Documentation du style `uml`

9 avril 2008

Fabien Dagnat[1]

Ce style permet de faciliter la réalisation de quelques uns des diagrammes d'UML. Les macros et environnements du styles sont configurables par des options que l'on donne sous la forme suivante :

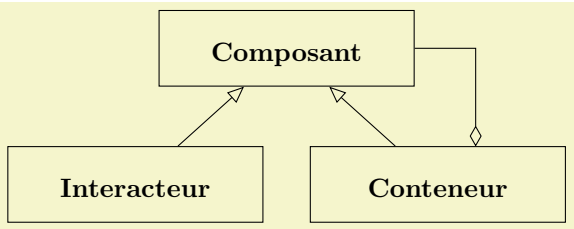
```
\commande[opt1,opt2=unevaleur]{...}

\begin{commande}[opt1,opt2=unevaleur]
  ...
\end{commande}
```

Chaque option sera présentée en indiquant son nom, son type, son but et sa (ou ses) valeurs par défauts.

1 Des exemples

1.1 Les diagrammes de classes

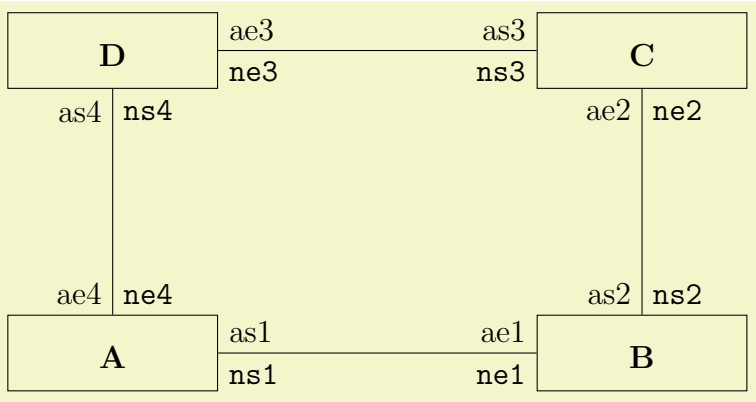
Un premier exemple de diagramme de classes qui introduit la notion de classe et d'héritage.



[1] fabien.dagnat@telecom-bretagne.eu

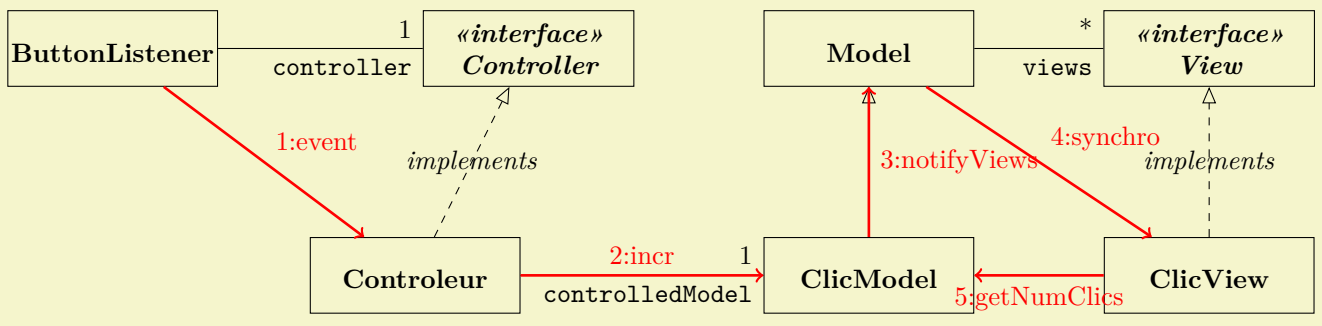
```
\begin{tikzpicture}[font=\footnotesize]
  \umlclass[name=Composant,label=Composant,width=2.6cm](4,2.8){}
  \umlclass[name=Conteneur,label=Conteneur,width=2.6cm](6,1){}
  \umlclass[name=Interacteur,label=Interacteur,width=2.6cm](2,1){}
  \umlinherit(Conteneur)(Composant)
  \umlinherit(Interacteur)(Composant)
  \draw[open diamond-] ([xshift=5mm]Conteneur.north) |- (Composant.east);
\end{tikzpicture}
```

Un autre exemple qui illustre le placement automatique des informations sur les associations qui ne peut être utilisé de manière optimale que pour des associations horizontale ou verticale. Pour les autres formes d'association si le placement ne convient vous devez revenir à du dessin à la main.




```
  \begin{tikzpicture}
    \umlclass[name=A,label=a](1,1){}
    \umlclass[name=B,label=b](8,1){}
    \umlclass[name=C,label=c](8,5){}
    \umlclass[name=D,label=d](1,5){}
    \umlasso[name=1,name end=ne1,arity end=ae1,name start=ns1,arity start=as1](a)(b)
    \umlasso[name=2,name end=ne2,arity end=ae2,name start=ns2,arity start=as2](b)(c)
    \umlasso[name=3,name end=ne3,arity end=ae3,name start=ns3,arity start=as3](c)(d)
    \umlasso[name=4,name end=ne4,arity end=ae4,name start=ns4,arity start=as4](d)(a)
  \end{tikzpicture}
```

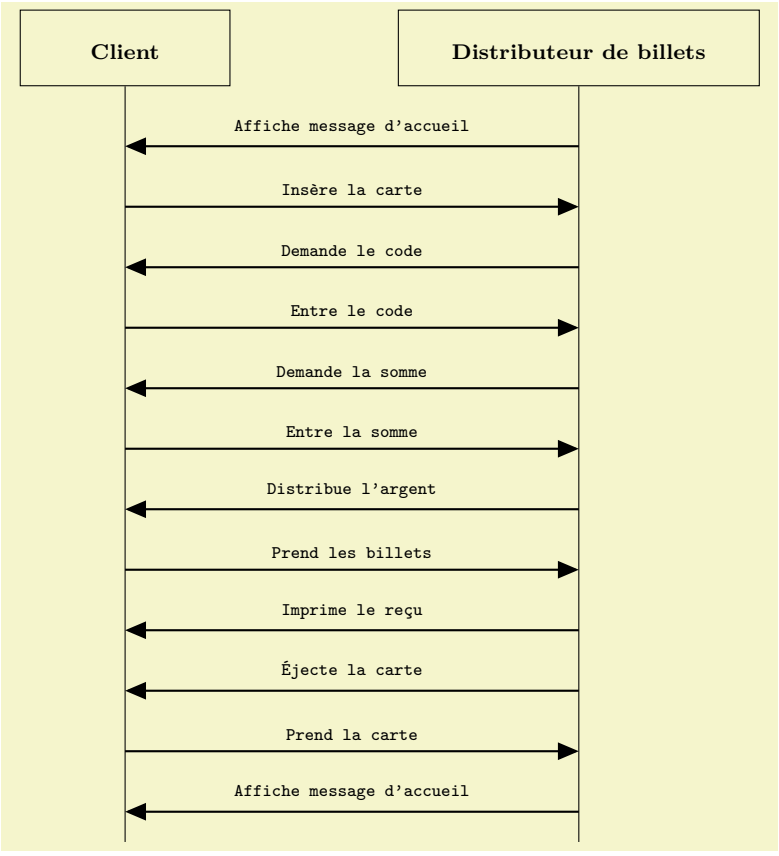
Aller un petit dernier pour la route...



```
\begin{tikzpicture}[font=\footnotesize,remember picture]
  \umlclass[name=Controller,label=c,interface](-4.5,6){}
  \umlclass[name=Model,label=m](0,6){}
  \umlclass[name=View,label=v,interface](4.5,6){}
  \umlclass[name=Controleur,label=ctlr](-6,3){}
  \umlclass[name=ClicModel,label=cm](0,3){}
  \umlclass[name=ClicView,label=cv](4.5,3){}
  \umlclass[name=ButtonListener,label=bl](-10,6){}
  \umlasso[arity end=*,name end=views](m)(v)
  \umlasso[arity end=1,name end=controller](bl)(c)
  \umlinherit(cm)(m)
  \umlimplem(cv)(v)
  \umlimplem(ctlr)(c)
  \umlasso[arity end=1,name end=controlledModel](ctlr)(cm)
  \begin{scope}[red,->,line width=1pt]
    \draw (bl) - node[above right] {1\string:event} (ctlr);
    \draw (ctlr) - node[above] {2\string:incr} (cm);
    \draw (cm) - node[right] {3\string:notifyViews} (m);
    \draw (m) - node[above right] {4\string:synchro} (cv);
    \draw (cv) - node[below] {5\string:getNumClics} (cm);
  \end{scope}
\end{tikzpicture}
```

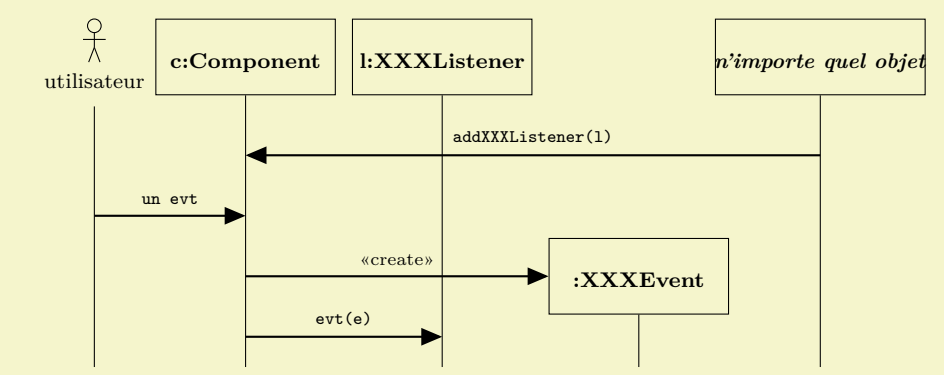
1.2 Les diagrammes de séquences

Un premier exemple basique pour l'introduction des commandes.

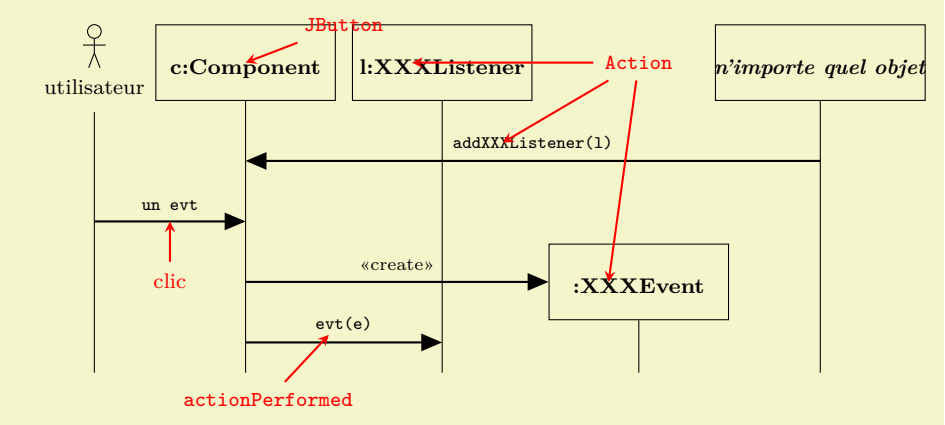


```
\begin{diagseq}[font=\scriptsize,step=8mm,base=5cm]
  \umldefclass[name={Client},label=client](2cm)
  \umldefclass[name={Distributeur de billets},width=4cm,label=dab](8cm)
  \umlsend(dab)(client){Affiche message d'accueil}
  \umlsend(client)(dab){Insère la carte}
  \umlsend(dab)(client){Demande le code}
  \umlsend(client)(dab){Entre le code}
  \umlsend(dab)(client){Demande la somme}
  \umlsend(client)(dab){Entre la somme}
  \umlsend(dab)(client){Distribue l'argent}
  \umlsend(client)(dab){Prend les billets}
  \umlsend(dab)(client){Imprime le reçu}
  \umlsend(dab)(client){Éjecte la carte}
  \umlsend(client)(dab){Prend la carte}
  \umlsend(dab)(client){Affiche message d'accueil}
  \umlclassend(client,dab)
\end{diagseq}
```

Un exemple un peu complet qui permet d'illustrer la création et le même ensuite avec des labels qui permet d'illustrer la création de nœuds réutilsable par la suite.

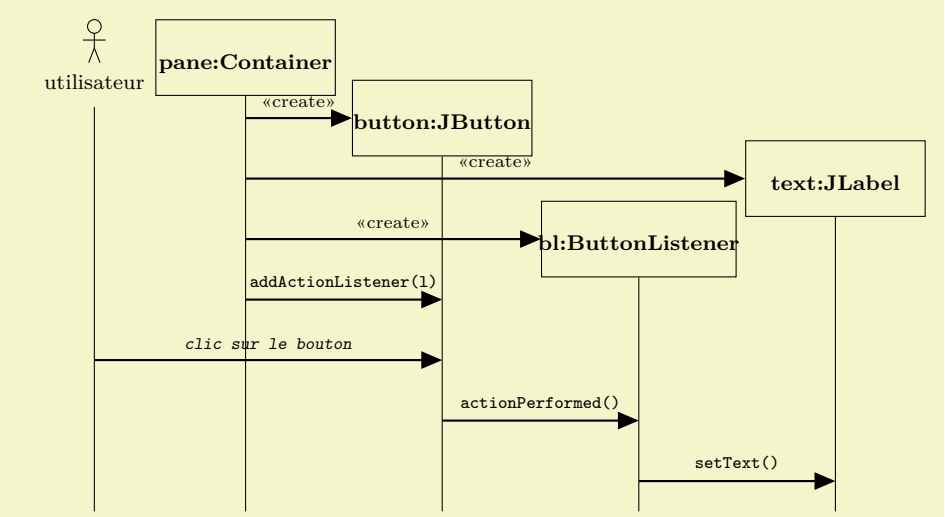



```
\begin{diagseq}[font=\scriptsize,step=8mm,base=5cm]
  \umlactor[scale=0.3](2,5){utilisateur}
  \umldefclass[width=1.6cm,name={c\string:Component},label=c](4cm)
  \umldefclass[width=1.6cm,name={l\string:XXXListener},label=listener](6.6cm)
  \umldefclass[width=2cm,name={\emph{n'importe quel objet}},label=source](11.6cm)
  \umlsend(source)(c){addXXXListener(l)}
  \umlsend(utilisateur)(c){un evt}
  \umlcreate[width=1.6cm,name={\string:XXXEvent},label=event](c)(9.2cm)
  \umlsend(c)(listener){evt(e)}
  \umlclassend(utilisateur,c,listener,event,source)
\end{diagseq}
```



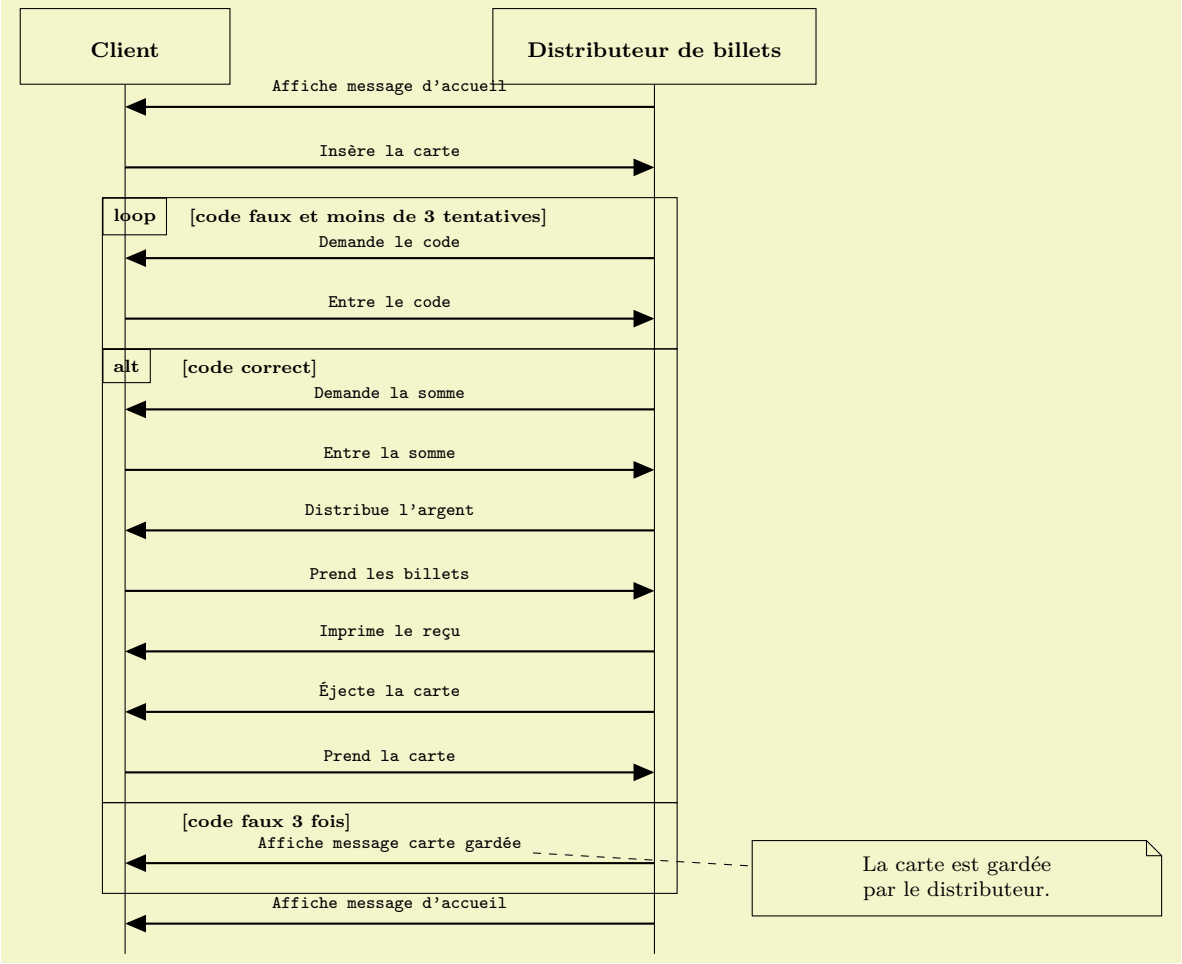

```
\begin{diagseq}[font=\scriptsize,step=8mm,base=5cm]
  \umlactor[scale=0.3](2,5){utilisateur}
  \umldefclass[width=1.6cm,name={c\string:Component},label=c](4cm)
  \umldefclass[width=1.6cm,name={l\string:XXXListener},label=listener](6.6cm)
  \umldefclass[width=2cm,name={\emph{n'importe quel objet}},label=source](11.6cm)
  \umlsend[label=addXXX](source)(c){addXXXListener(l)}
  \umlsend[label=evt](utilisateur)(c){un evt}
  \umlcreate[width=1.6cm,name={\string:XXXEvent},label=event](c)(9.2cm)
  \umlsend[label=funevt](c)(listener){evt(e)}
  \begin{scope}[line width=0.8pt,red]
    \draw(9.2,5) node (action) {\texttt{Action}};
    \draw[->](action) - ([xshift=-4mm]listener.center);
    \draw[->](action) - ([xshift=-4mm]event.center);
    \draw[->](action) - ([xshift=-4mm]addXXX.center);
    \draw(5.3,5.5) node (button) {\texttt{JButton}};
    \draw[->](button) - (c.center);
    \draw([yshift=-1cm]evt.center) node (clic) {clic};
    \draw[->](clic) - (evt);
    \draw([xshift=-1cm,yshift=-1cm]funevt.center) node (ap) {\texttt{actionPerformed}};
    \draw[->](ap) - ([xshift=-2mm,yshift=1mm]funevt.south);
  \end{scope}
  \umlclassend(utilisateur,c,listener,event,source)
\end{diagseq}
```

Un diagramme un peu plus réaliste.



```
\begin{diagseq}[font=\scriptsize,step=8mm,start=0mm,base=7cm]
  \umlactor[scale=0.3](2,7){utilisateur}
  \umldefclass[width=1.6cm,name={pane\string:Container},label=pane](4cm)
  \umlcreate[width=1.6cm,name={button\string:JButton},label=button](pane)(6.6cm)
  \umlcreate[width=1.6cm,name={text\string:JLabel},label=text](pane)(11.8cm)
  \umlcreate[width=1.8cm,name={bl\string:ButtonListener},label=listener](pane)(9.2cm)
  \umlsend(pane)(button){addActionListener(l)}
  \umlsend(utilisateur)(button){\textsl{clic sur le bouton}}
  \umlsend(button)(listener){actionPerformed()}
  \umlsend(listener)(text){setText()}
  \umlclassend(utilisateur,pane,button,text,listener)
\end{diagseq}
```

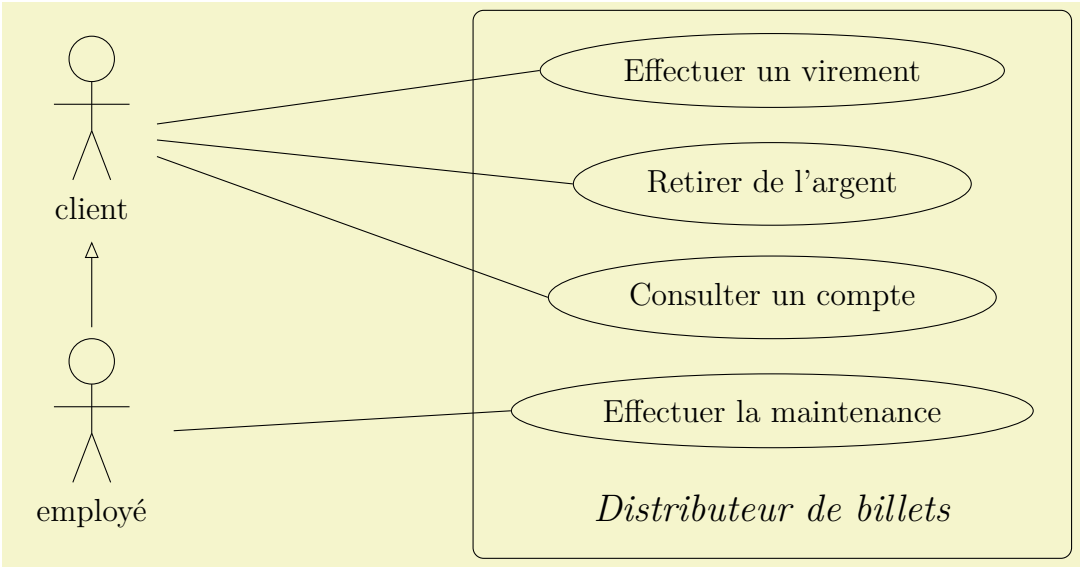
On peut également faire des boucles et des alternatives.



```
\begin{diagseq}[font=\scriptsize,step=8mm,start=0mm,base=5cm]
  \umldefclass[name={Client},label=client](2cm)
  \umldefclass[name={Distributeur de billets},width=3.5cm,label=dab](9cm)
  \umlsend(dab)(client){Affiche message d'accueil}
  \umlsend(client)(dab){Insère la carte}
  \umlloop{code faux et moins de 3 tentatives}(client)(dab){
    \umlsend(dab)(client){Demande le code}
    \umlsend(client)(dab){Entre le code}
  }
  \umlalt(client)(dab){code correct}{%
    \umlsend(dab)(client){Demande la somme}
    \umlsend(client)(dab){Entre la somme}
    \umlsend(dab)(client){Distribue l'argent}
    \umlsend(client)(dab){Prend les billets}
    \umlsend(dab)(client){Imprime le reçu}
    \umlsend(dab)(client){Éjecte la carte}
    \umlsend(client)(dab){Prend la carte}
  }{code faux 3 fois}{
    \umlsend[label=msg](dab)(client){Affiche message carte gardée}
  }
  \umlsend(dab)(client){Affiche message d'accueil}
  \umlclassend(client,dab)
  \umlnote[width=5cm,label=lanote](13,-6){La carte est gardée par le distributeur.}
  \draw[dashed] (msg) - (lanote);
\end{diagseq}
```

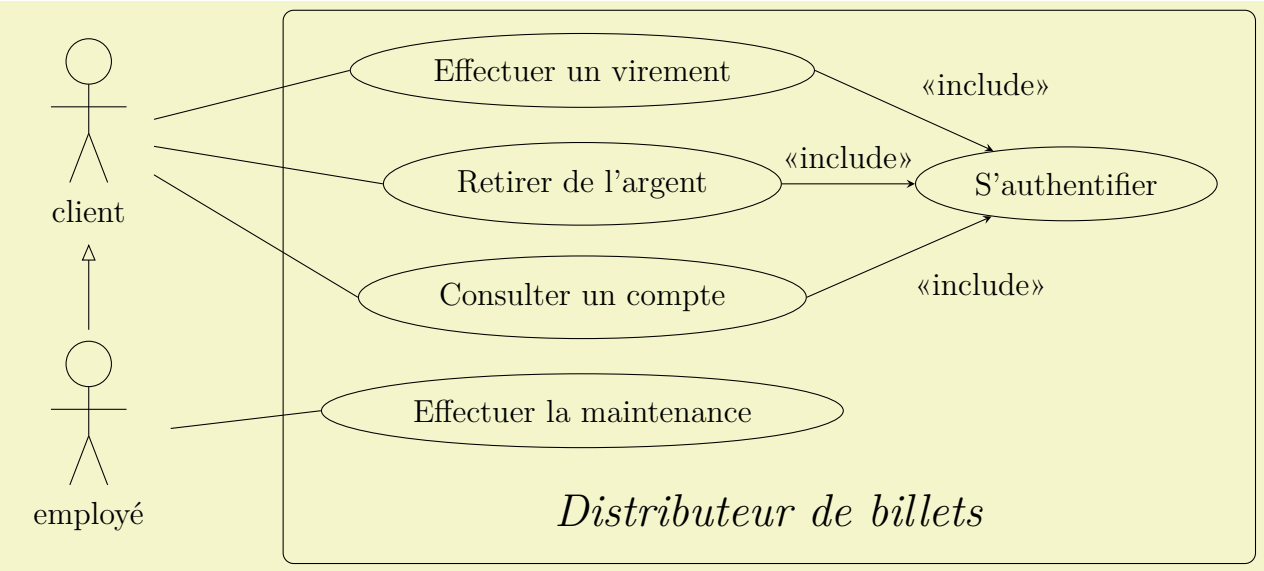
1.3 Les diagrammes de cas d'utilisation

Il est enfin possible de réaliser des diagrammes de cas d'utilisation. Attention, vu la technique utilisée deux compilations sont necéssaires pour que les liens soient corrects.



```
\begin{tikzpicture}[>=stealth,,remember picture=true]
  \umlactor(1,5.5){client}
  \umlactor(1,1.5){employé}
  \umlinherit(employé)(client)
  \umlsyst(10,3.5){\large Distributeur de billets}{
    \umlfunc[label=virement](0,6.5){Effectuer un virement}
    \umlfunc[label=retrait](0,5){Retirer de l'argent}
    \umlfunc[label=consult](0,3.5){Consulter un compte}
    \umlfunc[label=maint](0,2){Effectuer la maintenance}
  }
  \draw (client) - (virement.west);
  \draw (client) - (retrait.west);
  \draw (client) - (consult.west);
  \draw (employé) - (maint.west);
\end{tikzpicture}
```

Et pour finir un dernier diagramme de cas d'utilisation un peu plus complexe.




```
\begin{tikzpicture}[>=stealth,remember picture=true]
  \umlactor(1,5.5){client}
  \umlactor(1,1.5){employé}
  \umlinherit(employé)(client)
  \umlsyst(10,3.5){\Large Distributeur de billets}{
    \umlfunc[label=virement1](-0.2,6.5){Effectuer un virement}
    \umlfunc[label=retrait1](-0.2,5){Retirer de l'argent}
    \umlfunc[label=consult1](-0.2,3.5){Consulter un compte}
    \umlfunc[label=maint1](-0.2,2){Effectuer la maintenance}
    \umlfunc[label=authen](6.2,5){S'authentifier}
    \draw[<-] (authen) - node[pos=0.5,above right] {<<include>>} (virement1.east);
    \draw[<-] (authen) - node[pos=0.5,above] {<<include>>} (retrait1.east);
    \draw[<-] (authen) - node[pos=0.5,below right] {<<include>>} (consult1.east);
  }
  \draw (client) - (virement1.west);
  \draw (client) - (retrait1.west);
  \draw (client) - (consult1.west);
  \draw (employé) - (maint1.west);
\end{tikzpicture}
```

2 Modes et variables

2.1 L'environnement `diagseq`

L'environnement `diagseq` peut être configuré par les options suivantes :

Nom	Type	Intention	Valeur(s) par défaut
`step`	longueur	l'écart entre les messages	10mm
`start`	longueur	l'écart au départ	5mm
`base`	longueur	la ligne de base des objets pré-existant	0mm
`font`	chaîne	font de la figure	`\normalsize`