.



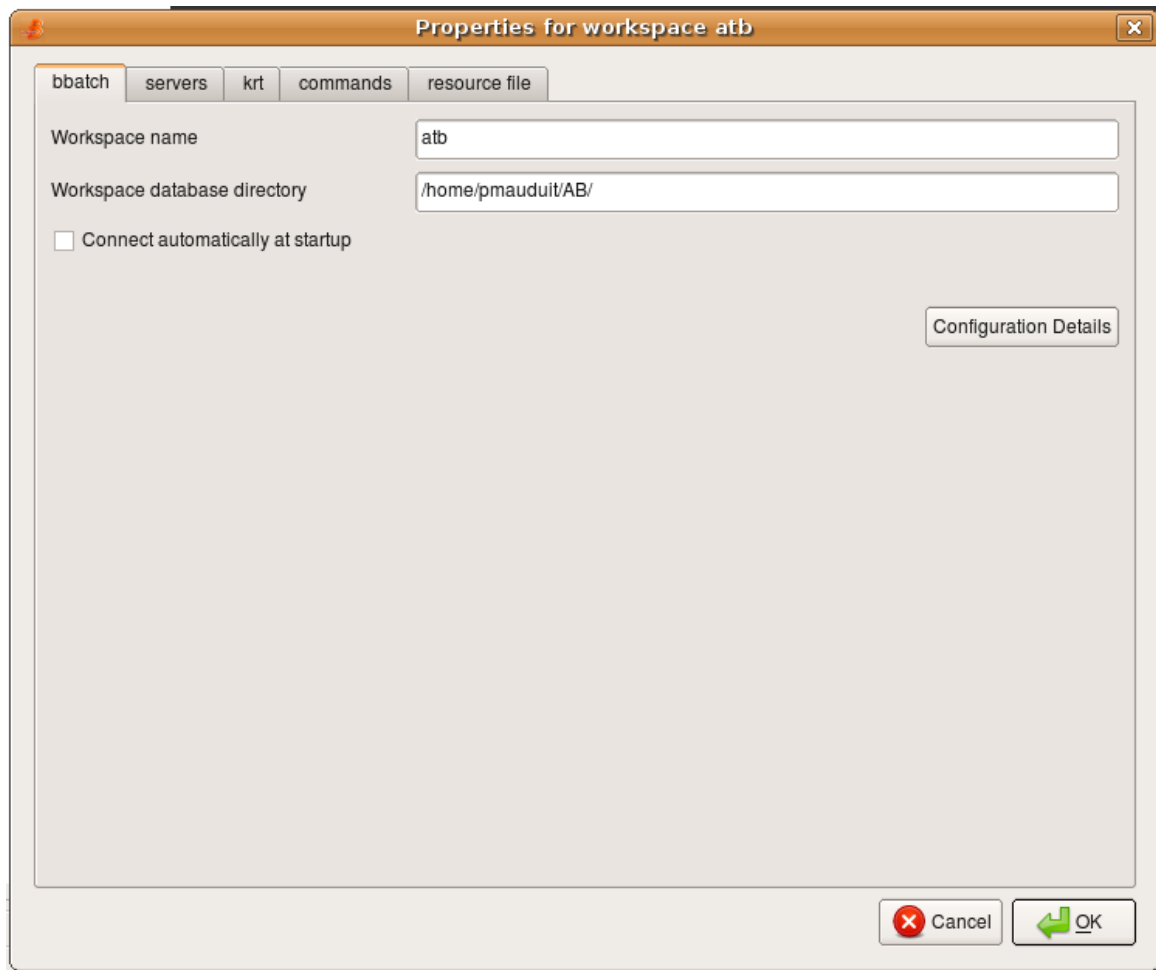
Une fois l'assistant terminé, vous êtes maintenant en mesure de gérer des projets B.

### 5.2.2 Modification des espaces de travail

Il peut être intéressant pour l'utilisateur de modifier les propriétés de ses espaces de travail. Cela est possible via le menu "Workspace -> Properties" (ou via click droit sur l'espace de travail désiré).



Vous aurez alors accès à l'écran de configuration suivant :



Ici, nous retrouvons les propriétés déjà rencontrées lors de la création, excepté ce qui concerne l’aspect distant ou local. Le bouton “Configuration Details” vous donnera des informations sur la version des différents outils.

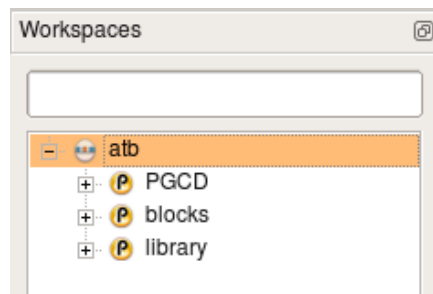
- L’onglet “servers” permet de spécifier des serveurs distants, permettant par la suite d’exécuter en parallèle différentes tâches sur les projets B,
- L’onglet “krt” contient des réglages concernant le noyau de l’Atelier B (l’exécutable krt),
- L’onglet “Commands” permet de configurer certains outils : le “Typechecker”, xref, le générateur d’obligations de preuve, le prouveur, le prouveur de prédicats, le prouveur de lemmes, et le “composant Delta”,
- Enfin, l’onglet “Resource file” permet de personnaliser son fichier de ressources, et est ainsi réservé aux utilisateurs avertis. Veuillez vous référer au paragraphe 8 relatifs aux modifications des paramètres de l’Atelier B pour plus d’informations.

### 5.2.3 Suppression d'un espace de travail

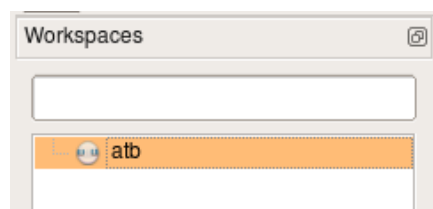
La suppression d'un espace de travail s'effectue simplement via le menu "Workspace -> Remove". Cela ne supprime toutefois aucun fichier relatif à vos projets de votre disque dur.

### 5.2.4 Ouverture et fermeture d'un espace de travail

Un double-click sur l'espace de travail, ou un passage par le menu "Workspace -> open" permet d'ouvrir ce dernier ; l'icône change alors, pour notifier du changement, et la liste des projets sous-jacents apparaît :



Pour fermer un espace de travail, utilisez à l'inverse, le menu "Workspace -> close". La liaison entre l'interface graphique et l'interface en ligne de commande est alors fermée, ce qui se traduit visuellement par un changement d'icône.





### 5.3 Analyse syntaxique et contrôle de type

#### Description

Cette fonction regroupe l'analyse syntaxique et le contrôle de type d'expressions B. L'analyse syntaxique permet de vérifier que les sources des machines sélectionnées respectent les règles de construction du langage B. A ce sujet, le lecteur pourra se reporter au Manuel de référence du langage B.

Le contrôle de type détecte :

- Les conflits d'identificateurs,
- Les erreurs de type dans l'utilisation des constructions,
- L'absence de déclarations,
- La mauvaise utilisation des opérateurs,
- Les violations de règles de visibilité,
- etc.

Ce contrôle est nécessaire pour permettre la production des obligations de preuve. Le contrôle de type appliqué à un composant s'applique automatiquement à tous les composants "requis" par le composant courant, au travers des liens SEES, USES, INCLUDES, IMPORTS, EXTENDS et REFINES.

Ce contrôle de type sur les composants requis ne s'applique que si nécessaire, c'est-à-dire si une modification du composant a été effectuée depuis le dernier contrôle de type. Les modifications de "forme" (commentaires, espaces, ...) ne sont pas prises en compte.

Les erreurs de syntaxe sont affichées dans une fenêtre d'erreur et dans la fenêtre de lancement.

Elles sont affichées de la manière suivante :

```
<fichier analysé>:<numéro de ligne>:<numéro de colonne>:<description de l'erreur>
```

Exemple:

```
AA.mch:6:17 Sequential (';') substitution is not allowed in a specification
```

Les numéros de ligne et de colonne permettent un positionnement précis sur l'endroit où l'erreur est détectée.

Les erreurs sémantiques sont affichées dans la fenêtre de lancement de l'Atelier B. Elles sont affichées de la façon suivante :

Type Checking machine AA

```
    Loading referenced machines
    Checking names clashes
    Checking CONSTRAINTS clause
[...]
```

```
    Checking INVARIANT clause
Error: 1+2 in ( aa: 1+2 ) should be a set
Error: Variable aa has not been typed
[...]
```

```
    Checking operation b_demander_code
    Checking operation b_code_saisi
    No information saved for AA
```

End of Type checking

Le contrôleur de type affiche un message d'information pour chaque clause traitée.

Le document *Contrôleur de types - Manuel des messages d'erreur* décrit en détail tous les messages d'erreurs produits lors de cette phase.

### Interface graphique utilisateur

L'interface est déjà lancée, un projet est ouvert.

Afin d'effectuer l'analyse syntaxique et le contrôle de types sur des composants, il faut effectuer les opérations suivantes :

1. Sélectionnez des composants dans la liste des composants,
2. Cliquez sur le bouton *TypeCheck*. Chaque composant est contrôlé l'un après l'autre.  
S'il y a une erreur syntaxique, celle-ci apparaît dans la vue des erreurs. Cette vue contient les descriptions des erreurs détectées.  
S'il y a une erreur sémantique sur l'un des composants, un avertissement apparaît.  
Dans ce cas vous devez regarder la nature de l'erreur dans les messages affichés dans la fenêtre de lancement.

**Remarque :** Vous pouvez interrompre le traitement en utilisant la fonctionnalité décrite au paragraphe 5.15.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet.

Pour effectuer l'analyse syntaxique et le contrôle de type sur un composant *nom\_comp*, il faut taper la commande suivante :

```

typecheck nom_comp
ou
t nom_comp

```

Les messages d'information et d'erreur du contrôleur de type sont affichés dans la fenêtre de lancement.

### Paramètres utilisables

ATB*BCOMP*Allow _ANY
FALSE
Autorisation ou non la substitution ANY en implantation.
ATB*BCOMP*Allow _Becomes _Member _Of
FALSE
Autorisation ou non la substitution “devient un élément de” en implantation.
ATB*BCOMP*Allow _Becomes _Such _That
FALSE
Autorisation ou non la substitution “devient tel que” en implantation.
ATB*BCOMP*Allow _CHOICE
FALSE
Autorisation ou non la substitution CHOICE en implantation.
ATB*BCOMP*Allow _LET
FALSE
Autorisation ou non la substitution LET en implantation.
ATB*BCOMP*Allow _Parallel
FALSE
Autorisation ou non la substitution parallèle en implantation.
ATB*BCOMP*Allow _PRE
FALSE
Autorisation ou non la substitution PRE en implantation.
ATB*BCOMP*Allow _Read _In _Values
FALSE
Autoriser ou non la lecture des variables dans la clause VALUES.
ATB*BCOMP*Allow _SELECT
FALSE
Autorisation ou non la substitution SELECT en implantation.
ATB*BCOMP*Allow _Tab _Width
8
Nombre de caractères pour former une tabulation.

## 5.4 Raffinement automatique

### Description

Une fonctionnalité de l'Atelier B permet de générer des raffinements de façon automatique, à l'aide de l'outil **BART**.

### Interface utilisateur graphique

On considère l'interface lancée, et un projet ouvert.

Afin d'utiliser la fonctionnalité de raffinement automatique, cliquez sur le composant sur lequel vous souhaitez appliquer cette fonction, puis sélectionnez dans le menu "*Component -> Automatic Refinement*".

Les composants nouvellement créés sont alors automatiquement ajoutés au projet, et un commentaire est écrit en début de fichier :

```
/* <nom_comp>
 * This file has been generated by BART (B Automatic Refinement Tool)
 * Generated the ION10/2008
 * DO NOT EDIT
 */
```

### Interface utilisateur mode commande

On suppose l'interface lancée, et un projet ouvert.

Pour utiliser la fonctionnalité de raffinement automatique sur le composant **nom\_comp**, il faut taper la commande suivante :

```
bart <nom_comp>
```

## 5.5 Génération des obligations de preuve

### Description

Cette fonction produit les obligations de preuve associées à un composant préalablement contrôlé (voir paragraphes précédents).

Ces obligations de preuve sont fixées par la méthode B. Elles varient en fonction du stade de développement du logiciel :

- Au niveau des spécifications, il faut démontrer que le modèle mathématique retenu est cohérent.
- Aux stades suivants, il faut démontrer que les transformations (raffinements) préservent les propriétés du modèle du stade précédent.

Le document *Obligations de preuve - Manuel de référence* décrit de manière théorique les obligations de preuve.

En théorie, il y a une obligation de preuve pour l'initialisation et une pour chacune des opérations. Dans la pratique, ces obligations de preuve pouvant être des formules “grandes” et complexes, la fonction *Générer les obligations de preuve* effectue un travail de simplification pour produire des formules plus facilement démontrables. En contrepartie, le nombre de formules initialement prévu augmente.

Certaines obligations de preuve, dites triviales, sont automatiquement éliminées par l'outil.

Avant de générer les obligations de preuve d'un composant, l'Atelier B vérifie que le contrôle de type a bien été effectué sur tous les composants dont il dépend. Si ce n'est pas le cas, le contrôle de type est automatiquement effectué.

La génération des obligations de preuve crée 4 fichiers dans la BDP :

- Les obligations de preuve sont enregistrées dans le fichier *nom\_comp.po*,
- Les obligations de preuve éliminées par le générateur sont enregistrées dans le fichier *nom\_comp.opo*,
- Le fichier *nom\_comp.pmi* contient les états des obligations de preuve (démontrées / non démontrées) ainsi que les démonstrations interactives,
- Une description du composant est enregistrée dans le fichier *nom\_comp.stc*.

Si l'option *differential* est utilisée, et si les O.P. du composant ont déjà été générées au moins une fois, l'Atelier B compare le composant avec la description qui a été enregistrée dans le fichier *nom\_comp.stc*. Pour chaque opération, et pour l'initialisation, il ne regénère les O.P. que si l'une des informations intervenant dans leur construction a été modifiée. Sinon, il recopie les O.P. des anciens fichiers.

Si l'option *Full* est utilisée, l'Atelier B génère à nouveau toutes les O.P. même si elles n'ont pas été modifiées.

Une fois les nouvelles O.P. générées, et si elles avaient déjà été générées au moins une fois auparavant, elles sont automatiquement comparées aux anciennes afin de récupérer les démonstrations associées. La règle de comparaison est la suivante : une O.P. B “se déduit” d'une O.P. A si elles ont le même but et si les hypothèses de B contiennent les hypothèses de A. Si un O.P. peut se déduire d'une ou plusieurs anciennes O.P., alors elle reçoit par ordre de préférence :

- La démonstration de l'ancienne O.P. de même numéro ci celle-ci aboutissait,
- La première des anciennes démonstrations des O.P. prouvées de la même clause,
- La première des anciennes démonstrations des O.P. prouvées d'une clause différente,
- La démonstration de l'ancienne O.P. de même numéro qui n'aboutissait pas,

- la première des anciennes démonstrations des O.P. non prouvées de la même clause.

Si une nouvelle O.P. ne peut se déduire d’aucune ancienne O.P., mais si son numéro et sa clause d’appartenance existaient, alors elle reçoit la démonstration de l’ancienne O.P. de même numéro et de même clause. Ainsi l’utilisateur récupère ses démonstrations même s’il a procédé à des renommages d’identificateurs.

Grâce à ce mécanisme, l’utilisateur peut conserver les démonstrations interactives (et automatiques) si les O.P. sont identiques.

Les messages du générateur d’obligations de preuve sont affichés à l’utilisateur de la façon suivante :

```
Generating proof obligations of Implementation block_occupancy_i
```

```
ValuesLemmas :
```

```
    proof obligations:          0
```

```
InstanciatedConstraintsLemmas :
```

```
    proof obligations:          0
```

```
0 proof obligations generated
```

```
Generation complete
```

```
Normalizing...
```

```
  Initialisation : unchanged
  mask_blocks    : unchanged
  release_blocks  : unchanged
  release_tdl_alarm : unchanged
  set_tdl_alarm   : unchanged
  unmask_blocks   : unchanged
  read_ob         : unchanged
```

```
Merging...
```

```
Done
```

Pour chaque clause présente dans le composant et pour laquelle il a généré des O.P., l’Atelier B affiche le nombre d’O.P. à démontrer (`proof obligations :`) ainsi que le nombre d’O.P. éliminées (`obvious proof obligations :`).

L’Atelier B affiche un caractère “.” chaque fois qu’une nouvelle O.P. est générée.

Les clauses dont les O.P. ont été recopiées sont listées à la fin de la génération.

### Interface utilisateur graphique

On suppose l'interface déjà lancée, un projet ouvert.

Pour générer les obligations de preuve sur des composants, il faut effectuer les opérations suivantes :

1. Sélectionnez les composants dans la liste des composants,
2. Cliquez sur le bouton *Po Generate ....* Si les O.P. ont déjà été générées pour un composant, l'Atelier B demande s'il est nécessaire d'exécuter une commande en mode *full*.

En cas d'erreur sur l'un des composants, un message est ajouté dans la vue d'erreurs.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet.

Pour générer les obligations de preuve du composant `nom_comp`, avec l'option *Differential*, il faut taper la commande suivante :

```
pogenerate nom_comp 1
ou
po nom_comp 1
```

Avec l'option *full*, il faut taper la commande suivante :

```
pogenerate nom_comp 0
ou
po nom_comp 0
```

L'option par défaut proposée par l'Atelier B est le mode *Differential*.

**Remarque :** Vous pouvez interrompre le traitement en utilisant la fonctionnalité décrite au paragraphe 5.15. En cas de tentative de génération sur des composant ayant déjà subi une génération des obligations de preuve, l'interface de l'Atelier B vous demandera s'il est nécessaire de forcer l'action.

ATB*POG*Generate_Obvious_PO
Configuré à l'installation de l'Atelier B sous systèmes UNIX
Non généré par défaut sous systèmes Windows et Mac.
Générer ou non les obligations de preuves triviales.

## 5.6 Affichage des obligations de preuve

### Description

L'Atelier B fournit deux méthodes pour afficher les obligations de preuve :

- Utilisation du PO Viewer
- Utilisation du prouveur interactif

L'utilisation du prouveur interactif est recommandée s'il y a :

- Des hypothèses complexes : dans ce cas les fonctionnalités de recherche du prouveur interactif sont indispensables pour l'analyse de ces obligations de preuve.
- De nombreuses obligations de preuve par opération : dans ce cas, il faut les afficher une par une, ce qui est impossible avec le POViewer.

Le document *Prouveur interactif - Manuel Utilisateur* décrit les différentes méthodes de visualisation des obligations de preuve.

Ce paragraphe décrit l'utilisation du *POViewer*.

Cet outil permet :

- L'affichage des obligations de preuve d'un composant clause par clause,
- L'affichage des obligations de preuve triviales, c'est à dire celles qui ont été éliminées par le Générateur d'Obligations de Preuve.
- L'affichage et l'impression des obligations de preuve en utilisant des polices mathématiques par l'intermédiaire d'un traitement de texte(L<sup>A</sup>T<sub>E</sub>X ou RTF).

Les obligations de preuve contiennent des commentaires qui indiquent :

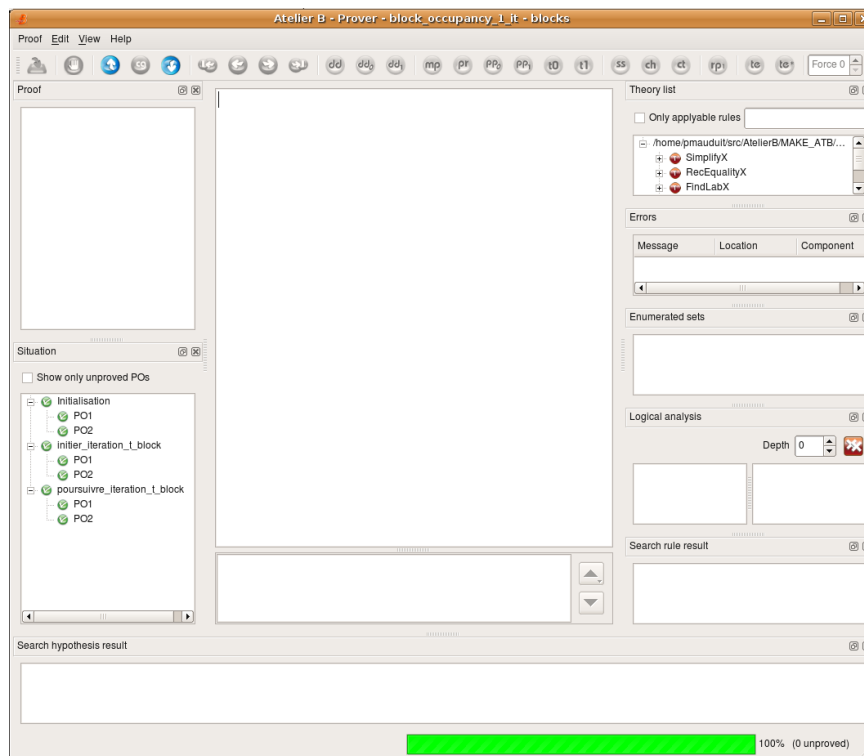
- L'origine des hypothèses (par exemple : **Component invariant**),
- La justification théorique de cette obligation de preuve.  
Dans ce cas le commentaire fait référence à un chapitre du document *Obligations de preuve - Manuel de référence*.

### Interface utilisateur graphique

On suppose l'interface lancée, un projet ouvert. Pour ouvrir l'interface du prouveur interactif, il faut cliquer sur le composant concerné, et sélectionner dans le menu "*Component -> Proof -> Interactive Proof ...*".

La fenêtre suivante s'affiche alors :





Les obligations de preuve sont alors visibles dans la vue *"Situation"*.

### Interface utilisateur mode commande

#### Paramètre optionels

ATB*OPT_TOOLS_<SYSTEM>*Latex_Binary_Directory
Configuré à l'installation de l'Atelier B ou bien directement dans l'interface (cf. Paragraphe 2.3.5)
Répertoire où trouver le binaire $\text{\LaTeX}$
ATB*OPT_TOOLS_<SYSTEM>*Latex_Viewer
Configuré à l'installation de l'Atelier B
Nom du visualisateur $\text{\LaTeX}$

## 5.7 Démonstration automatique

### Description

Cette fonction automatise, dans la mesure de ses capacités, la démonstration des obligations de preuve associées à chaque composant B.

L'activité de preuve est essentielle dans la méthode B, c'est pour cette raison que deux manuels lui sont consacrés :

- *Prouveur interactif - Manuel utilisateur*
- *Prouveur interactif - Manuel de référence*

Le prouveur automatique de l'Atelier B dispose de plusieurs forces. Ces forces sont décrites dans le document *prouveur interactif - manuel utilisateur*.

Les messages du prouveur automatique sont affichés, dans la fenêtre de lancement, de la façon suivante :

Proving block\_occupancy\_1\_i

Proof pass 0, still 33 unproved P0

```

clause mask_blocks_1
  ++
clause release_blocks_2
  +++
clause release_blocks_6
  +++
clause release_tdl_alarm_1
  +
clause release_tdl_alarm_2
  ++++++++
clause set_tdl_alarm_1
  +
clause unmask_blocks_1
  ++

```

End of Proof

ValuesLemmas	Proved 0	Unproved 0	
Instanciati	Constraints	Lemmas	
Initialisation	Proved 0	Unproved 0	Unproved 0
mask_blocks_1	Proved 2	Unproved 0	
release_blocks_1	Proved 0	Unproved 0	
release_blocks_2	Proved 3	Unproved 4	
release_blocks_6	Proved 3	Unproved 4	
release_tdl_alarm_1	Proved 1	Unproved 0	
release_tdl_alarm_2	Proved 9	Unproved 3	
set_tdl_alarm_1	Proved 1	Unproved 1	
unmask_blocks_1	Proved 2	Unproved 0	
read_ob	Proved 0	Unproved 0	
release_blocks_3	Proved 0	Unproved 0	
release_blocks_4	Proved 0	Unproved 0	
release_blocks_5	Proved 0	Unproved 0	

```

release_blocks_7      Proved 0      Unproved 0
TOTAL for block_occupancy_1_i  Proved 21      Unproved 12

```

Pour chaque clause présente dans le composant, l'Atelier B affiche un + chaque fois qu'une obligation de preuve est démontrée, et, un - chaque fois que le prouveur échoue sur une obligation de preuve.

### Interface utilisateur graphique

L'interface utilisateur est déjà lancée, un projet est déjà ouvert.

Pour lancer le prouveur automatique sur des composants, il faut effectuer les opérations suivantes :

1. Sélectionnez les composants dans la liste,
2. Cliquez dans le menu "Component -> Proof", sur la force désirée.

Vous avez par ailleurs la possibilité de faire une preuve personnalisée, via le menu "Component -> Proof -> Customize Proof". La fenêtre suivante s'affiche alors :



vous pouvez ajouter une séquence programmée de différentes forces à appliquer à votre sélection de composant, et ainsi de personnaliser entièrement votre travail de preuve.

Vous pouvez en outre interrompre le traitement, et définir un temps maximum d'exécution.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancé, vous avez déjà ouvert un projet.

Pour lancer le prouveur automatique sur le composant `nom_comp`, il faut taper la commande suivante :

```

prove nom_comp <force>
ou
pr nom_comp <force>

```

où `<force>` peut prendre l'une des valeurs suivantes :

**0,1,2,3** pour les différentes forces du prouveur,

- 1 pour la force Rapide,
- 2 pour l'option "Replay",
- 3 pour l'option "User Pass",

## 5.8 Démonstration interactive

### Description

Le rôle essentiel de cette fonction est de permettre à l'utilisateur de démontrer manuellement les obligations de preuve qui n'ont pu l'être automatiquement.

### Interface utilisateur graphique

L'interface utilisateur est lancée, un projet est ouvert.

Pour lancer le prouveur interactif, il faut effectuer les opérations suivantes :

1. cliquez sur le composant dans la liste des composants.
2. Cliquez sur le menu "Component -> Proof -> Interactive Proof", ou utilisez le raccourci "Ctrl+I".

Depuis l'interface du prouveur (c.f. paragraphe 5.8), il vous est possible d'effectuer un travail de preuve manuel. L'avancement est indiqué par l'intermédiaire d'une barre de progression, en bas à droite. Par ailleurs, l'état de la preuve est indiqué dans la vue "*situation*". Reportez-vous au manuel du prouveur interactif pour de plus amples informations (menu "*Help -> Help contents*").

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, un projet est ouvert.

Pour lancer le prouveur interactif, il faut taper la commande suivante :

```
browse nom_comp  
ou  
b nom_comp
```

après avoir entré cette commande, l'invite du prouveur interactif apparaît : **PRI>**.

Vous pouvez ensuite taper les différentes commandes du prouveur interactif.

Tapez **qu** pour sortir du prouveur.

## 5.9 Annulation des démonstrations

### Description

Cette fonction est utilisée pour annuler les démonstrations effectuées sur un composant et permettre le rejeu des preuves interactives.

Les démonstrations interactives ne sont pas perdues. Seul l'état des obligations de preuve est modifié.

### Interface utilisateur graphique

L'interface est lancée, et un projet est ouvert.

Pour annuler les démonstrations des composants, il faut suivre les étapes suivantes :

1. Sélectionnez les composants dans la liste des composants
2. Utilisez le menu "*Component -> Unprove*"

Les messages de retour dans la vue des tâches doit indiquer alors :

Unproving successful

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet.

Pour annuler les démonstrations du composant `nom_comp`, il suffit de taper la commande suivante :

```
unprove nom_comp  
ou  
u nom_comp
```

Le message suivant est affiché dans la fenêtre de lancement en cas de succès :

Unproving successful

## 5.10 Contrôle du langage d'implantation

### Description

Avant d'utiliser les traducteurs vers les langages informatiques courants, il est nécessaire de vérifier que le langage utilisé dans les implantations est bien traduisible. Les constructions autorisées dans les implémentations sont décrites dans le document *Langage B - Manuel de référence*.

Les messages d'erreur du contrôleur de B0 sont affichés dans la vue d'erreur de la manière suivante :

<fichier analysé>:<numéro ligne>:<numéro colonne> (B0Check) <description erreur>

Par exemple :

```
B0 Checking B_Clavier_code_1
B_Clavier_code_1.imp:5:11 (B0Check) binary expression is not a simple term
B_Clavier_code_1.imp:5:19 (B0Check) binary expression is not a simple term
B0 Check error in B_Clavier_code_1
```

Les numéros de ligne et de colonne permettent un positionnement précis sur l'endroit où l'erreur est détectée.

Lorsque le composant est correct, le message suivant est affiché dans la fenêtre de lancement :

```
B0 Checking B_clavier_code_1
B0 Checking B_clavier_code_1 successful
```

**Remarque :** Pour utiliser le traducteur HIA, il est nécessaire de typer les tableaux avec des constantes concrètes. Or par défaut, le contrôleur de B0 signale que la constante concrète n'est pas implémentable. Pour passer le contrôleur de B0 dans ce cas là, il faut donc positionner la ressource suivante :

```
ATB*BCOMP*Enable_Typing_Identifiers: TRUE
```

Le plus simple est de mettre cette ressource dans le fichier de ressources projets des projets qui seront traduits en HIA.

### Interface utilisateur graphique

L'interface utilisateur est déjà lancée, un projet est déjà ouvert. Pour effectuer le contrôle du langage d'implantation sur des composants, il suffit d'effectuer les opérations suivantes :

1. Sélectionnez les composants dans la liste des composants.
2. Utilisez le menu *"Component -> B0 Check"*, ou directement le raccourci *Ctrl+B*.

S'il y a une erreur sur un des composants, cette dernière est affichée dans la vue d'erreurs. Un double-click sur l'erreur ouvrira directement l'éditeur interne (si configuré comme éditeur par défaut) à l'emplacement provoquant l'erreur.

### Interface utilisateur mode commande

On suppose l'interface déjà lancée, et un projet déjà ouvert.

Pour contrôler le composant `nom_comp`, il faut taper la commande suivante :

b0check nom\_comp  
ou  
b0c nom\_comp

### Paramètres utilisables

<i>ATB*BCOMP*Disable_Array_Compatibility_Check</i>
FALSE
Effectuer ou non les contrôles de compatibilité des ensembles indices des tableaux
<i>ATB*BCOMP*Disable_Concrete_Constants_Type_Check</i>
FALSE
Effectuer ou non les contrôles de type des constantes concrètes
<i>ATB*BCOMP*Disable_Expression_Syntax_Check</i>
FALSE
Effectuer ou non les contrôles syntaxiques des expressions
<i>ATB*BCOMP*Disable_Formal_Params_Type_Check</i>
FALSE
Effectuer ou non les contrôles de type des paramètres formels
<i>ATB*BCOMP*Disable_Variables_Initialisation_Checker</i>
FALSE
Contrôler ou non l'initialisation des variables
<i>ATB*BCOMP*Disable_Locale_Variables_Type_Check</i>
FALSE
Effectuer ou non les contrôles de type des variables
<i>ATB*BCOMP*Disable_Operation_Input_Parameters_Type_Check</i>
FALSE
Effectuer ou non les contrôles de type des paramètres d'entrée d'opération
<i>ATB*BCOMP*Disable_Operation_Output_Parameters_Type_Check</i>
FALSE
Effectuer ou non les contrôles de type des paramètres de sortie d'opération
<i>ATB*BCOMP*Disable_Parameter_Instanceiation_Check</i>
FALSE
Effectuer ou non le contrôle des instanciations des paramètres de machines
<i>ATB*BCOMP*Disable_Predicate_Syntax_Check</i>
FALSE
Effectuer ou non les contrôles syntaxiques des prédicats

<i>ATB*BCOMP*Disable_Valuation_Check</i>
FALSE
Effectuer ou non les contrôles de la clause VALUES

<i>ATB*BCOMP*Enable_Typing_Identifiers</i>
FALSE
Typier les tableaux avec des constantes concrètes

## 5.11 Contrôle global d'un projet

### Description

Cette fonction, disponible uniquement dans l'interface en mode commande, réalise des contrôles sur l'ensemble des composants d'un projet. Les règles vérifiées par cette fonction sont :

- Une machine ne peut être importée qu'une seule fois dans un projet,
- Une machine vue doit être importée par un composant du projet,
- la clause SEES doit être transversale à un composant,
- le graphe de dépendance ne doit pas contenir de cycle,
- Les noms des composants d'un projet doivent être différents (une différence sur la casse ne suffit pas).

Ces contrôles sont décrits dans le document *le langage B - Manuel de référence*.

Ces contrôles sont nécessaires pour la traduction du projet, ils sont lancés automatiquement par l'Atelier B avant la traduction du projet.

Certains de ces contrôles sont réalisés automatiquement avant l'analyse syntaxique et le contrôle de type des composants, afin de prévenir au plus tôt l'utilisateur.

L'utilisateur peut aussi effectuer ces contrôles quand il le désire en suivant les procédures décrites ci-dessous.

### Interface utilisateur mode commande

L'interface utilisateur est supposée lancée, un projet est ouvert.

Pour effectuer les contrôles sur ce projet, il faut taper la commande suivante :

```
project_check nom_comp
ou
pchk nom_comp
```



Le paramètre de cette commande est le nom de l'implémentation point d'entrée du projet.

## 5.12 Traduction

### Description

Dans la version de base de l'Atelier B, seul le traducteur C **ComenC** est livré.

Cette fonction permet la traduction en langage cible des implémentations du projet.

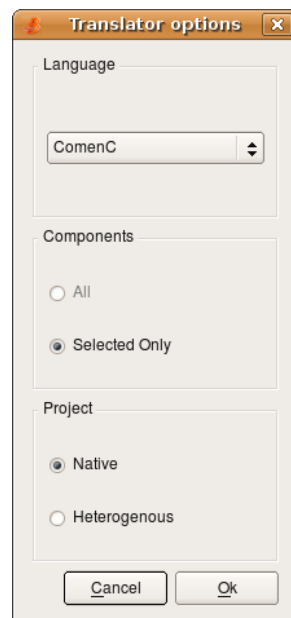
Pour plus d'informations, reportez-vous au manuel utilisateur du traducteur que vous souhaitez utiliser.

### Interface graphique utilisateur

Pour traduire un composant en C via le traducteur **ComenC**, il faut effectuer les opérations suivantes :

1. Sélectionnez le composant à traduire dans la liste,
2. Sélectionnez le menu *"Component -> Generate Code"*.

La fenêtre suivante s'affiche alors :



Après sélection des options désirées, cliquez sur *"Ok"* pour lancer le processus de traduction.

### Interface utilisateur mode commande

Pour traduire une implémentation `nom_imp` en C en utilisant le traducteur **ComenC**, il faut utiliser la commande suivante :

```
ComenCtrans nom_imp  
ou  
b2c nom_imp
```

## 5.13 Appliquer un outil à l'ensemble des composants d'un projet

### Description

La fonction *Make project* permet de réaliser, sur l'ensemble des composants d'un projet les opérations suivantes :

- L'analyse syntaxique et le contrôle de types,
- La génération des obligations de preuve,
- La preuve,
- La vérification du langage d'implantation.

Cette fonction tient compte des liens entre les composants du projet ; son fonctionnement est similaire à celui de la fonction "make" d'UNIX.

Cette fonction propose 2 options :

**Mode forcé :** Les opérations demandées seront réalisées, quel que soit l'état des composants. Par exemple, si un composant est déjà dans l'état *TypeChecked* et que l'utilisateur demande un make forcé du projet, alors le contrôle de type sera refait sur ce composant.

**Mode normal :** Les opérations demandées ne sont réalisées que si cela est nécessaire.

**Attention :** Lorsqu'on demande un Make forcé sur le projet, seule l'opération demandée sera refaite systématiquement sur tous les composants du projet.

Par exemple, si vous demandez un make forcé pour l'opération *POgenerate*, alors la génération des obligations de preuve sera refaite sur tous les composants du projet, le contrôle de type ne sera pas refait.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet.

Pour effectuer une opération sur tous les composants du projet, il faut taper la commande suivante :

```
make_all operation force  
ou  
m operation force
```

Le paramètre `operation` peut prendre l'une des valeurs suivantes : `typecheck`, `pogenerate`, `b0check`, `prove`

Le paramètre `force` doit être égal à 0 en mode normal, et égal à 1 en mode forcé.

Si l'opération demandée est `prove`, vous pouvez donner un troisième paramètre qui est la force du prouveur à appliquer. (Se référer au paragraphe concernant la preuve pour plus d'informations).

## 5.14 Mise à jour d'un projet

### Description

Lorsque plusieurs utilisateurs travaillent simultanément sur un projet de taille importante, la modification d'un composant peut avoir des conséquences sur beaucoup d'autres composants du projet (tous les composants liés à ce composant).

La fonction *Remake project* permet la mise à jour de l'ensemble des composants d'un projet.

Elle tient compte des dépendances entre les composants du projet et de l'état de chaque composant.

Cette fonction "refait" pour chaque composant toutes les actions qui ont déjà été effectuées au moins une fois.

Cette fonction propose 2 options :

**Mode forcé :** Les opérations qui ont été déjà réalisées au moins une fois seront effectuées à nouveau, quel que soit l'état des composants. Par exemple, si un composant est déjà dans l'état *TypeChecked* et que l'utilisateur demande un *Remake forcé* du projet, alors le contrôle de type sera refait sur ce composant.

**Mode normal :** Les opérations qui ont été réalisées au moins une fois par le passé, ne sont effectuées que si cela est nécessaire.

### Interface utilisateur mode commande

L'interface est déjà lancée, un projet est déjà ouvert.

Pour effectuer la mise à jour de tous les composants du projet, il faut taper la commande suivante :

```
remake force  
ou  
r force
```

Le paramètre `force` doit être égal à 0 en mode normal, et égal à 1 en mode forcé.

## 5.15 Interruption des outils

### Description

Cette fonctionnalité permet l'interruption des actions suivantes :

- Analyse syntaxique et contrôle de type,
- Génération des obligations de preuve,
- Démonstration automatique.

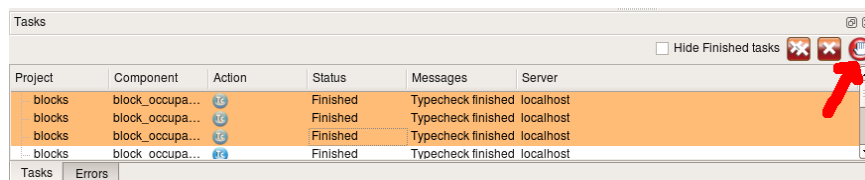
Elle offre différentes possibilités en fonction de l'action qui est en train de s'exécuter.

### Interface utilisateur graphique

On suppose l'interface déjà lancée, et un projet déjà ouvert.

Après avoir sélectionné un ou plusieurs composants, cliquez sur le bouton correspondant à l'action de votre choix : *TypeCheck*, *PO Generate*, *Proof*, *Make Project*, *Remake project*, ....

Quand l'action commence à s'exécuter, cette dernière apparaît dans la vue des tâches. Il est alors possible d'interrompre les tâches sélectionnées en cliquant sur l'icône d'interruption présentée ci-après :



## 5.16 Gestion des dépendances

### Description

Pour chacune des actions décrites dans cette section, l'Atelier B vérifie que cette action n'a pas déjà été effectuée sur le composant.

L'action n'est exécutée que si le composant, ou les composants dont il dépend, ont été modifiés.

Si ce n'est pas le cas les messages suivants sont affichés :

`<comp_name> has already been type checked`

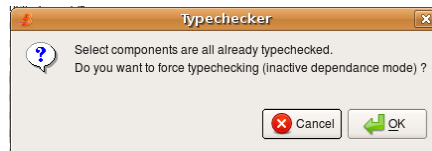
ou

`Proof obligations for <comp_name> have already been generated`

Les fonctions décrites ci-dessous vous permettent de "forcer" l'exécution de l'action.

### Interface utilisateur graphique

Lorsque des actions semblent être effectuées inutilement, l'interface le détecte et propose alors l'utilisateur de forcer malgré tout ou non l'action. Par exemple, la question suivante peut vous être posée suite à la demande d'un *Typecheck* :



### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée.

Pour arrêter la gestion des dépendances et donc pouvoir forcer une action, il faut taper la commande suivante :

```
disable_dependence_mode  
ou  
ddm
```

vous pouvez ensuite taper votre commande, par exemple : **typecheck AA**.

pour remettre en fonctionnement la gestion des dépendances, il faut taper la commande suivante :

```
enable_dependence_mode  
ou  
edm
```

## 6 Analyse des développements B

### 6.1 Présentation

*"Analyser un développement B"* est une fonction regroupant plusieurs commandes qui permettent d'obtenir des informations sur les composants d'un projet ou, plus finement, sur les constituants des composants d'un projet (opérations, variables ...).

Il est ainsi possible :

- De connaître l'état des composants d'un projet (contrôlés syntaxiquement, démontrés ...),
- De connaître l'état de la preuve d'un composant du projet (nombre d'obligations de preuve par opérations, ...),
- D'établir un graphe de dépendance entre les composants d'un projet,

## 6.2 État d'un projet

### Description

Cette fonction permet la création d'un tableau récapitulatif donnant des informations sur tous les composants du projet.

Elle est utilisée pour voir l'avancement du projet.

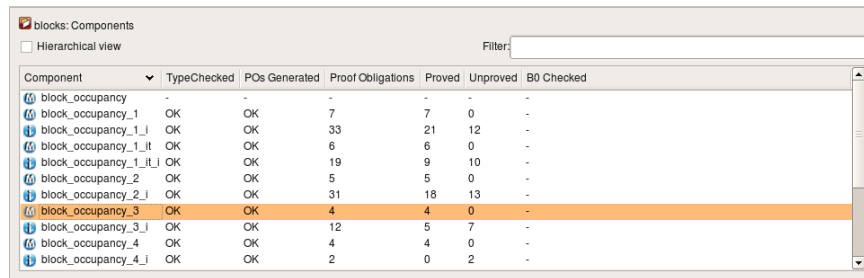
Cette fonction tient compte des dépendances entre les composants du projet. Si des composants ne sont pas encore présents dans le projet, un message d'avertissement est affiché dans la fenêtre de lancement de l'Atelier B. Ce tableau peut également être utilisé par les outils de documentation de l'Atelier B.

Les différentes colonnes, dont la présence peut varier selon que vous utilisez l'interface en ligne de commande ou l'interface graphique sont les suivantes :

- TC (ou Typechecked) : indique si l'analyse syntaxique et le contrôle de type ont été effectués avec succès ou non sur le composant, et sont à jour par rapport aux composants dont il "dépend".
- GOP (ou PO Generated) : La valeur OK signifie que les obligations de preuve ont été générées sur le composant.
- nPo (ou "Proof obligations") : Cette colonne indique le nombre d'obligations de preuve non triviales du composant.
- nUn (ou "Unproved") : Cette colonne contient le nombre d'obligations de preuve non encore démontrées du composant.
- %Pr : Cette colonne contient le pourcentage d'obligations de preuve déjà démontrées. Ce pourcentage ne tient pas compte des OP. éliminées par le générateur d'obligations de preuve.
- B0C (ou "B0 checked") : La valeur OK signifie que le contrôle du langage d'implantation a été effectué avec succès sur ce composant.

### Interface utilisateur graphique

Une fois le projet ouvert, l'état global est directement disponible sur la vue des composants. Un exemple est donné sur la capture d'écran suivante :



Component	TypeChecked	POs Generated	Proof Obligations	Proved	Unproved	B0 Checked
block_occupancy	-	-	-	-	-	-
block_occupancy_1	OK	OK	7	7	0	-
block_occupancy_1_l	OK	OK	33	21	12	-
block_occupancy_1_it	OK	OK	6	6	0	-
block_occupancy_1_it_l	OK	OK	19	9	10	-
block_occupancy_2	OK	OK	5	5	0	-
block_occupancy_2_l	OK	OK	31	18	13	-
block_occupancy_3	OK	OK	4	4	0	-
block_occupancy_3_l	OK	OK	12	5	7	-
block_occupancy_4	OK	OK	4	4	0	-
block_occupancy_4_l	OK	OK	2	0	2	-

On remarque que le nombre de colonnes constituant le tableau peut varier d'une interface à l'autre, et suivant la configuration.

### Interface utilisateur mode commande

On suppose l'interface déjà lancée et un projet déjà ouvert. Afin d'obtenir l'état du projet, il est nécessaire de taper la commande suivante :

```
status_global
```

```
ou
```

```
sg
```

## 6.3 État d'un composant

### Description

Un composant pour lequel la méthode B est appliquée traverse un certain nombre d'états :

- Modified : Après modification du source du composant,
- Parsed : Après analyse syntaxique du composant,
- TypeChecked : Après contrôle de type du composant,
- POGenerated : Après génération des obligations de preuve du composant,
- AutoProved : Après démonstration automatique ou interactive de toutes les obligations de preuve du composant.

Ces états sont exclusifs. Un composant perd son état Modified dès qu'un contrôle de type a été effectué.

Selon l'état actuel du composant, la commande permettant d'obtenir les informations sur le statut seront différentes. À titre d'exemple, si le composant QUERY a passé le contrôle de type, les informations retournées seront de la forme suivante :

Printing the status of QUERY



```
QUERY TypeChecked /path/to/QUERY.mch
```

End of Printing the status

Toutefois, si le composant est dans l'état POGenerated, la fonction affiche un tableau donnant des informations plus précises sur les obligations de preuve du composant. Par exemple :

Printing the status of FILE\_BUFFER\_1

```
FILE_BUFFER_1 POGenerated /path/to/FILE_BUFFER_1.mch
```

	NbObv	NbPO	NbPRi	NbPRa	%Pr
Initialisation	3	0			
load_buffer	12	3	0	3	100
create_record	4	1	0	0	0
not_in_buffer	7	0			
mod_buffer	8	0			
val_buffer	8	0			
size_file	5	0			
FILE_BUFFER_1	47	4	0	3	75

End of Printing the status

Les colonnes de ce tableau indiquent, pour chaque opération du composant :

- NbObv : Cette colonne contient le nombre d'obligations de preuve "triviales" de l'opération. Il s'agit des obligations de preuve suffisamment simples pour être éliminées automatiquement par le générateur d'obligations de preuve.
- NbPO : Cette colonne contient le nombre d'obligations de preuve non triviales de l'opération.
- NbPRi : Cette colonne contient le nombre d'obligations de preuve de l'opération démontrées à l'aide du prouveur interactif.
- NbPRa : Cette colonne contient le nombre d'obligations de preuve de l'opération démontrées à l'aide du prouveur automatique.
- %Pr : Cette colonne contient le pourcentage d'obligations de preuve déjà démontrées

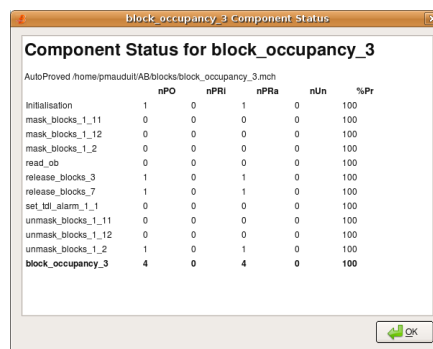
de l'opération. Ce pourcentage ne tient pas compte des O.P. éliminées par le générateur d'obligations de preuve.

La dernière ligne du tableau (TOTAL) fait la somme de chaque information sur toutes les opérations du composant.

Ces informations peuvent être incluses dans les documents générés automatiquement par l'Atelier B.

### Interface utilisateur graphique

On suppose l'interface graphique déjà lancée, un projet est ouvert. Afin d'obtenir les informations sur l'état d'un composant, il suffit de cliquer sur ce dernier et de sélectionner dans le menu "Component", le sous-menu "Status" (ou bien d'utiliser le raccourci clavier Ctrl+s). Une fenêtre s'affiche alors, comme présenté sur la capture d'écran suivante :



	nPO	nPRI	nPRa	nUn	%Pr
Initialisation	1	0	1	0	100
mask_blocks_1_11	0	0	0	0	100
mask_blocks_1_12	0	0	0	0	100
mask_blocks_1_2	0	0	0	0	100
read_ob	0	0	0	0	100
release_blocks_3	1	0	1	0	100
release_blocks_7	1	0	1	0	100
set_tdi_alarm_1_1	0	0	0	0	100
unmask_blocks_1_11	0	0	0	0	100
unmask_blocks_1_12	0	0	0	0	100
unmask_blocks_1_2	1	0	1	0	100
<b>block_occupancy_3</b>	<b>4</b>	<b>0</b>	<b>4</b>	<b>0</b>	<b>100</b>

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, un projet est déjà ouvert. Pour obtenir l'état du composant nom\_comp, il suffit de taper la commande suivante :

```
status nom_comp
ou
s nom_comp
```

## 6.4 Graphes de dépendances

### Description

Cette option n'est disponible seulement si vous avez configuré l'outil dot, du paquetage GraphViz, disponible librement sur <http://www.graphviz.org/>.

Un graphe de dépendance est fourni pour un composant sélectionné, ou bien pour l'ensemble des composants du projet.

La recherche des dépendances est récursive. Ainsi, si un composant X dépend d'un composant Y (par exemple par "IMPORTS"), lui-même dépendant d'un composant Z (par exemple par "SEES"), les composants X, Y et Z seront représentés sur le graphe.

Les composants sont regroupés par module. Un module contient une machine et ses raffinements jusqu'à l'implémentation.

Étant donné la complexité des projets, plusieurs options sont disponibles pour filtrer les liens et les modules affichés.

En ce qui concerne le filtrage des composants, l'utilisateur peut choisir entre plusieurs options pour les composants. Les options disponibles sont :

All : Tous les composants du projet sont présents.

Selected only : Seul le composant sélectionné est affiché.

Selected and transitively linked : Seuls le composant sélectionné et tous ses raffinements et abstractions sont affichés.

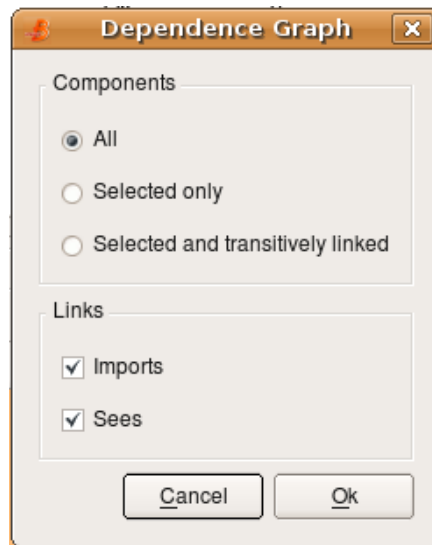
Filtrage des liens : L'utilisateur peut choisir les types de liens (SEES, IMPORTS, ...) qui sont affichés.

### Interface utilisateur graphique

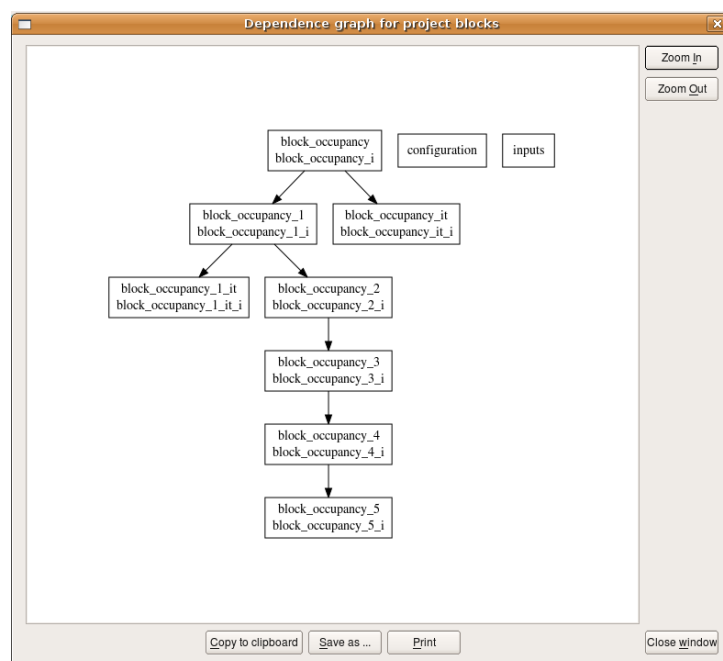
L'interface est déjà lancée, un projet est ouvert. Pour obtenir un graphe de dépendances, il faut effectuer les opérations suivantes :

1. Si vous souhaitez avoir un graphe à partir d'un composant particulier, sélectionnez-le dans la liste.
2. Cliquez sur le menu "Component -> Dependence Graph".
3. Sélectionnez les options désirées et cliquez sur le bouton "Ok".

Ci-dessous une capture d'écran avec les options personnalisables :



La fenêtre avec le graphe s'affiche alors :



### Interface utilisateur mode Commande

L'interface est lancée et un projet est ouvert. Pour obtenir un graphe de dépendances, il faut utiliser la commande suivante :

```
project_status  
ou  
ps
```

Cette commande prend sept paramètres :

1. Le nom d'un composant ou la valeur "\*" pour obtenir un graphe de tout le projet.
2. L'option sur les composants : A pour All, S pour Selected only, G pour selected and transitively linked.
3. L'option sur les bibliothèques : A pour Show All, S pour Show, G pour Group, N pour Hide.
4. Le sens du graphe : U pour ascendant, D pour descendant.
5. Les types de liens à afficher. Les liens sont dans l'ordre suivant : EXTENDS, IMPORTS, INCLUDES, SEES, USES. La valeur 0 annule l'affichage du lien. Exemple : pour n'afficher que les IMPORTS et SEES : 01010.
6. L'option sur les composants non liés : 1 pour cacher les composants isolés, 0 pour les voir.
7. 1 pour ne voir que le graphe d'instanciation, 0 sinon.

## 6.5 Graphes d'appel d'opération

### Description

Cette commande n'est disponible que si la station de travail dispose de l'outil DaVinci, et uniquement via l'interface en ligne de commande.

Le graphe d'appels d'opération permet de visualiser les cascades d'appels d'opération dans les clauses OPERATIONS, LOCAL\_OPERATIONS et INITIALISATION d'un composant B. Un tel graphe est utile lors de la phase de preuve, car il permet de comprendre d'où viennent les éléments d'une obligation de preuve liée à une opération.

Pour distinguer les spécifications et les implémentations des opérations locales, les noms des implémentations des opérations locales sont précédés de *refinement\_of\_* dans ce graphe.

Le graphe d'appels d'opération est fourni pour les opérations d'une liste de composants sélectionnés, pour ces composants et tous ceux dont ils dépendent ou pour l'ensemble des composants du projet.

Les graphes d'appels d'opération peuvent être inclus dans les documentations produites automatiquement par l'Atelier B.

### Interface utilisateur mode commande

On suppose l'interface déjà lancée, et un projet ouvert.

Pour obtenir un graphe d'appels d'opération, il faut utiliser la commande suivante :

`opcall_graph`  
 ou  
`ocg`

Cette commande prend en argument 3 paramètres :

1. Le nom d'un composant ou la valeur "\*" pour obtenir un graphe de tout le projet,
2. Le nom d'une opération, le mot clé INITIALISATION ou la valeur "\*" pour avoir les graphes de toutes les opérations,
3. 1 si on souhaite étendre le graphe aux raffinements et aux abstractions de l'opération, 0 sinon.

## 6.6 Références croisées

**Description** Cette fonction effectue des recherches sur les identificateurs définis dans les composants du projet.

Pour chaque identificateur, la fonction affiche :

- Sa nature : variable, ensemble, ...
- L'endroit où il est typé,
- Les noms des composants et des clauses où il est défini.
- Les noms des composants et des clauses où il est utilisé
- Les noms des composants et des clauses où il est modifié (pour les variables uniquement).

Les identificateurs sont présentés triés par nature. L'utilisateur peut demander ces informations :

- Pour un identificateur particulier,
- pour tous les identificateurs définis dans un composant,
- Pour tous les identificateurs définis dans le projet : il obtient alors un dictionnaire des terme utilisés dans le projet. Pour distinguer les spécifications et les implémentations des opérations locales, les noms des implémentations des opérations locales sont précédés de `refinement_of_`.

Exemple :

-----

**VARIABLES****fin\_delai**

concrete variable

Definition of "fin\_delai" in B\_Delais.mch

Use of "fin\_delai" in B\_Delais.mch (INVARIANT)

Use of "fin\_delai" in B\_Delais.mch (INVARIANT)

Modification of "fin\_delai" in B\_Delais.mch (INITIALISATION)

Modification of "fin\_delai" in B\_Delais.mch (b\_init\_delai)

Modification of "fin\_delai" in B\_Delais.mch (b\_stopper\_delai)

Modification of "fin\_delai" in B\_Delais.mch (b\_delai\_ecoule)

.....

**OPERATIONS****b\_delai\_ecoule**

operation name

Definition of "b\_delai\_ecoule" in B\_Delais.mch

.....

**OPERATION PARAMETERS****fin\_del**

operation output parameter

Definition of "fin\_del" in B\_Delais.mch (b\_delai\_ecoule)

Modification of "fin\_del" in B\_Delais.mch (b\_delai\_ecoule)

Ces informations peuvent être incluses dans une documentation de projet générée automatiquement par l'Atelier B.

Attention : L'appel de cette fonction provoque une analyse sémantique des composants concernés.

**Interface utilisateur mode commande**

L'interface utilisateur est déjà lancée, vous avez déjà ouvert un projet. Pour obtenir des références croisées il faut utiliser la commande suivante :

**get\_project\_xref**

ou

**gpx**

Cette commande prend un ou deux paramètres :

1. Le filtre sur les identificateurs :
  - 0 pour tous les identificateurs du composant donné en second paramètre,

- 1 pour un identificateur du projet courant donné en second paramètre,
  - 2 pour tous les identificateurs du projet,
2. Le composant pour le filtre 0 ou l'identificateur pour le filtre 1.

## 6.7 Extraction de métriques

### Description

Cette fonction extrait des métriques sur une implémentation, ou sur toutes les implémentations d'un projet.

Ces métriques permettent :

- De mesurer la complexité d'une implémentation ou du projet analysé,
- De vérifier que l'implémentation ou le projet analysé obéissent à des règles de programmation,
- De mesurer la place mémoire nécessaire au minimum pour l'implémentation des structures de données utilisées dans les sources B.

Les métriques sont extraites par rapport aux données présentes dans un fichier de configuration. Ce fichier de configuration contient les informations suivantes :

- La valeur de référence pour chaque métrique,
- La liste des métriques à afficher dans le rapport projet ; le rapport projet est un tableau qui contient des métriques sur toutes les implémentations du projet,
- La liste des métriques à afficher dans le rapport implémentation ; le rapport implémentation est un tableau qui contient des métriques pour une implémentation donnée.
- La liste des métriques à afficher dans le rapport opération ; le rapport opération est un tableau qui contient des métriques pour une opération d'une implémentation donnée.

L'utilisateur peut utiliser les fichiers de configuration fournis par l'Atelier B ou créer ses propres fichiers de configuration.

Les fichiers de configuration fournis avec l'Atelier B se trouvent dans le sous répertoire **AB/press/lib/LC**. Ces fichiers ont un suffixe **.cvl**.

Ci-dessous, un exemple de rapport projet :

```
+-----+-----+-----+-----+-----+-----+-----+
```



Name	(1)	(2)	(3)	(4)	(5)	(6)
Reference Values	500	100	10	3	2	100
Distributeur_imp	14	12	1	2	1	5
Ecran_imp	13	6	1	!4!	1	1

(1)=NB\_INST\_OPER  
 (2)=NB\_INST\_SEQ  
 (3)=NB\_CTRL\_SEQ  
 (4)=NB\_CTRL\_IMB  
 (5)=NB\_WHILE\_IMB  
 (6)=LG\_CONDITION

La première ligne donne les noms des métriques qui sont rappelés après le tableau.

La seconde ligne rappelle la valeur de référence de chaque métrique.

Les lignes suivantes donnent pour chaque implémentation du projet les valeurs des métriques.

Si une valeur de métrique est supérieure à la référence, elle est écrite de la façon suivante :

!<valeur> ! (exemple !4!).

Ci-dessous un exemple de rapport implémentation :

METRICS FOR IMPLEMENTATION : Ecran\_imp

Title	Value	Ref	% > ref	CR
NB_INST_OPER	13	500	--	OK
NB_INST_SEQ	6	100	--	OK
NB_CTRL_SEQ	1	10	--	OK
NB_CTRL_IMB	4	3	25	KO
NB_WHILE_IMB	1	2	--	OK
LG_CONDITION	1	100	0	OK

La première colonne donne les noms des métriques, la seconde donne la valeur maximum de chaque métrique sur toutes les opérations de l'implémentation, la troisième donne la valeur de référence de chaque métrique, la quatrième donne le pourcentage de dépassement par rapport à la valeur de référence. La dernière colonne indique OK si la valeur est en dessous de la valeur de référence, KO sinon.

Ci-dessous un exemple de rapport opération :

METRICS FOR OPERATION : message\_controler\_code

Title	Value	Ref	% > ref	CR
NB_INST_OPER	6	500	--	OK
NB_INST_SEQ	4	100	--	OK
NB_CTRL_IMB	4	3	25	KO
NB_CTRL_SEQ	1	10	--	OK
NB_WHILE_IMB	0	2	--	OK
LG_CONDITION	1	100	0	OK

La seconde colonne donne la valeur de chaque métrique pour l'opération analysée. La signification des autres colonnes du tableau est la même que pour le rapport implémentation. Le tableau suivant donne la liste des métriques disponibles :

Code métriques	Calcul effectué
NB_MACH_SEE	nombre de machines vues
NB_MACH_IMPORT	nombre de machines importées
NB_MACH_EXTEND	nombre de machines étendues
NB_SET_ENUM	nombre d'ensembles énumérés
NB_SET_ABSTRAIT	nombre d'ensembles abstraits
NB_ITEM_ENUM	nombre maximal d'éléments dans un ensemble énuméré
NB_PAR_MACH	nombre de paramètres formels de la machine
NB_TOT_ENUM	nombre total d'éléments énumérés
NB_VAR_CONC	nombre de variables concrètes
NB_OPER	nombre d'opérations
NB_CONST	nombre de constantes concrètes
NB_VAR_LOCAL	nombre maximal de variables locales par opération
NB_PAR_ENTREE	nombre de paramètres d'entrée par opération
NB_PAR_SORTIE	nombre de paramètres de sortie par opération
NB_INST_OPER	nombre total d'instructions dans une opération
NB_CTRL_IMB	nombre maximal d'instructions de contrôle imbriquées dans une opération
NB_CTRL_SEQ	nombre maximal d'instructions de contrôle en séquence dans une opération
NB_WHILE_IMB	nombre de while imbriqués
NB_INST_SEQ	nombre d'instructions en séquence dans une opération
NB_VAR_IN	nombre de VAR IN dans une opération
LG_PREFIXE	taille maximale des préfixes de renommage
LG_PAR_MACH	taille maximale d'un paramètre machine
LG_IMP	taille du nom de l'implémentation
LG_PAR_ENTREE	taille maximale d'un paramètre d'entrée d'une opération
LG_PAR_SORTIE	taille maximale d'un paramètre de sortie d'une opération
LG_SET	taille maximale d'un identificateur d'ensemble
LG_ITEM_SET	taille maximale d'un identificateur d'élément d'un ensemble
LG_CST	taille maximale d'un identificateur de constante visible
LG_VAR_CONC	taille maximale d'un identificateur de variable visible
LG_LITERAL	taille maximale d'un littéral chaîne de caractère dans une opération
LG_VAR_LOCAL	taille maximale d'un identificateur de variable locale
LG_OPER	taille maximale d'un identificateur d'opération
LG_OPER_MACH	taille maximale d'un identificateur opération + identificateur machine
LG_CONDITION	taille maximale d'une expression de condition pour une opération (nombre d'opérateurs)
PLACE_TABLEAU	place mémoire occupée par les tableaux
NB_OP_PROMUE	nombre d'opérations promues et étendues
PLACE_VAR_CONC	place mémoire occupée par les variables visibles de type autre que les tableaux
PLACE_CST	place mémoire occupée par les constantes visibles de type autre que les tableaux

### interface graphique utilisateur

Cette fonction n'est pas disponible dans l'interface graphique.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez ouvert un projet.

Pour calculer des métriques sur toutes les implémentations du projet, il faut taper la commande suivante :

```
lchecker_project path_config output  
ou  
lcp path_config output
```

Le premier paramètre est le chemin complet du fichier de configuration à utiliser, par exemple `<rep_atelierb>/AB/press/lib/LC/CONFIG_clause.cvl`.

Le second paramètre est le format de sortie ; les valeurs 0, 4 et 8 correspondent respectivement à un affichage en  $\text{\LaTeX}$ , ASCII et RTF. Les valeurs 2 ou 5 correspondent respectivement à une impression en  $\text{\LaTeX}$  ou ASCII.

Pour calculer les métriques sur une implémentation du projet, il faut taper la commande suivante :

```
lchecker_mach nom_imp path_config output  
ou  
lcm nom_imp path_config output
```

Le premier paramètre est le nom de l'implémentation. Les deux autres paramètres ont la même signification que pour la commande précédente.

## 7 Documentation des projets B

### 7.1 Présentation

Ces commandes ne sont disponibles que si le poste de travail dispose d'outils complémentaires, dont  $\text{\LaTeX}$ , un visualisateur de fichiers PDF, ou Microsoft Word.

L'Atelier B fournit plusieurs commandes permettant de créer automatiquement des documents complets contenant les informations suivantes :

- Les fichiers sources B,
- Les fichiers de règles utilisateur (`.pmm`),
- Les tableaux d'état du projet et de chaque composant,
- Les différents graphes,
- les références croisées.

Dans ces documents :

- Les symboles du langage B sont affichés en utilisant des polices mathématiques,
- Les mots-clé du langage sont affichés en caractères gras.

Les deux commandes proposées par l'Atelier B sont :

- Affichage d'un source B uniquement,
- Création d'un document complet pouvant contenir toutes les informations précitées dans un ordre choisi par l'utilisateur.

### 7.2 Afficher un source B

#### Description

Cette fonction s'applique sur n'importe quel composant d'un projet B. Le fichier source B (spécification, raffinement ou implémentation) est converti dans un format choisi.

Lors de la conversion du fichier source B, l'utilisateur peut choisir deux types de présentations, il peut :

- Conserver sa présentation d'origine,
- Utiliser la présentation fournie par l'Atelier B.

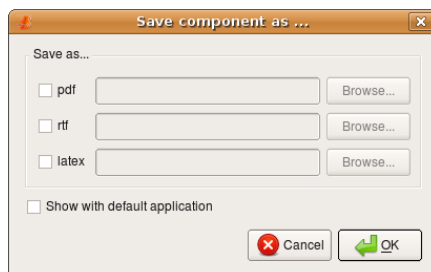
Les commentaires sont supprimés du fichier sources B, seuls les commentaires compris entre les séquences de caractère “/\*” et ”\*/” sont conservés.

### Interface utilisateur graphique

On considère l'interface déjà lancée, un projet déjà ouvert.

1. Sélectionnez le composant dans la liste des composants,
2. Sélectionnez le menu “*Component -> Save as...*”

La fenêtre suivante apparaît :



Sélectionnez alors vos options d'export puis cliquez sur *Ok*.

### Interface utilisateur mode commande

L'interface utilisateur est déjà lancée, vous avez ouvert un projet.

Pour afficher et convertir un source B au format d'un traitement de texte, il faut utiliser l'une des commandes suivantes :

- `show_doc_latex` ou `sdl` : Cette fonction convertit le source B au format  $\text{\LaTeX}$ .
- `print_doc_latex` ou `pd1` : Cette fonction convertit le source B au format  $\text{\LaTeX}$ , puis envoie automatiquement le résultat à l'imprimante courante. Pour modifier l'imprimante courante, utilisez la fonction `set_print_params`
- `create_doc_rtf` ou `cdr` : Cette fonction convertit le source B au format Word. Le nom du fichier produit est affiché dans la fenêtre de lancement. Vous devez ensuite éditer le fichier par l'intermédiaire de Microsoft Word, ou tout éditeur compatible RTF.

Le premier paramètre de ces fonctions est le nom du composant. Le second paramètre est le type de présentation souhaitée, si vous souhaitez conserver la présentation d'origine, il faut donner la valeur `PLAIN`. Il faut utiliser la valeur `NORM` pour avoir une présentation normalisée.

### Paramètres utilisables

ATB*OPT_TOOLS_<SYSTEM>*Latex_Binary_Directory
Configuré à l'installation de l'Atelier B sous systèmes UNIX.
Répertoire où trouver les binaires $\text{\LaTeX}$

## 8 Modification des paramètres de l'Atelier B

### 8.1 Introduction

Cette partie décrit comment l'utilisateur peut paramétrer l'Atelier B afin de modifier :

- Les outils connexes utilisés par l'Atelier B (éditeur, browser HTML, etc ...),
- La place mémoire occupée par l'Atelier B,
- Les paramètres des outils de l'Atelier B,
- Le script d'impression des documents générés par l'Atelier B.

### 8.2 Présentation du système de paramétrage

Il existe trois niveaux de paramétrage possible d'un Atelier B :

- Créer un fichier de ressources **.AtelierB** dans le répertoire personnel de l'utilisateur (\$HOME sous UNIX) qui paramétrera tous les Atelier B lancés par l'utilisateur concerné. Il est toutefois déconseillé d'utiliser ce fichier, car cela peut rentrer en conflit avec les configurations de projet,
- Créer un fichier AtelierB dans le répertoire BDP de son projet, qui paramètre l'Atelier lorsqu'un utilisateur ouvre le projet concerné,
- Créer un fichier quelconque que l'utilisateur indique explicitement à l'Atelier B lorsqu'il veut l'utiliser.

Lorsqu'une même ressource est décrite dans plusieurs de ces fichiers, l'ordre de priorité est le suivant : le fichier donné explicitement est prioritaire, puis vient le fichier associé au projet, et enfin celui associé à l'utilisateur. Si deux fichiers sont donnés explicitement, le deuxième est prioritaire sur le premier.

Si une ressource n'est décrite dans aucun de ces fichiers, elle prend la valeur par défaut définie dans le fichier de ressources général de l'Atelier B.

Les différents paramètres de l'Atelier B sont, par défaut, définis dans le fichier **AB/AtelierB**. Ce fichier contient les paramètres communs à tous les utilisateurs de l'Atelier B. Il est également possible de définir des paramètres communs à tous les utilisateurs de l'Atelier B, ainsi que des paramètres spécifiques pour un utilisateur ou pour un projet B.

Le format de ce fichier est le suivant : les lignes commençant par un point d'exclamation '!' sont des commentaires ; les autres contiennent chacune le nom d'une ressource suivi de sa valeur.

Le contenu initial de ce fichier est décomposé en plusieurs parties :



- Les localisations des outils internes :  
**ATB\*ATB\*<nom\_de\_la\_ressource>**
- Les localisations des outils optionnels :  
**ATB\*OPT\_TOOLS\_<système>\*<nom\_de\_la\_ressource> <système>** vaut  
Linux pour le système d'exploitation Linux, etc.
- Les ressources des outils internes :  
**ATB\*<outil\_concerné>\*<nom\_de\_la\_ressource>**

Exemple :

**ATB\*POG\*Generate\_Obvious\_PO : FALSE**

avec cette ressource positionnée à FALSE, l'Atelier B ne sauvegarde pas les obligations de preuve triviales.

Pour modifier une ressource pour tous les utilisateurs de l'Atelier B, la procédure à suivre est donc la suivante :

1. Se connecter sous le compte **atelierb** (sous UNIX uniquement),
2. Se placer dans le répertoire d'installation de l'Atelier B,
3. Éditer le fichier de ressources de l'atelier,
4. Modifier la ligne de la ressource concernée avec la valeur souhaitée,
5. Sauver les modifications.

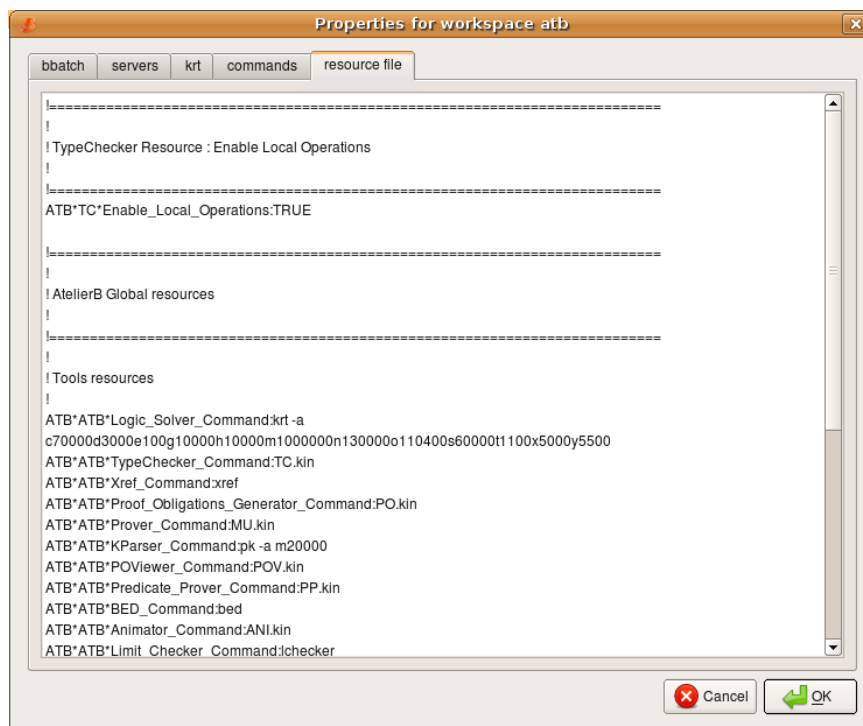
## 8.3 Création d'un fichier de ressources

### Description

Cette fonction permet d'éditer les fichiers de ressources. Si le fichiers n'existe pas, il est créé, et initialisé avec un modèle. Le modèle contient la liste de toutes les ressources. Pour activer une ressource, il suffit de décommenter la ligne et de mettre à jour la valeur de la ressource.

### Interface graphique utilisateur

Pour éditer un fichier de ressources d'un espace utilisateur, sélectionnez ce dernier, utilisez le menu *"Workspace -> Properties"*, et cliquez sur l'onglet *resource file*. Un aperçu de la fenêtre attendue est donné ci-après :



Pour éditer le fichier de ressources projet, ouvrez le projet souhaité, actionnez le menu *"Project -> Properties"*, puis cliquez sur l'onglet *"resource file"*. Une fenêtre du même type que celle de la capture précédente s'ouvre alors.

## 8.4 Affichage des valeurs des ressources et de la version de l'Atelier B

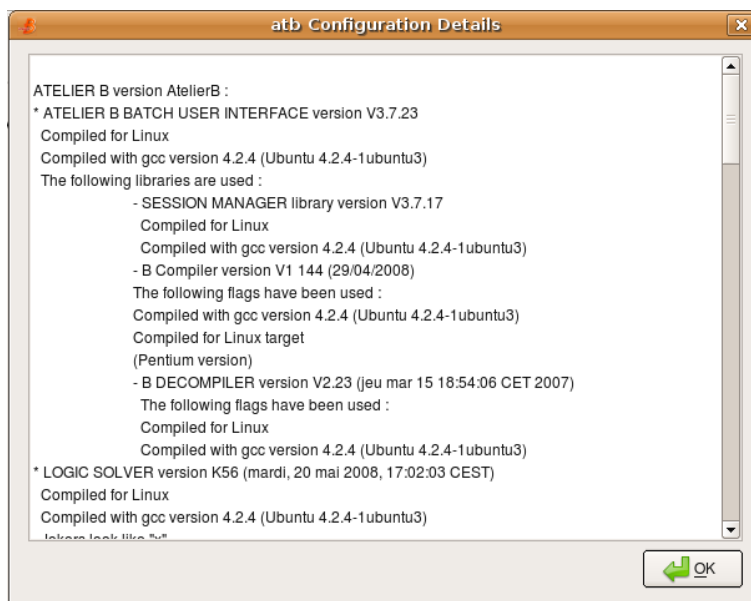
### Description

La commande d'affichage de version affiche en outre les informations suivantes :

- La version globale de l'Atelier B,
- La version de tous les outils de l'Atelier B,
- Les valeurs courantes des ressources.

### Interface utilisateur graphique

Allez dans les propriétés de l'espace de travail, et sur le premier onglet (*"bbatch"*), vous disposez d'un bouton *"Configuration details"*. Un click sur ce dernier aura pour effet d'ouvrir une fenêtre ressemblant à celle ci-dessous :



## 8.5 Modification des outils connexes

### Présentation

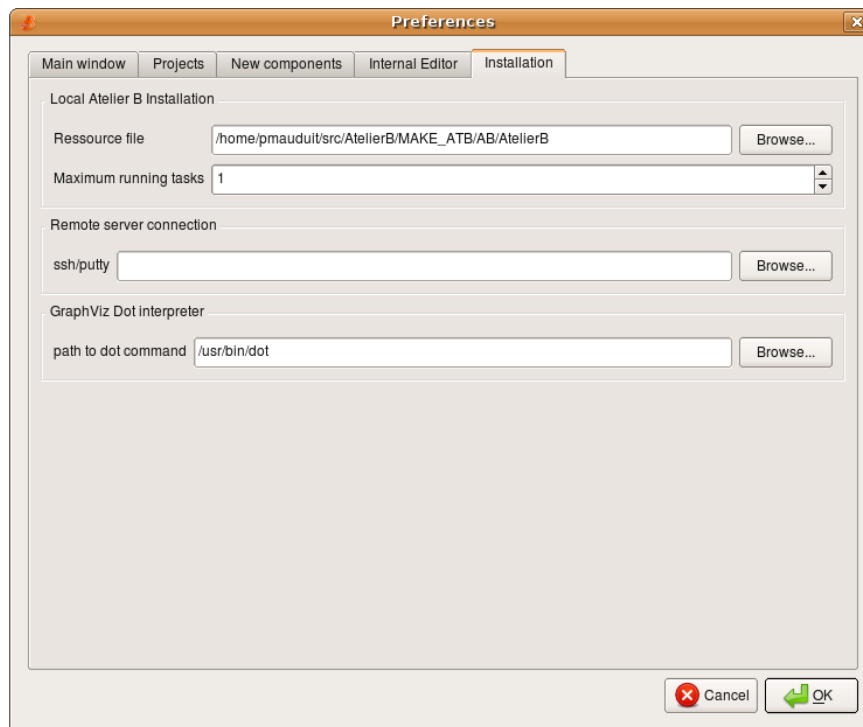
Les localisations des outils optionnels sont définis par défaut à l'installation, et stockées dans le fichier de ressources **AB/AtelierB**.

Pour modifier l'un de ces outils pour tous les utilisateurs de l'Atelier B, il faut modifier la valeur de la ressource associée :

Outil	Ressource
Éditeur de texte	ATB*OPT_TOOLS_<SYSTEM>*Editor_path
L <sup>A</sup> T <sub>E</sub> X	ATB*OPT_TOOLS_<SYSTEM>*Latex_Binary_Directory

### Interface utilisateur graphique

Il est possible de configurer certains outils connexes directement dans l'interface graphique, indépendamment des ressources présentées précédemment. Sélectionnez dans le menu de l'interface "Atelier B -> Preferences" et cliquez sur l'onglet "Installation". La fenêtre suivante s'affiche alors :



Vous pouvez de cette interface reconfigurer le chemin d'accès au binaire SSH et celui de DOT (fourni par le paquetage GraphViz).

## 8.6 Modification de la place mémoire du Logic Solver

La modification des paramètres du Logic Solver doit être effectuée dans les cas suivants :

- Votre machine ne possède pas assez de mémoire pour exécuter l'Atelier.  
Message du type :  
`Cannot launch the Logic Solver (check if there is enough memory).`
- La complexité de votre projet B vous oblige à augmenter la taille mémoire.  
Messages du type :  
`Compiler Memory Full`  
`SEQUENCE ID NUMBERS OVERFLOW`  
`SEQUENCE MEMORY OVERFLOW`  
`SEQUENCE _chn ID NUMBERS OVERFLOW`  
`SEQUENCE _chn MEMORY OVERFLOW`  
`stopping forward because ...`  
`OBJECTS OVERFLOW`  
`MAXIMUM NUMBER OF THEORIES xx REACHED`  
`GOAL STACKS OVERFLOW`

SYMBOLS OVERFLOW at <symb>

Pour paramétrer le Logic Solver pour tous les utilisateurs, il faut modifier la valeur de la ressource **ATB\*ATB\*Logic\_Solver\_Command** dans le fichier **AB/AtelierB** (c.f. chapitre 8.2).

Suivant la taille du noyau souhaitée, il faut remplacer la ligne :

```
ATB*ATB*Logic_Solver_Command: krt
```

par une des lignes suivantes :

- pour un petit Logic Solver :  

```
ATB*ATB*Logic_Solver_Command: krt -a m700000e40
```
- pour un Logic Solver moyen :  

```
ATB*ATB*Logic_Solver_Command: krt -a m1000000e40
```
- pour un Logic Solver large :  

```
ATB*ATB*Logic_Solver_Command: krt -a m4000000e40
```
- pour un Logic Solver extra-large :  

```
ATB*ATB*Logic_Solver_Command: krt -a m6000000e40
```

Si un utilisateur souhaite utiliser un Logic Solver paramétré différemment de celui utilisé par les autres utilisateurs, il doit créer un fichier **.AtelierB** dans son répertoire et y mettre la valeur de la ressource **ATB\*ATB\*Logic\_Solver\_Command** qu'il souhaite utiliser.

## 8.7 Modification de la place mémoire du Parser K

La modification des paramètres du Parser K doit être effectuée dans le cas où la taille de vos fichiers sources B est très importante et contient une grosse quantité de symboles différents :

Messages du type :

```
SEQUENCE ID NUMBERS OVERFLOW
SEQUENCE MEMORY OVERFLOW
SEQUENCE _chn ID NUMBERS OVERFLOW
SEQUENCE _chn MEMORY OVERFLOW
SYMBOLS OVERFLOW at <symb>
```

Pour modifier le paramétrage du Parser K pour tous les utilisateurs, il faut :

1. Se connecter en tant que **atelierb**,

2. Se placer dans le répertoire des exécutable de l'atelier,
3. Éditer le fichier de paramétrage du Parser K
4. Modifier les paramètres du Parser K sur la dernière ligne du fichier  
Le fichier contient pour chaque message d'erreur le nom du paramètre à modifier, ainsi que des exemples d'utilisation.
5. Sauver les modifications,
6. Se placer dans le répertoire de l'atelier,
7. Éditer le fichier de ressources de l'atelier,
8. Remplacer le Parser K par son script de paramétrage.

```
remplacer la ligne :  
ATB*ATB*KParser_Command: pk  
par la ligne :  
ATB*ATB*KParser_Command: sc_pk
```

9. Sauver les modifications.

Si un utilisateur souhaite utiliser un Parser K paramétré différemment de celui utilisé par les autres utilisateurs, il doit recopier le fichier `sc_pk` dans un de ses répertoires et y faire les modifications décrites au point 4. Puis il doit créer un fichier **.AtelierB** dans son répertoire personnel et créer la ligne :

```
ATB*ATB*KParser_Command: <repertoire choisi>/sc_pk
```

La procédure à suivre pour la création de ce fichier est décrite plus loin dans ce document. Attention, les autres fichiers décrits dans ce document ne permettent pas de modifier la ressource *KParser\_Command*.

## 8.8 Paramétrage du script d'impression

### Description

L'Atelier B est livré avec un utilitaire d'impression nommé `bprint`. Ce dernier est invisible de l'interface graphique utilisateur, mais peut être utilisé via l'interface en ligne de commande.

Par défaut, le script d'impression utilisé par l'AtelierB est le script `atelierb_directory/AB/bbin/bprint`. Pour utiliser un autre script d'impression :

1. Créez un nouveau script,

2. Modifiez la valeur de la ressource **ATB\*ATB\*Print\_Command** dans le fichier **AB/AtelierB** en donnant le chemin complet du nouveau script.

## Annexes

### Limitations des outils de documentation projet

Les sorties au format *Word* (.rtf) sont limitées. Cela est dû aux possibilités des logiciels et des formats.

#### Limitations de Word :

1. Le logo n'est pas inclus dans le document généré.
2. La table des matières n'est pas générée.
3. Il y a des précautions à prendre lors de l'inclusion de fichiers PostScript :
  - Il n'y a pas de vérification de présence du fichier à la génération,
  - Le fichier doit être présent sur le disque,
  - Word doit savoir reconnaître le fichier PostScript,
  - L'imprimante doit savoir interpréter le PostScript.
4. Les fichiers sont destinés à des PC et ne sont à priori pas utilisables directement sur systèmes Mac.



## Fichiers créés par l'Atelier B

Le tableau suivant décrit tous les fichiers créés par l'Atelier B :

Fichier	Localisation	Contenu
nom_projet.desc	base de données atelier	descripteur de projet (répertoires, gérant, utilisateurs, librairies)
nom_projet.db	base de données projet	composants du projet (nom, localisation, propriétaire)
.usedby_*	base de données projet	Marqueur indiquant que le projet est ouvert par un utilisateur
.project	base de données projet, répertoire des traductions	marqueur indiquant que le répertoire est occupé par un projet
.lib	base de données projet	répertoire des bdp des projets bibliothèques
*.lock	base de données projet	marqueurs utilisés pour réaliser l'exclusion mutuelle entre les utilisateurs d'un même projet
deB*,versB*	/tmp	FIFOs de communication entre l'interface utilisateur et le Logic Solver
src/*.*	base de données projet	Fichiers sources B avec définitions expansées. Il y a un fichier par composant même si utilise des fichiers multi-composants.
expand_src/*.*	base de données projet	Fichiers sources B. Il y a un fichier par composant même si utilise des fichiers multi-composants.
*.nf	base de données projet	Forme normale des composants
*.tse	base de données projet	Table des symboles étendue générée par le B0Checker
*.po	base de données projet	Obligations de preuve des composants
*.opo	base de données projet	Obligations de preuve triviales des composants
*.pmi	base de données projet	Sauvegarde des preuves interactives des composants
*.pmm	base de données projet	Règles définies par l'utilisateur par composant
*.stc	base de données projet	État des composants lors de la génération des obligations de preuve
nom_projet.gdl	base de données projet	Graphe de dépendances
nom_projet.stg	base de données projet	Tableau "Status Project" au format SGML.
*.stg	base de données projet	Tableau "Status Component" au format SGML.
*.tex	base de données projet	Fichiers générés par les outils de documentation pour la traduction en LATEX.
*.doc	base de données projet	Fichiers générés par les outils de documentation pour la traduction en Interleaf.
*.dvi	base de données projet	Fichiers générés par LATEX pour les visualisation ou impressions.
*.ps	base de données projet	Fichiers LATEX convertis au format PostScript pour les impressions.

<code>*.but</code>	base de données projet	Fichiers temporaires générés par le prouveur de prédicats.
<code>*.bod, *.str, *.blf</code>	répertoire des traductions sous-répertoire <code>ada</code>	Fichiers objets générés par le traducteur ADA.
<code>*.ads, *.adb</code>	répertoire des traductions sous-répertoire <code>ada</code>	Fichiers générés par le traducteur ADA après édition de liens.
<code>makefile</code>	répertoire des traductions sous-répertoire <code>ada</code>	Directives pour la compilation ADA.
<code>*.bod, *.str, *.blf</code>	répertoire des traductions sous-répertoire <code>hia</code>	Fichiers objets générés par le traducteur HIA.
<code>*.ads, *.adb</code>	répertoire des traductions sous-répertoire <code>hia</code>	Fichiers générés par le traducteur HIA après édition de liens.
<code>makefile</code>	répertoire des traductions sous-répertoire <code>hia</code>	Directives pour la compilation HIA.
<code>*.bdy, *.spe, *.blf</code>	répertoire des traductions sous-répertoire <code>cpp</code>	Fichiers objets générés par le traducteur C++.
<code>*.cpp, *.h</code>	répertoire des traductions sous-répertoire <code>cpp</code>	Fichiers générés par le traducteur C++ après édition de liens.
<code>makefile</code>	répertoire des traductions sous-répertoire <code>cpp</code>	Directives pour la compilation C++.
<code>*.bdy, *.spe, *.blf</code>	répertoire des traductions sous-répertoire <code>c</code>	Fichiers objets générés par le traducteur C.
<code>*.c, *.h</code>	répertoire des traductions sous-répertoire <code>c</code>	Fichiers générés par le traducteur C après édition de liens.
<code>makefile</code>	répertoire des traductions sous-répertoire <code>c</code>	Directives pour la compilation C.