

Premiers pas avec Bouml

par [Bruno Pagès](#)

Date de publication : 6/11/2006

Dernière mise à jour :

Ce tutoriel est le premier d'une série de tutoriels à lire dans l'ordre, celui-ci vous permettra de faire vos premiers pas avec BOUML, un modelleur UML2 gratuit fonctionnant sous Windows, Linux, Solaris, MacOS X disponible [ici](#).

- I - Introduction
- II - Commencer à utiliser BOUML
 - II-A - C'est parti !
 - II-B - Créer un nouveau projet
 - II-C - Vue, Use case
 - II-D - Créer un diagramme
 - II-E - Ajouter des éléments dans un diagramme
 - II-F - Sélectionner des éléments dans un diagrammes
- III - Destructurations
 - III-A - Retirer un élément d'un diagramme
 - III-B - Détruire / annuler la destruction d'un élément du modèle
- IV - Configuration graphique
 - IV-A - Drawing settings
 - IV-B - Profondeur d'affichage
 - IV-C - Zoom, taille et format des diagrammes
 - IV-D - Exporter un diagramme
- V - Conclusion
 - V-A - Epilogue
 - V-B - Liens

I - Introduction

BOUML est une série de logiciels comprenant un **modeleur UML2** et plusieurs programmes externes dont des **générateurs de code** et **reverse**. BOUML ne couvre pas intégralement la norme UML2 à ce jour, mais le principal est déjà présent. J'ai distribué la première version fin février 2005 et je continue son développement, toujours sur mon temps libre en dehors de mes activités professionnelles.

BOUML est distribué sous licence GPL, il peut donc être librement utilisé pour modéliser et produire du code y compris à des fins commerciales, voir sur le [site](#) pour plus de détails, le télécharger et accéder à la documentation.

BOUML est développé en C++ au dessus de [Qt](#), ce qui en permet l'utilisation sous *Windows* et la constellation *Unix* (*Linux*, *Solaris*, *Mac OS X* etc.).

Le but de BOUML est de permettre une utilisation allant des besoins à la génération de code, les langages pris en compte à ce jour étant C++, *Java* et *IDL* (la présence de ce dernier peut paraître anecdotique, elle est liée au contexte professionnel qui était le mien lorsque j'ai débuté le développement de BOUML). Je n'ai pas voulu que BOUML soit un simple jouet, mais qu'il puisse être utilisé dans un cadre professionnel, ce qui signifie qu'il doit être pratique d'utilisation, qu'il doit être peu gourmand en ressources CPU et mémoire (cela semble être le cas, voir [benchmaks](#)) et qu'il puisse être utilisé dans un contexte multi-utilisateurs avec gestion de configuration.

BOUML est extensible via l'écriture de **plug-outs** qui sont des programmes vous permettant d'accéder et/ou de modifier automatiquement vos modèles, les générateurs de code et reverse en sont des exemples. Je ne peux que vous conseiller d'écrire vos propres *plug-outs*, ils vous permettent de ne pas être limité à la définition de BOUML tel qu'il est distribué. Les *plug-outs* correspondent aux scripts de *Rose* ou *Rhapsody* à ceci près qu'ils sont écrits en utilisant un *vrai* langage de programmation, à savoir C++ ou *Java*, et qu'ils s'exécutent en dehors du modeleur, ce qui permet d'en tuer l'exécution sans perdre votre modèle (souvenir personnel cuisant avec *Rose*...). L'écriture d'un *plug-out* est l'objet d'autres tutoriels.

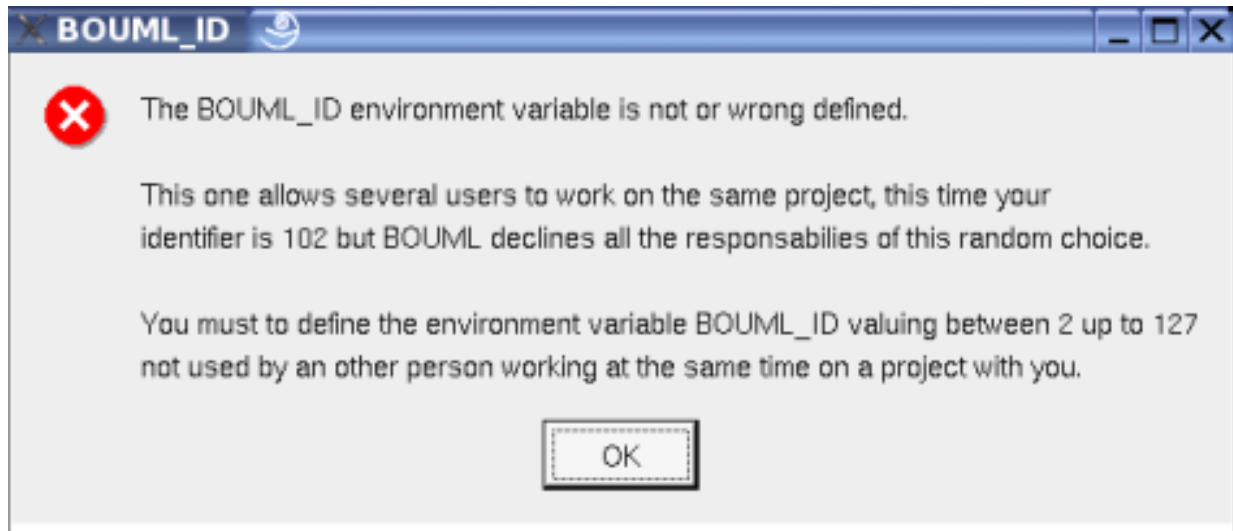
Seuls quelques points généraux de l'utilisation de BOUML sont exposés dans les tutoriels, seule la lecture de la documentation de référence disponible en Anglais sur le [site](#) décrit l'ensemble des caractéristiques de BOUML.

L'aspect graphique est légèrement dépendant de la version de Qt utilisée, les figures ont été réalisées en compilant BOUML avec Qt 2.3 disponible sous Windows et le monde Unix.

II - Commencer à utiliser BOUML

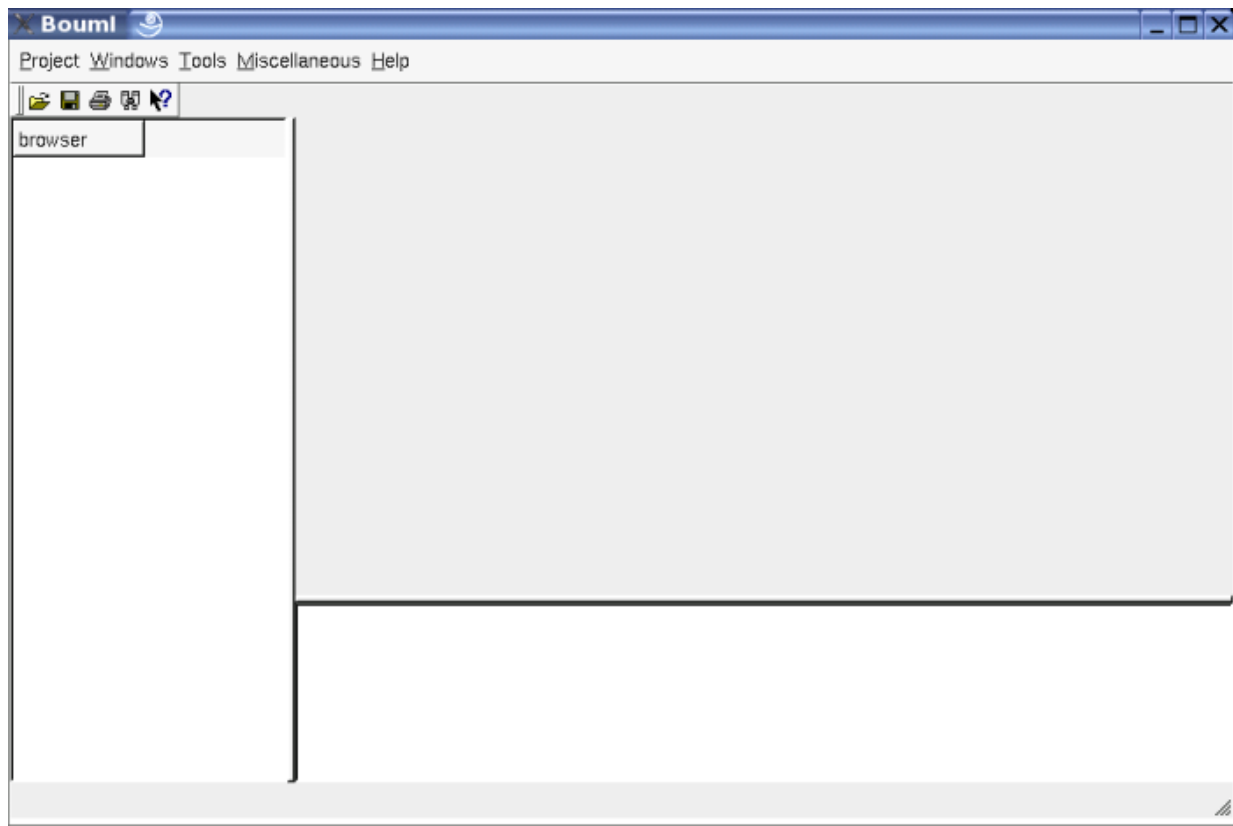
II-A - C'est parti !

Si vous lancez BOUML la première chose que vous verrez sera le message d'erreur suivant :



Ce n'est sans doute pas très agréable, mais BOUML l'affichera tant que n'aurez pas défini la variable d'environnement `BOUML_ID`. Vous devez choisir une valeur qui vous est propre, deux personnes travaillant en parallèle sur un même projet ne devant pas avoir le même `BOUML_ID`. Même si vous êtes seul, définissez cette variable. Grâce à cette définition, lorsque vous rechargerez un projet, les diagrammes qui étaient ouverts lors de la dernière utilisation du projet seront automatiquement ouverts et le *browser* semblable à ce qu'il était.

La fenêtre de BOUML apparaît :



Celle-ci comporte trois parties :

- La sous fenêtre gauche est un browser présentant votre projet, la navigation peut être faite avec la souris ou les flèches du clavier. La police de caractère grasse indique les éléments modifiables, un élément est en lecture seule lorsque vous n'avez pas les droits d'écriture pour les fichiers qui supportent sa définition (le *plug-out file control* vous permet de faire des *check-in/out* ou de modifier les droits des fichiers du modèle).
- La sous fenêtre en bas à droite est utilisée pour afficher la description du dernier élément sélectionné.
- La sous fenêtre en haut à droite est utilisée pour afficher les diagrammes, qui peuvent être vus comme des sous fenêtres ou maximisés.

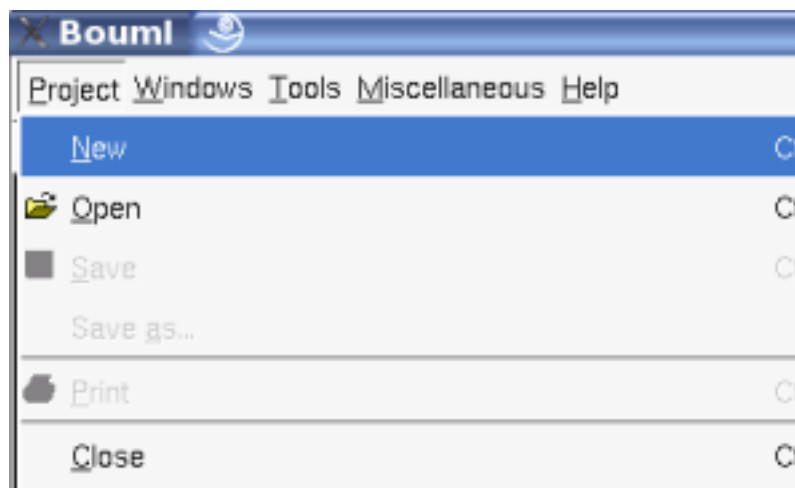
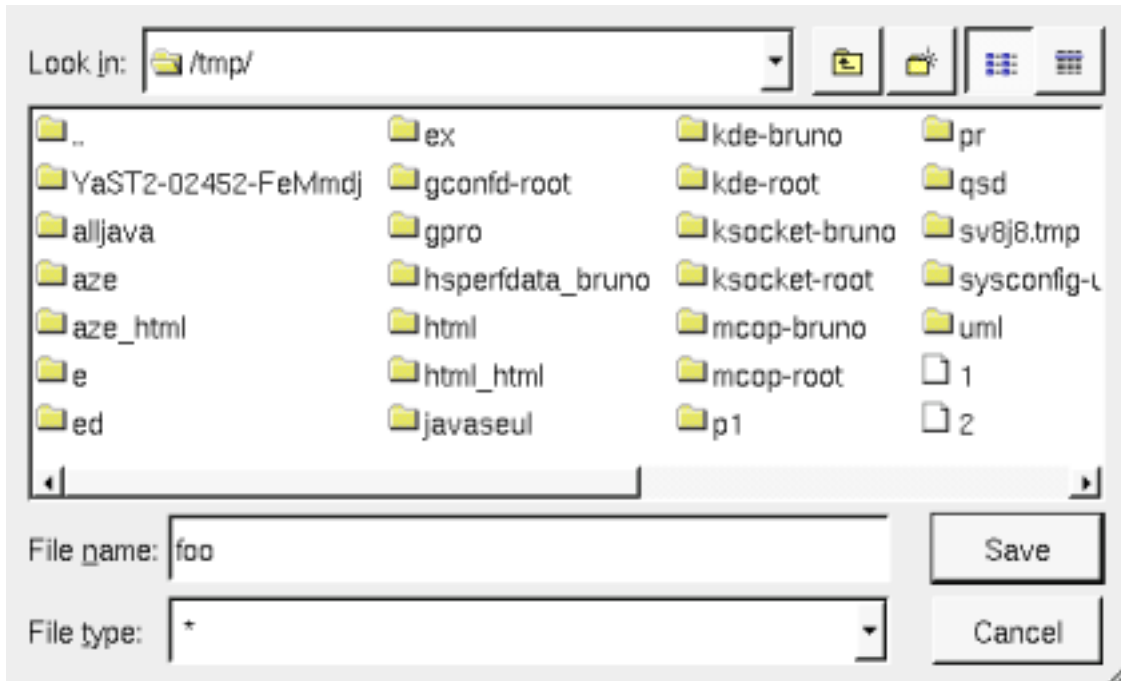
Bien évidemment, les tailles respectives des sous fenêtres peuvent être changées en plaçant la souris sur la barre les séparant.

Si vous avez une configuration avec plusieurs moniteurs je vous conseille de définir la variable d'environnement BOUML_LIMIT_DESKTOP, pour plus de détails voir [ici](#).

À ce niveau, vous pouvez lire un projet existant ou en créer un nouveau.

II-B - Créer un nouveau projet

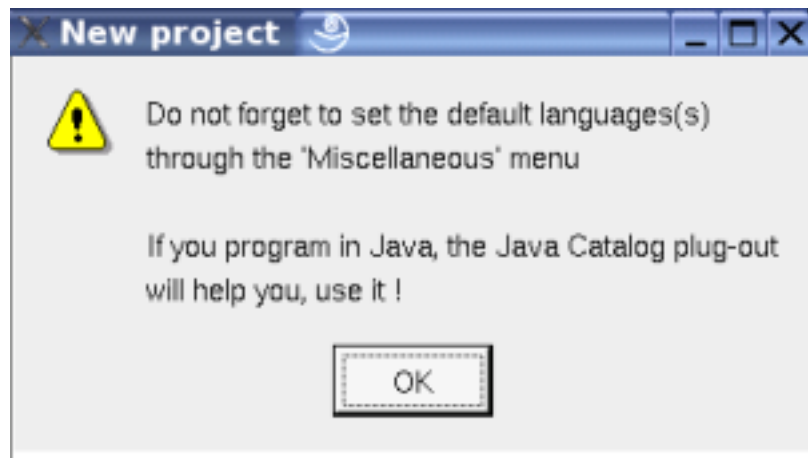
Dans le menu Project choisir *new*, un explorateur de fichier apparaît (son aspect dépend de l'OS utilisé et du gestionnaire de fenêtre), vous devez choisir un nom de projet et où le créer. Nommons le projet *foo* et plaçons le dans */tmp* (sous Windows vous pouvez prendre C: par exemple) :



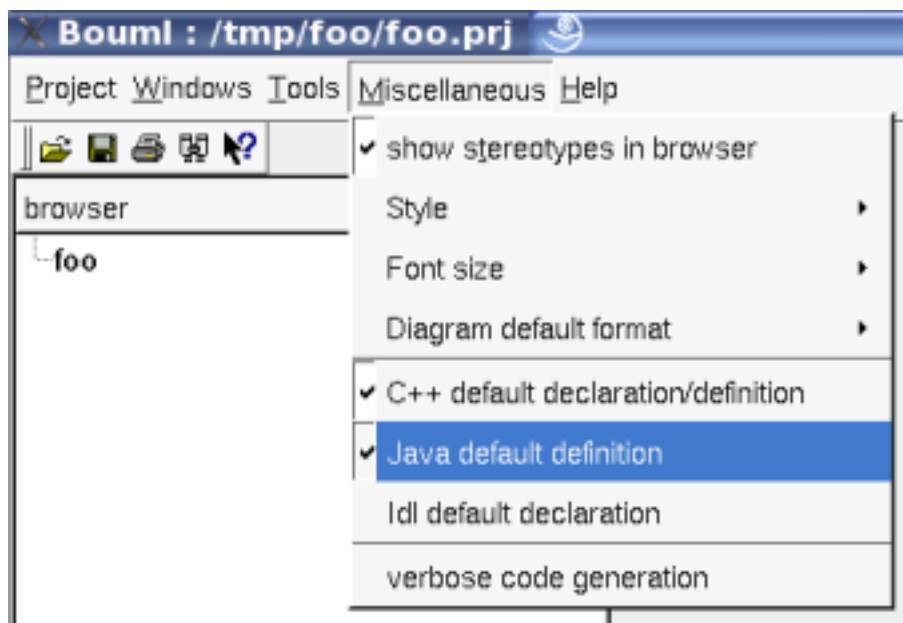
Dans ce cas BOUML crée le répertoire *foo* sous */tmp* et y crée quelques fichiers dont *foo.prj* qui représentent le projet et que vous choisirez quand vous rechargerez le projet une prochaine fois.

Ne renommez ni ne détruisez les fichiers produits par BOUML et ne renommez pas le répertoire les contenant (évidemment vous pouvez détruire totalement le répertoire !).

Lorsque vous aurez validé le nom du projet, un nouveau message apparaîtra :



C'est un pense bête, je suis la recommandation en demandant les définitions par défaut pour Java et C++ via le menu *Miscellaneous* menu car nous générerons du code pour ces deux langages (note : cela n'est pas obligatoire, et surtout pas de le faire pour ces deux langages, j'expliquerai plus en détail l'intérêt de la chose dans le second tutoriel traitant des classes) :



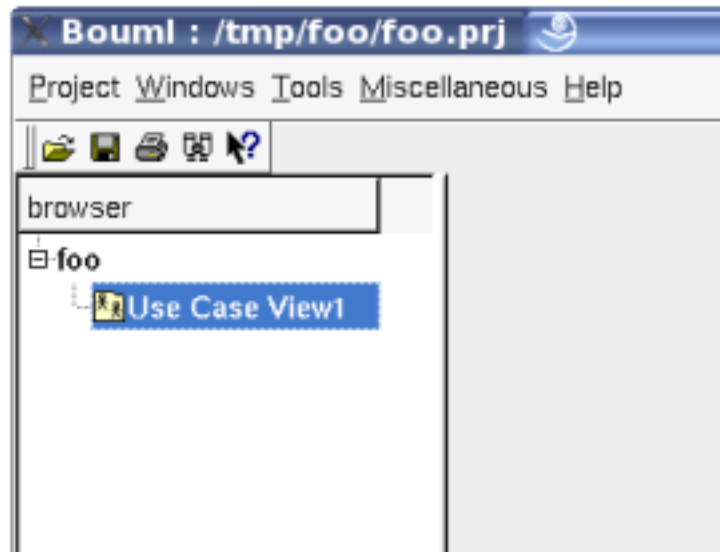
Le nom du projet (*foo*) apparaît en haut du *browser*, le projet est le *package* (au sens répertoire) de plus haut niveau, un clic droit dessus affiche le menu suivant :

foo
new package
new use case view
new class view
new component view
new deployment view
import project
edit
edit default stereotypes
edit class settings
edit generation settings
edit drawing settings
import ▶
referenced by
mark
generate ▶
reverse ▶
tool ▶

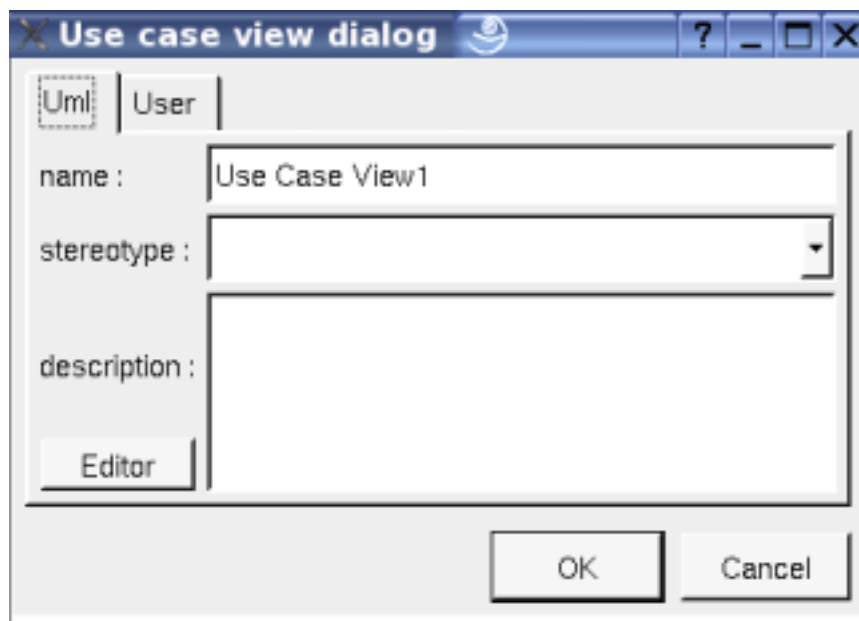
Comme vous pouvez le voir, un *package* peut contenir d'autres *packages* et des *vues* : *use case view*, *class view*, *component view* et *deployment view*. BOUML vous laisse une totale liberté sur la façon d'architecturer votre projet (sauf qu'un type donné de vue ne peut pas tout contenir), en particulier l'ordre des éléments est celui que vous voulez et non un ordre arbitraire tel que l'ordre alphabétique (voir le *plug-out sort*).

II-C - Vue, Use case

Pour définir des *use cases* vous devez utiliser les *use case view*, donc créons une *use case view* (vous pouvez en avoir plusieurs) *via* le menu précédant :

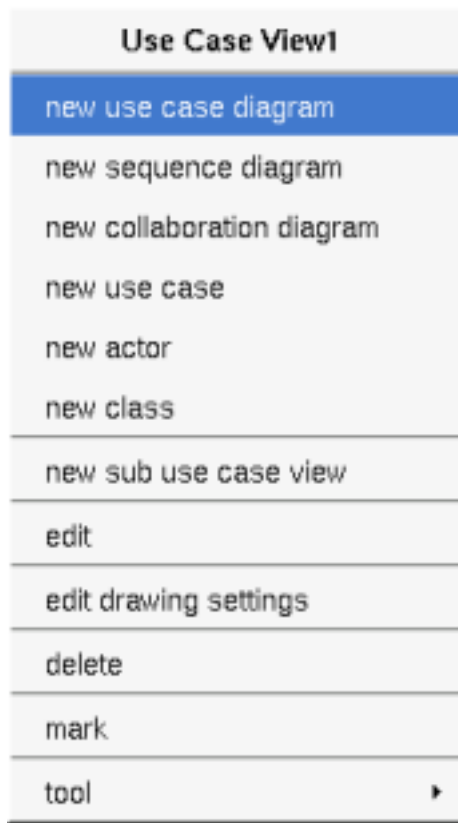


Le nom d'une vue n'étant pas véritablement déterminant, celui-ci est généré automatiquement, pour le changer vous éditez la vue *via* un double clic ou en choisissant *edit* dans menu apparaissant après un clic droit sur la vue, l'éditeur de *use case view* apparaît :

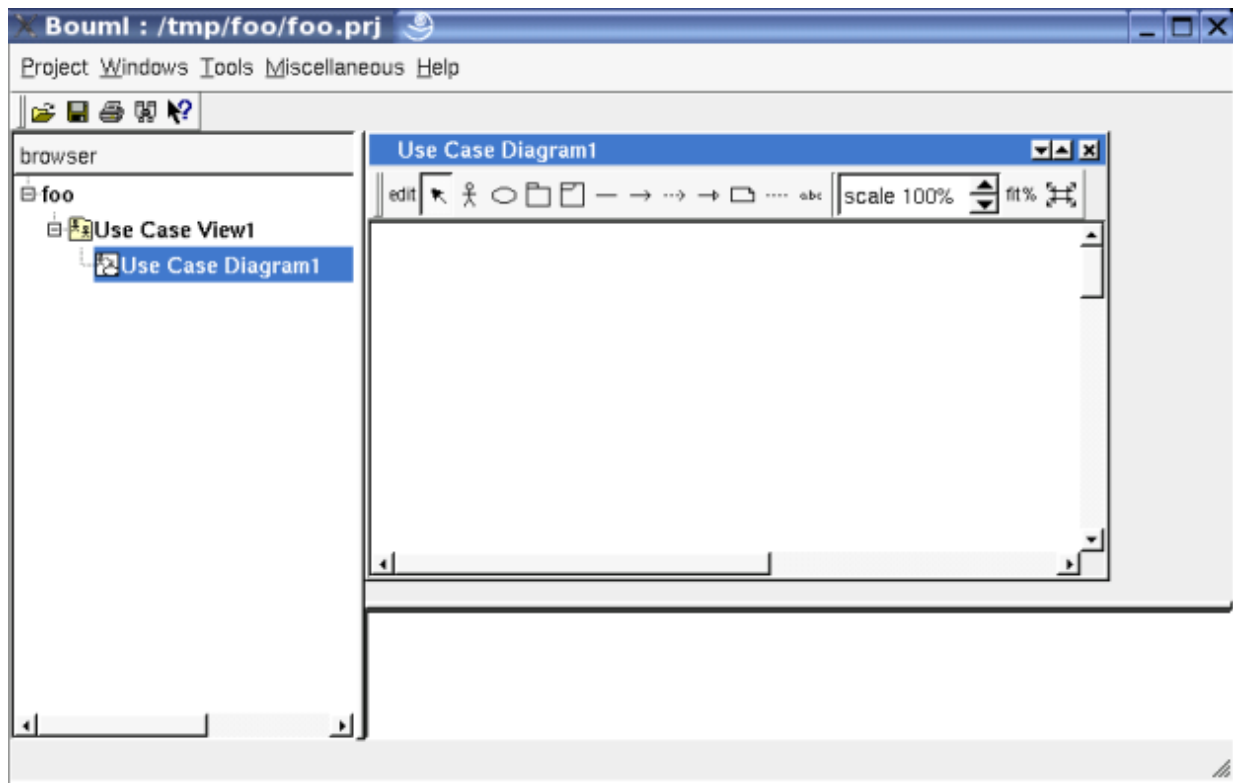


II-D - Créer un diagramme

Pour créer un *use case diagram* dans cette vue, faites un clic droit sur la vue dans le *browser* et choisissez l'entrée *new use case diagram* du menu :



Un nom est également arbitrairement choisi, pour le changer faites un clic droit sur le diagramme dans le *browser* et choisissez *edit* dans le menu. Normalement, un double clic sur un élément permet de l'éditer, sauf dans le cas particulier des diagrammes qui sont ouverts sur un double clic (c'est également ce qu'il se passe pour les éléments ayant un diagramme associé, voir *set associated diagram* dans le [manuel de référence](#)) :



II-E - Ajouter des éléments dans un diagramme

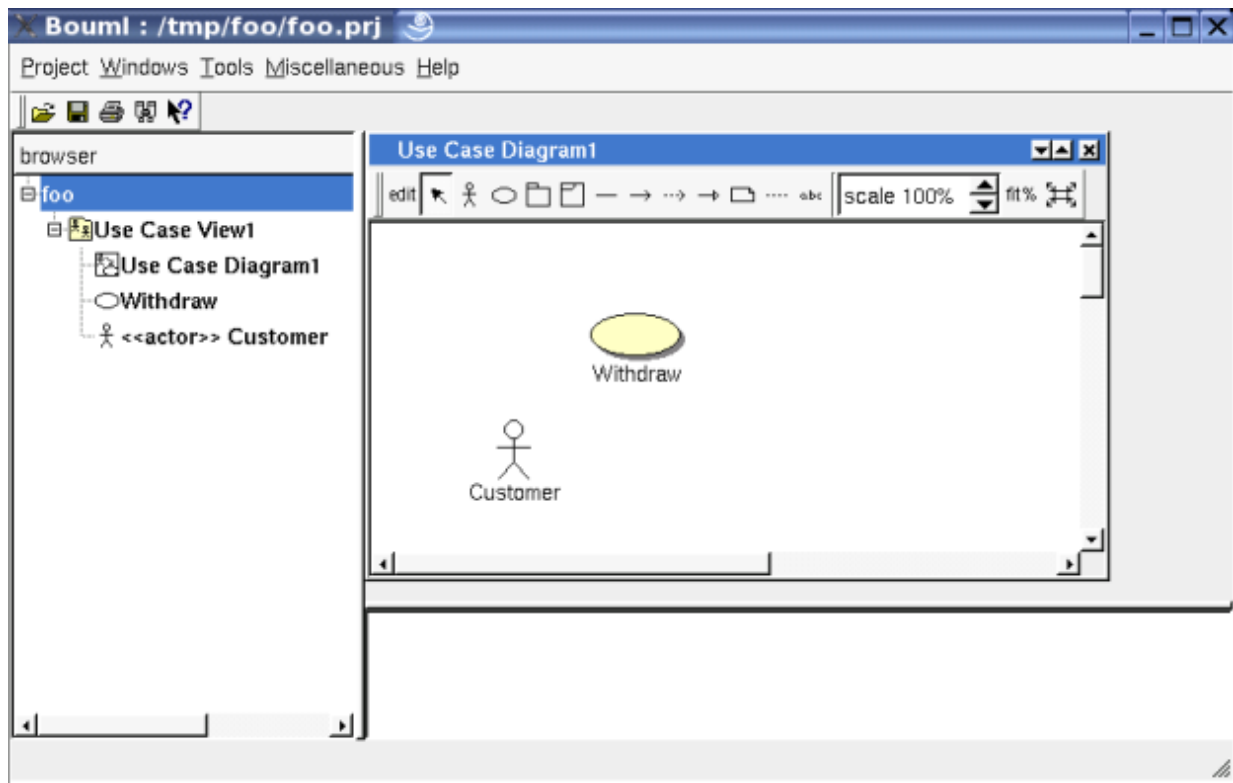
Pour créer et placer un *use case* dans le *use case diagram* vous avez deux possibilités :

- Faire un clic droit sur le *use case view* et choisir *new use case* dans le menu, puis le tirer (*drag and drop*) du *browser* dans le diagramme;
- Appuyez sur le bouton en forme d'ellipse en haut de la sous fenêtre du diagramme puis refaire un clic dans le diagramme. Remarquez que le *use case* est alors créé dans la vue contenant le diagramme (mais vous pouvez afficher des *use cases* n'appartenant pas à la même vue), ce sera la même chose dans les autres types d'éléments ou de diagrammes, sauf dans le cas particulier des machines à états et des activités.

Créez donc un *use case* et appelez le *Withdraw*.

Pour déplacer le *use case* dans le diagramme, faites un clic gauche dessus et déplacez la souris en maintenant le clic abaissé. Le nom suit le *use case* mais le nom peut être déplacé indépendamment, par exemple pour le mettre au centre de l'ellipse. Les déplacements peuvent aussi être faits en sélectionnant le *use case* et en utilisant les flèches du clavier.

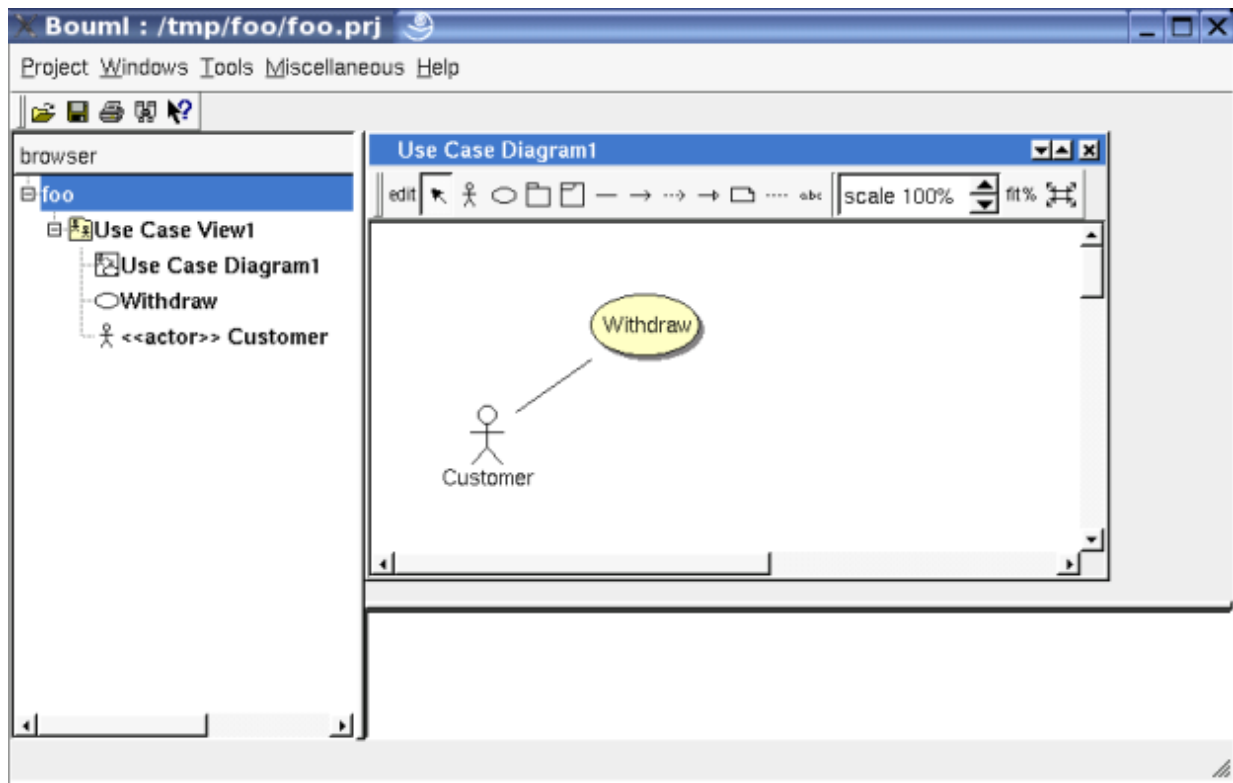
Créez un acteur nommé *Customer* de façon similaire à celle utilisée pour créer le *use case* pour avoir :



Remarque : un acteur est en fait une classe, une classe stéréotypée *actor* est dessinée avec l'icône correspondant dans le *browser* sinon avec l'icône d'une classe normale. Pour changer le stéréotype, éditez la classe (double clic sur la classe dans le *browser* ou clic droit dessus et choisissez *edit* dans le menu), choisissez parmi la liste des stéréotypes par défaut ou entrez-en un nouveau (voir le [référence manuel](#) pour modifier la liste des stéréotypes par défaut).

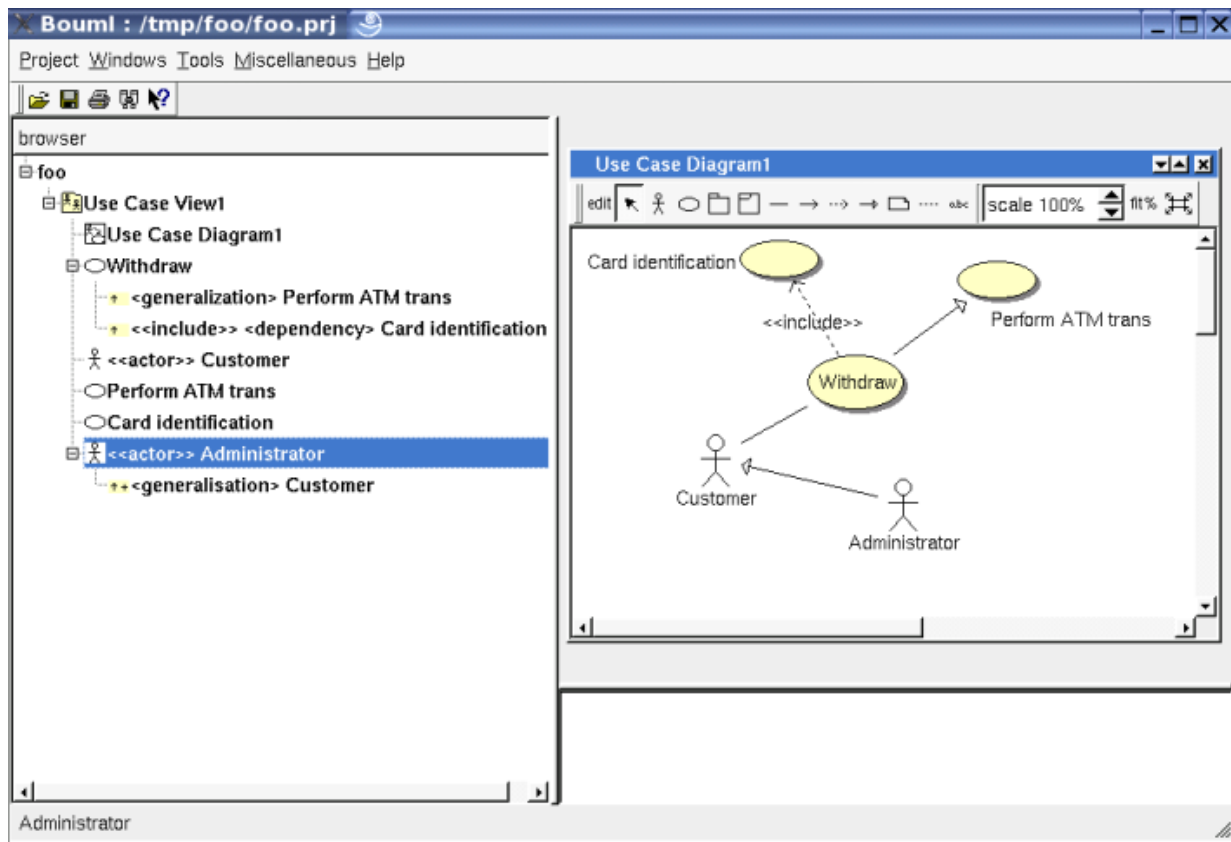
Le dessin d'un *use case* peut être re-dimensionné, sélectionnez le *use case*, cliquez sur un des carrés qui l'entourent et déplacez la souris en maintenant le clic appuyé. Note : tout les objets ne peuvent pas être redimensionnés, par exemple les acteurs.

Entre un acteur et un *use case* vous pouvez tracer une simple ligne comme le préconise la norme UML, soit utiliser une flèche. Appuyez sur le bouton désiré, abaissez le clic gauche sur l'acteur, déplacez la souris jusqu'au *use case* et relâchez le clic :



La ligne peut être coupée en tronçons lors de sa construction en relâchant le clic en dehors de tout élément puis en ré appuyant et déplaçant la souris etc, ou après construction en faisant un clic gauche sur le trait et en déplaçant la souris clic abaissé. Pour retirer un point de brisure, double clic sur celui-ci ou faites un clic droit dessus et choisissez *remove from view* dans le menu. Pour annuler la construction d'une ligne : double clic.

Ajoutez de nouveau éléments dans le diagramme comme ci-dessous (éditez la dépendance pour la stéréotyper *include*) :



Comme vous pouvez le voir, les généralisations et dépendances sont visibles dans le *browser*, et leur menu (clic droit) dans le *browser* permet de naviguer vers la cible.

Lorsque vous éditez la généralisation ou la dépendance entre les *use cases* vous obtenez un dialogue simple, cela n'est pas le cas pour la généralisation entre les acteurs car celle-ci peut produire du code : un acteur est une classe.

II-F - Sélectionner des éléments dans un diagrammes

Pour sélectionner plusieurs éléments dans un diagramme, par exemple pour ensuite les déplacer :

- *Control-a* sélectionne tout ;
- Un clic gauche sur un élément avec la touche contrôle appuyée permet d'ajouter ou retirer cet élément à la sélection ;
- Pour sélectionner les éléments contenus dans un rectangle appuyez sur le clic gauche en de hors de tout élément du diagramme, déplacez la souris en maintenant le clic et relâchez ;
- Choisir *select linked items* dans le menu d'un élément du diagramme sélectionne tous les éléments du diagramme directement ou indirectement reliés à l'élément initial par une ligne, pratique non ? Dans le cas des diagrammes précédents cela sélectionne tout car tous les éléments sont reliés.

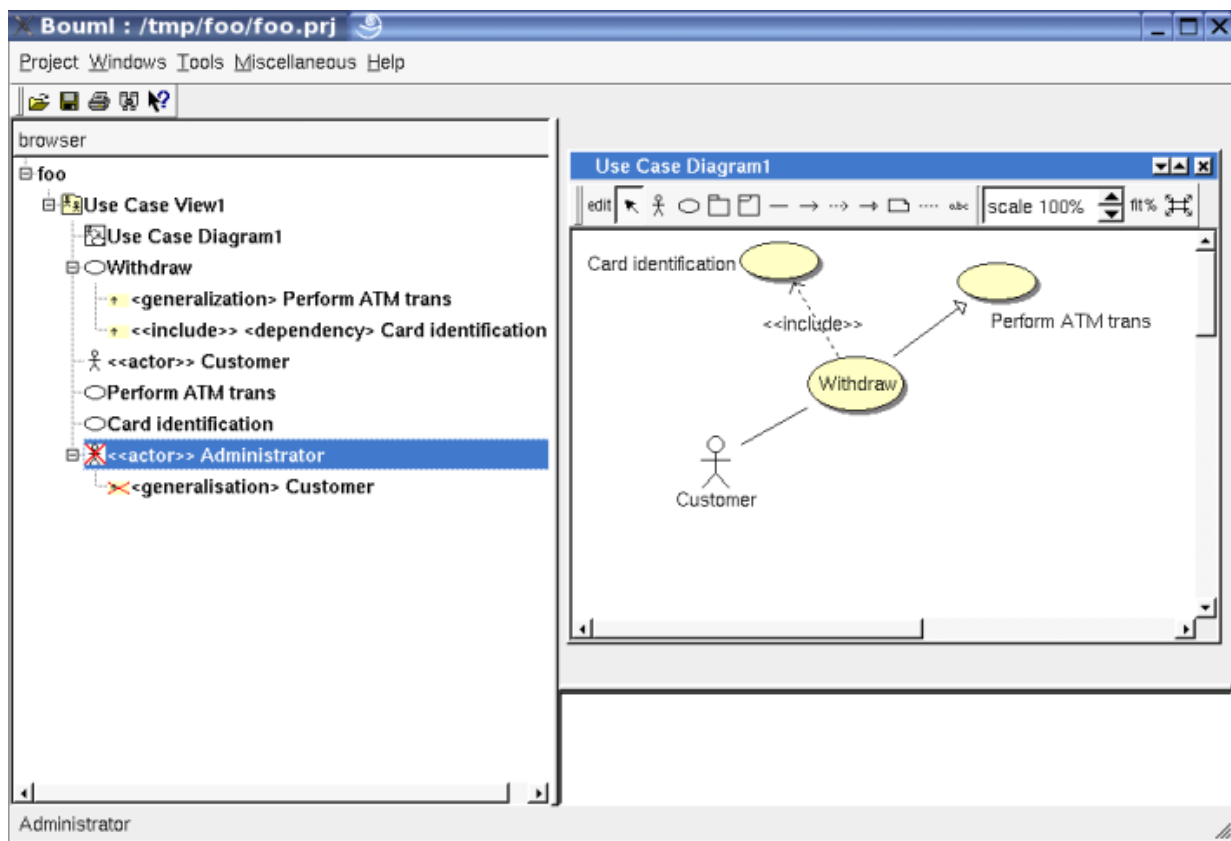
III - Destructures

III-A - Retirer un élément d'un diagramme

Pour retirer quelque chose dans un diagramme, sélectionnez ce qu'il faut retirer puis utilisez la touche *Suppr*, ou faites un clic droit après sélection ou sur l'objet à retirer pour faire apparaître le menu et choisissez *remove from view*. Remarque : il n'est pas possible de retirer une étiquette tel que le nom d'un *use case* sans retirer le *use case* lui-même.

III-B - Détruire / annuler la destruction d'un élément du modèle

Pour retirer quelque chose du modèle, sélectionnez la chose et faites un *control-d*, ou faire apparaître le menu *via* un clic droit sur la chose et choisissez *delete from model*, cela peut se faire dans le *browser* ou à partir de la représentation de l'élément dans un diagramme. Par exemple si je détruis *Administrator* :



Bien évidemment l'acteur *Administrator* et la généralisation ont disparu du diagramme, mais ce n'est pas le cas au niveau *browser*, ils sont seulement marqués comme étant détruits. Leur menu associé est modifié : vous pouvez juste annuler leur destruction, ainsi *Administrator* peut être restauré seul ou avec la généralisation (*undelete recursively*). Aucun menu n'apparaît pour la généralisation car celle-ci ne peut être restaurée tant que *Administrator* est détruit !

Les éléments marqués comme étant détruits ne sont pas sauvegardés, pour ne plus les voir il suffit donc de sauver le projet et de le relire.

Si vous annulez la destruction de l'acteur et de la généralisation ceux-ci ne réapparaissent cependant pas dans le diagramme.

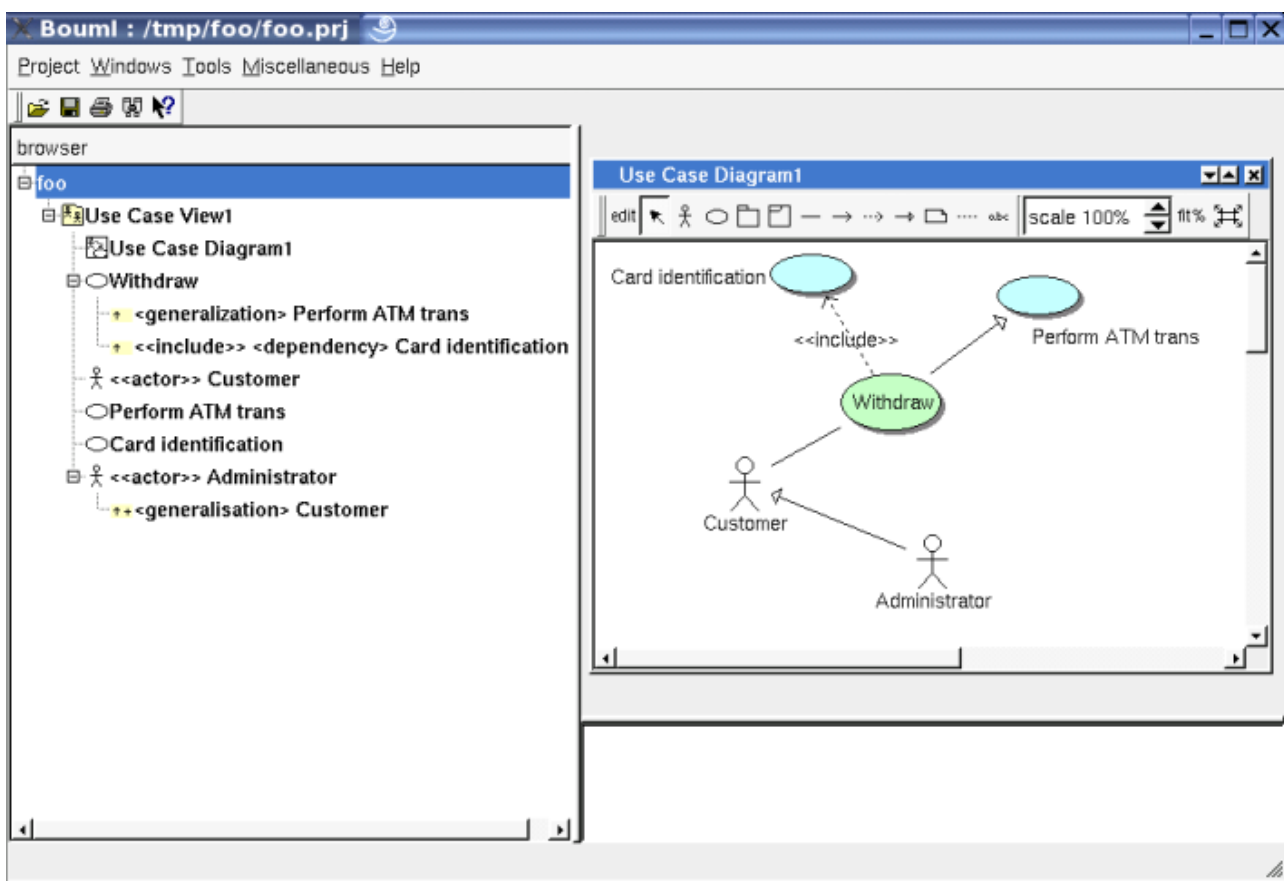
Redessinez l'acteur dans le diagramme en faisant un *drag and drop* ou en appuyant sur l'icône acteur puis en cliquant dans le diagramme et en sélectionnant *Administrator* dans la liste proposée. La généralisation n'est pas automatiquement dessinée, mais ce serait le cas par défaut dans un diagramme de classes (sauf contre ordre *via* les *drawing settings*). Pour voir la généralisation, faites un *drag and drop* de sa représentation dans le *browser* vers le diagramme, l'emplacement où vous la posez dans le diagramme n'ayant évidemment aucune importance puisqu'elle sera dessinée entre les acteurs.

IV - Configuration graphique

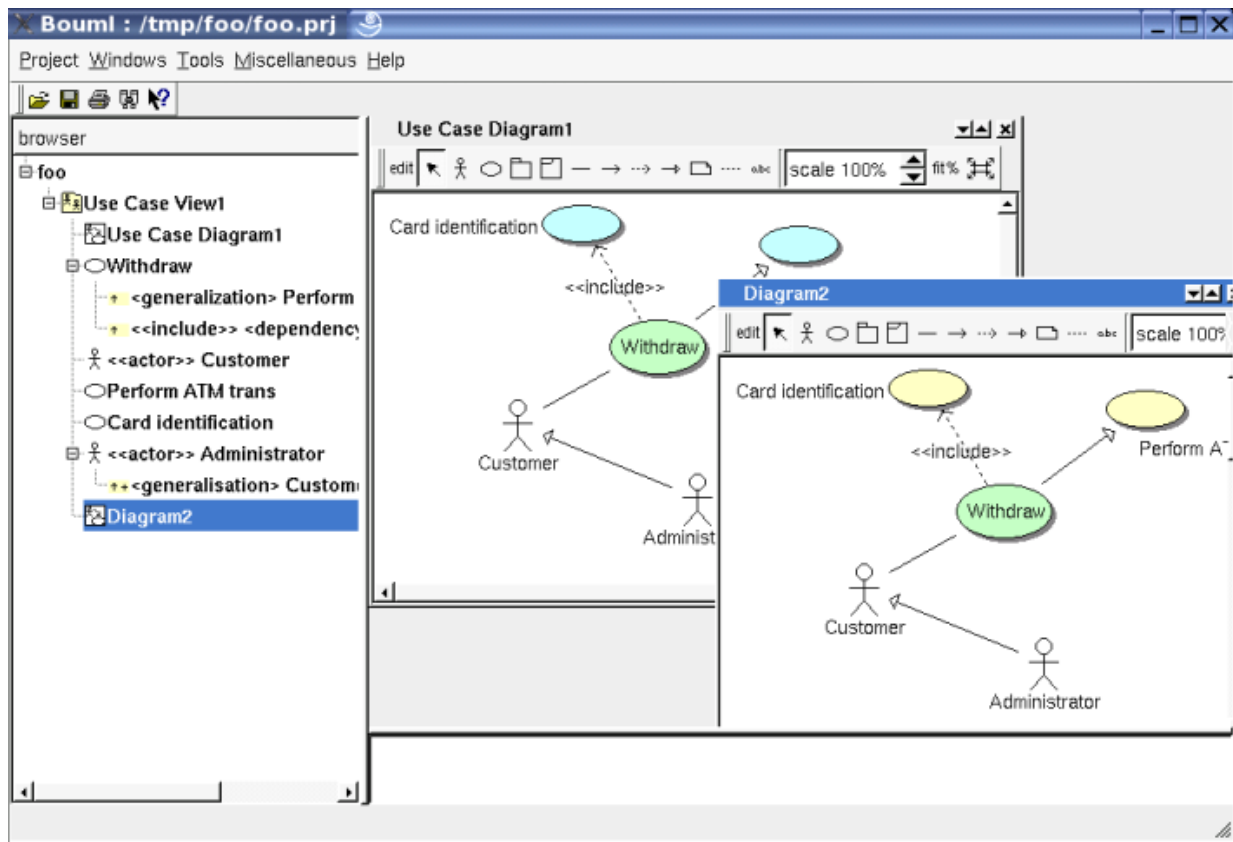
IV-A - Drawing settings

Les couleurs et autres caractéristiques graphiques peuvent être modifiées *via* les *drawing settings*. Si vous voulez toujours avoir des *use case* bleus, le mieux est d'appeler le menu du projet (i.e. un clic droit sur *foo* dans le *browser*) et de choisir *edit drawing settings*, allez dans le dernier intercalaire et changer *default use case color*. Si vous faites ce changement non pas au niveau du projet mais à un sous niveau (par exemple au niveau du diagramme), la modification ne s'appliquera qu'à ce qui sera au niveau choisi et récursivement aux sous niveaux. La valeur indiquée à un sur niveau est suivie dans les sous niveaux sous réserve que leur valeur soit *default*, ce qui est justement le cas par défaut à tous les niveaux sauf bien-sûr au niveau projet puisqu'il n'y a rien au dessus et qu'il faut bien finir par indiquer quoi faire.

Donc, si vous choisissez d'avoir des *use cases* bleus au niveau projet, tous les *use cases* deviennent bleus. Maintenant appelons le menu de *Withdraw* (dans le *browser* ou le diagramme, *via* un clic droit) et choisissons *edit drawing settings*, changeons la couleur qui était *default* (pour suivre la règle définie au niveau supérieur qui, lui-même ...) avec un vert, on obtient alors :

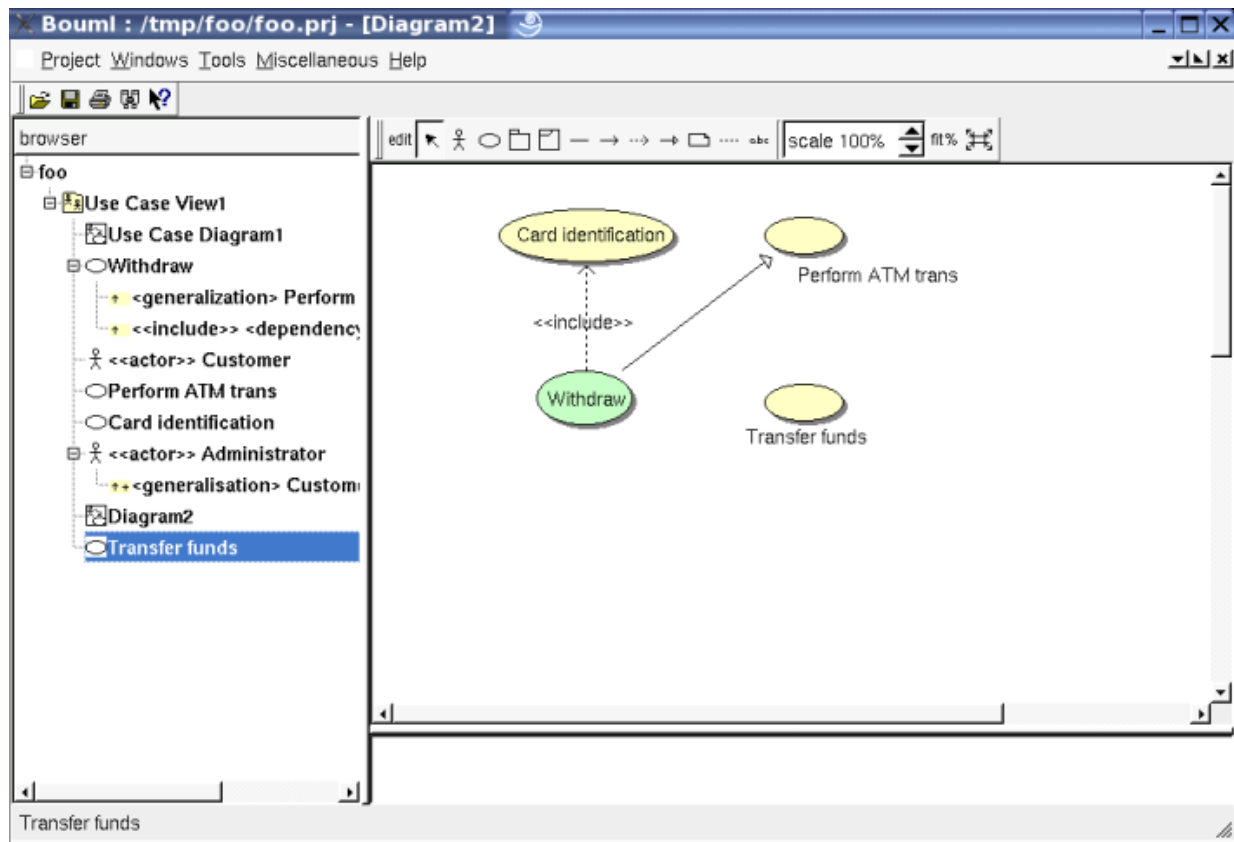


Maintenant, dans le *browser* appelez le menu du diagramme et choisissez *duplicate*, cela crée un clone du diagramme et provoque l'édition de ce clone vous permettant de choisir son nom. Validez l'édition et afficher le nouveau diagramme, celui-ci ressemble bien-sûr au diagramme d'origine, éditez ses *drawing settings* et demandez à avoir des *use case* jaunes, ce qui donne bien-sûr :

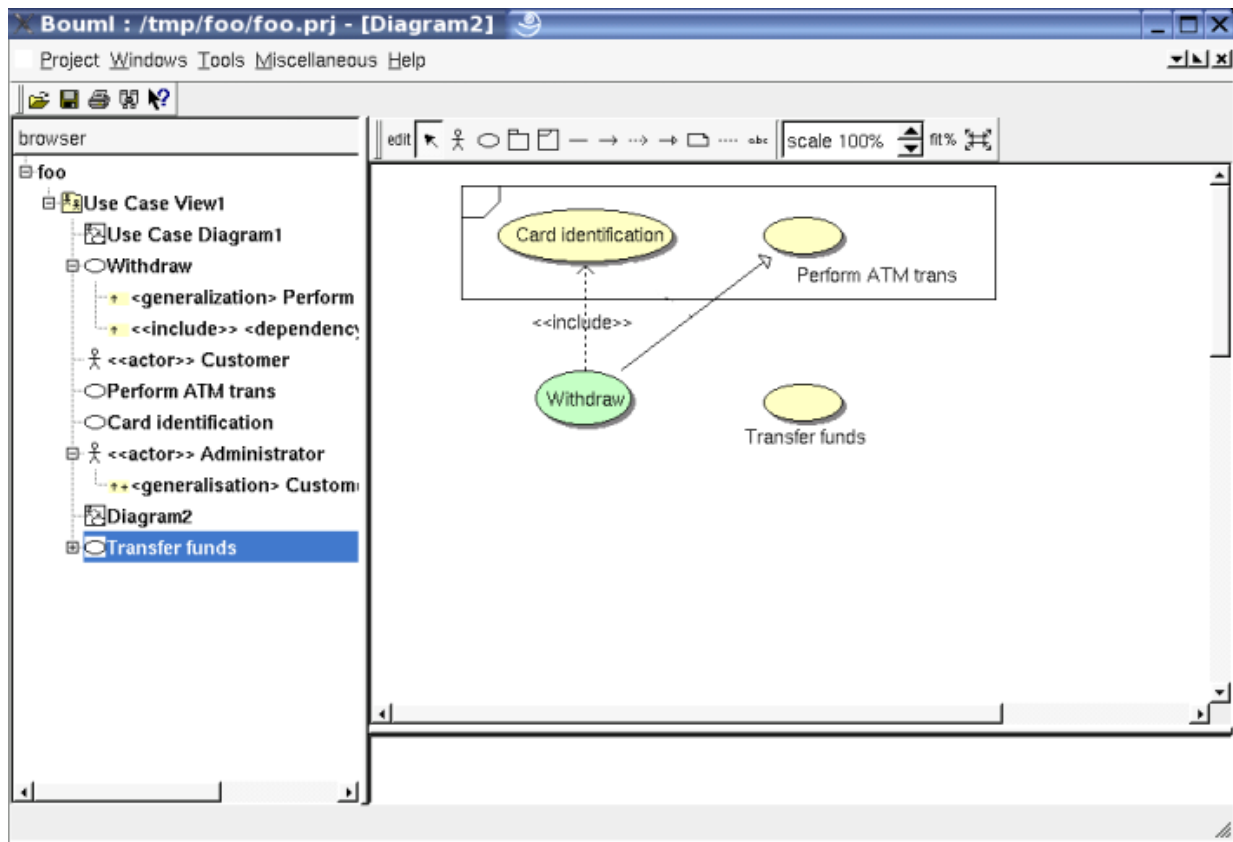


IV-B - Profondeur d'affichage

Ajoutez et retirez des éléments du second diagramme pour obtenir (le diagramme a de plus été *maximisé*) :



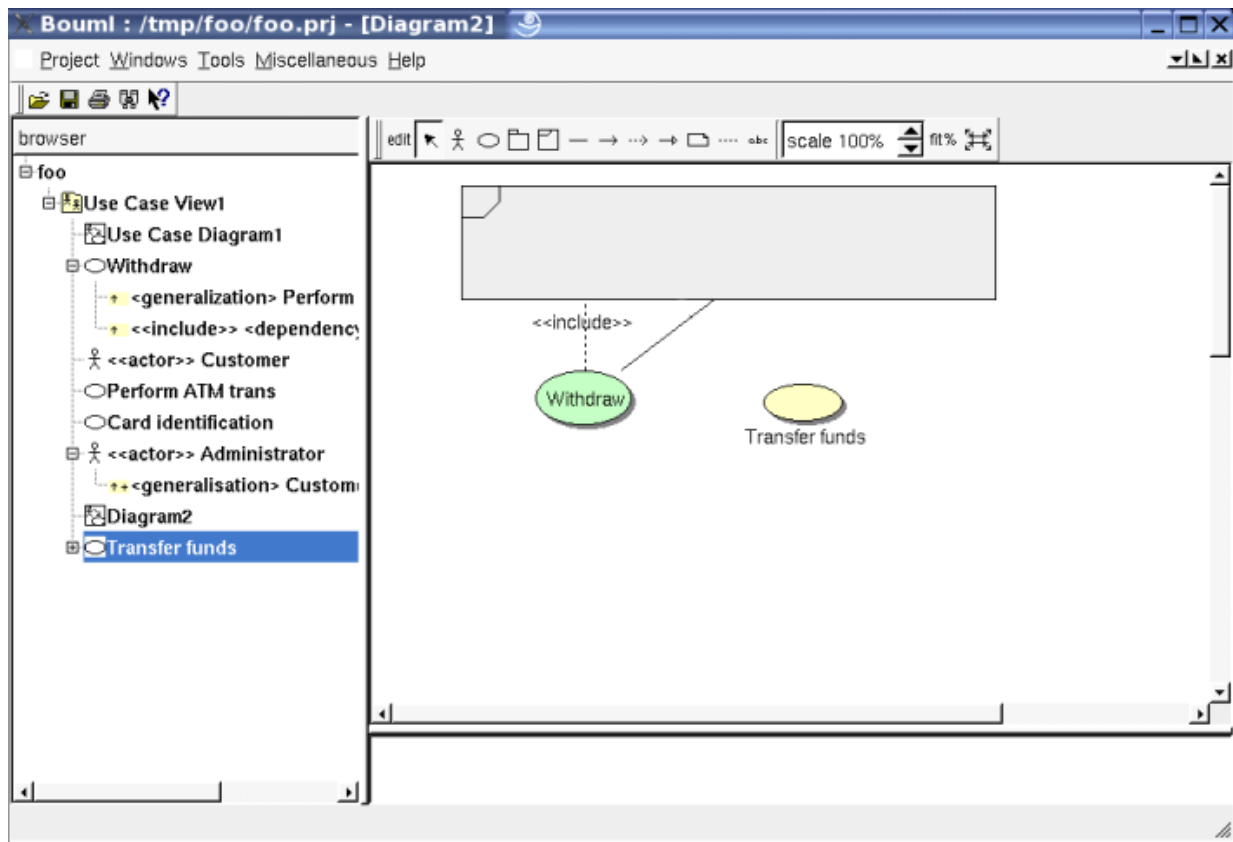
Ajoutez un *fragment* et redimensionnez-le pour avoir :



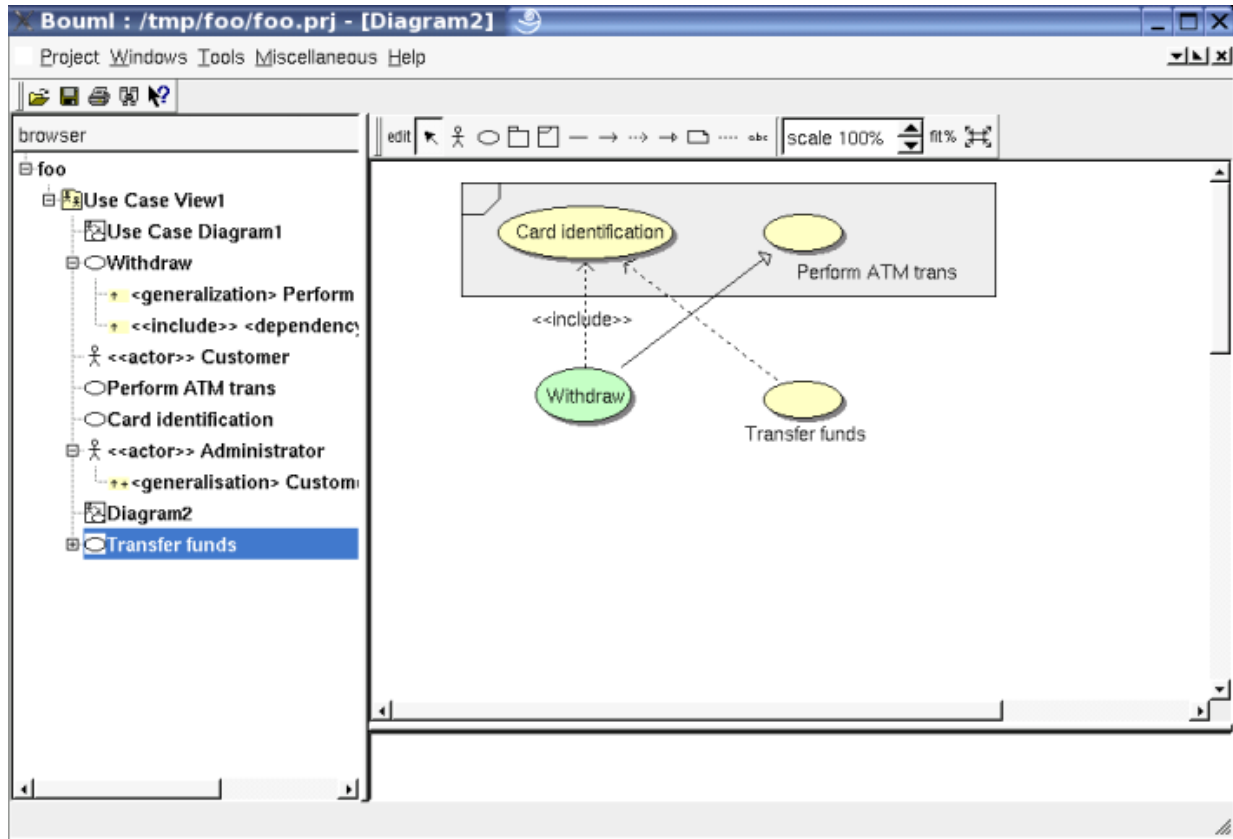
Maintenant essayez d'ajouter une dépendance de *Transfer funds* vers *Card identification* : BOUML refuse, que se passe-t-il ?

Parce que vous avez ajouté le *fragment* après *Card identification*, ce *fragment* est au dessus du *use case* et BOUML considère que vous essayez de faire une dépendance entre *Transfer funds* et le *fragment*, ce qui n'a pas de sens.

Si vous changez la couleur du *fragment* qui est transparent par défaut c'est encore plus visible :



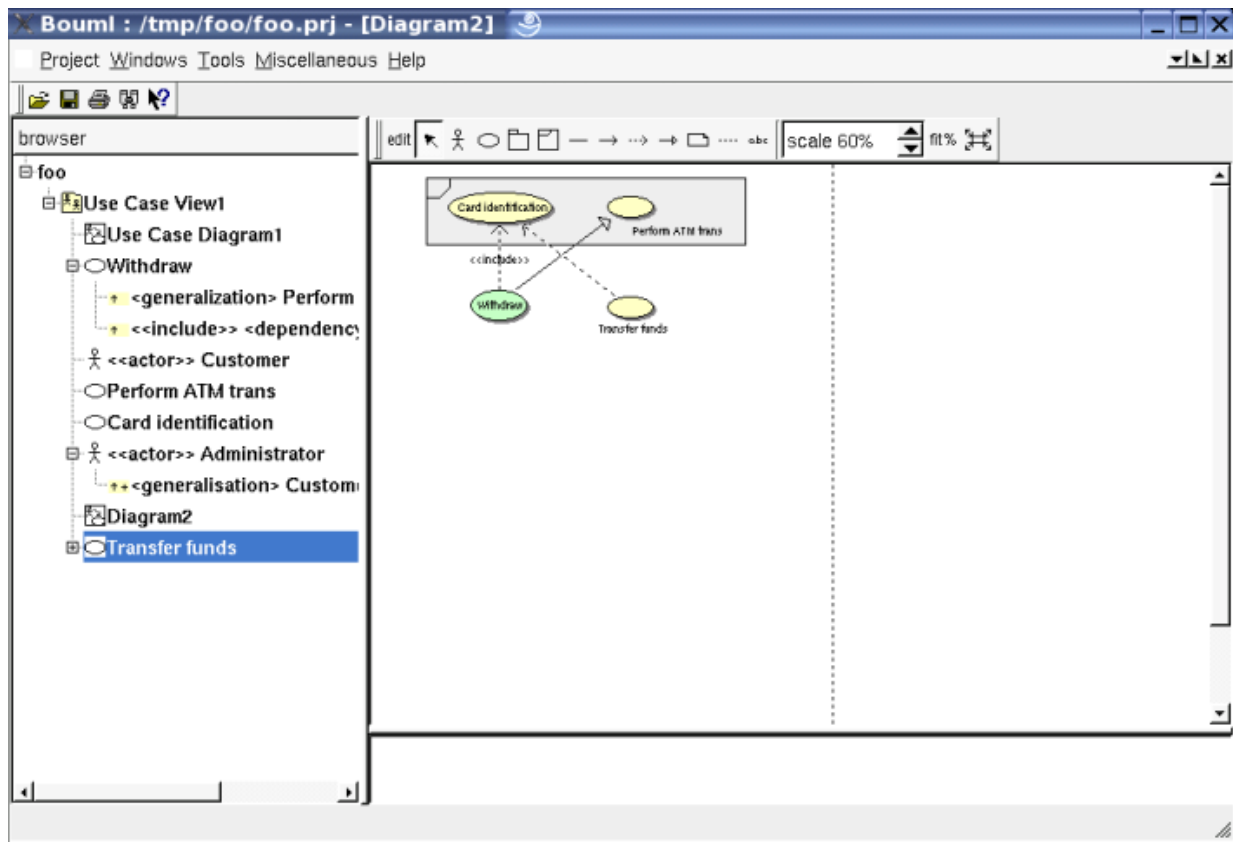
Pas de panique, contrairement à certains modeleurs vous n'avez pas tout à refaire en respectant le bon ordre, vous avez juste à appeler le menu du fragment et à choisir *lower*, maintenant vous pouvez ajouter la dépendance.



IV-C - Zoom, taille et format des diagrammes

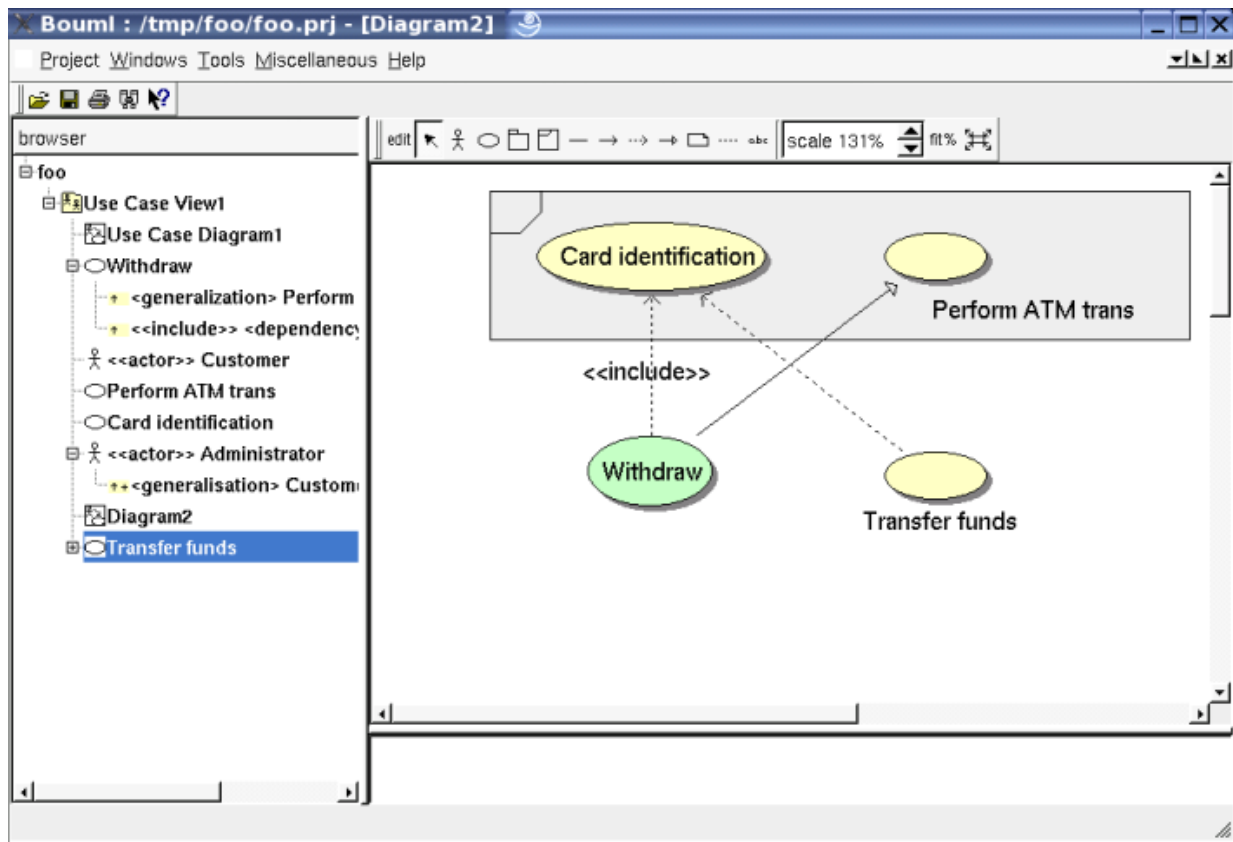
Quand vous ouvrez un diagramme le zoom est de 100% et la fenêtre d'affichage est de taille quelconque.

Quand vous changez le zoom en utilisant les deux petites flèches ou en tapant directement la valeur dans la *spin box* (ce qui est pratique vous revenir par exemple à 100% alors que le zoom courant est de 33%) la taille des flèches dans les diagrammes est inchangée, ce qui permet de les voir même avec un tout petit zoom, et inversement de ne pas prendre trop de place pour elles si le zoom est grand.

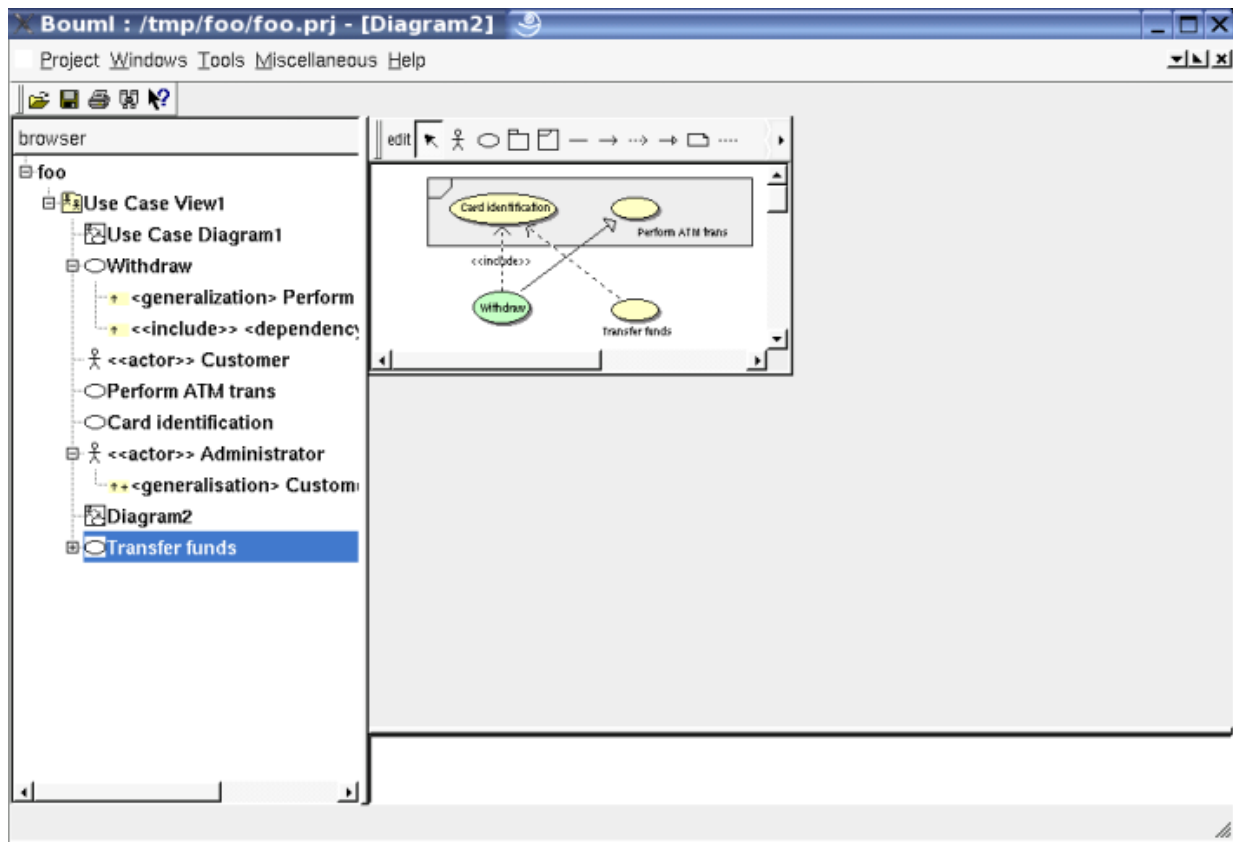


Les limites du diagramme sont représentées par deux lignes pointillées. Par défaut le format des diagrammes est le format européen *ISO A4*. Pour changer le format d'un diagramme, appelez son menu en faisant un clic droit dans le diagramme en dehors de tout élément et choisissez le format dans la liste proposée. A côté des formats européens en portrait et paysage vous avez le format 'A' qui correspond au format américain *lettre*, 'B' au format américain *large*, et 'C' qui est encore plus large. Pour éviter de changer le format après chaque création de diagramme, choisissez le format par défaut désiré via l'entrée *Diagram default format* du menu *Miscellaneous*.

fit% met le zoom à la plus grande valeur (limitée à 200%) compatible avec le fait qu'on puisse voir tous les éléments du diagramme étant donnée la taille de la sous fenêtre affichant le diagramme. La même chose peut être faite via l'entrée *optimal scale* du menu du diagramme.



A l'opposé, vous pouvez modifier automatiquement la taille de la sous fenêtre affichant le diagramme pour qu'elle soit la plus petite permettant de voir tous les éléments avec la valeur courante du zoom, via l'entrée *optimal window size* du menu du diagramme ou via le bouton à droite de *fit%*



Si vous fermez puis ré ouvrez le diagramme, le zoom redevient 100% et la taille de la sous fenêtre a peu de chance d'être ce qu'elle était à la fermeture. Pour sauver le zoom courant et la taille de la sous fenêtre afin de les retrouver à la prochaine ouverture du diagramme, choisissez *set preferred size and scale* dans le menu du diagramme.

IV-D - Exporter un diagramme

Il y a plusieurs façons d'exporter un diagramme sous forme d'une figure :

- En le copiant pour le copier dans un autre programme : si besoin ouvrez le diagramme pour le voir, choisissez l'entrée *copy visible picture part* dans le menu du diagramme. La partie visible d'un diagramme est ce qui apparaît en fonction de la taille de la sous fenêtre montrant le diagramme et de la position des *scrollbars*. Si vous voulez tout copier, y compris ce qui n'est pas visible, choisissez *copy optimal picture part*.
- En sauvant le dessin dans un fichier PNG : si besoin ouvrez le diagramme pour le voir, choisissez l'entrée *save visible picture part* dans le menu du diagramme. Si vous voulez tout sauver, y compris les parties non visibles étant donnée la taille courante de la sous fenêtre, choisissez *save optimal picture part*.
- En utilisant un *plug-out*, par exemple le générateur de document HTML. Dans le cas où un diagramme exporté n'est pas déjà ouvert, le zoom sera de 100% et l'équivalent d'un *optimal window size* sera effectué, sauf si vous avez fixé les règles *via set preferred size and scale*.
- En l'imprimant : si besoin ouvrez le diagramme pour le voir, et utilisez le bouton en forme d'imprimante en haut à gauche de la fenêtre principale de BOUML.

V - Conclusion

V-A - Epilogue

Nous voici au terme de ce premier tutoriel, sauvez votre projet car celui-ci sera réutilisé par le second tutoriel qui vous permettra de rentrer nettement plus dans le vif du sujet en définissant des classes et générant du code.

V-B - Liens

- Le site de Bouml : <http://bouml.free.fr>
- Le tutoriel *les classes et la génération de code sous Bouml* est [ici](#)
- Le tutoriel *Réalisation d'un plug-out de tri pour BOUML* est [ici](#)

Merci à [Yogui](#) et à [Nip](#) pour la relecture de ce tutoriel.

Bonnes modélisations !