Modélisation dynamique

Exercice 1 : Diagramme d'états simple

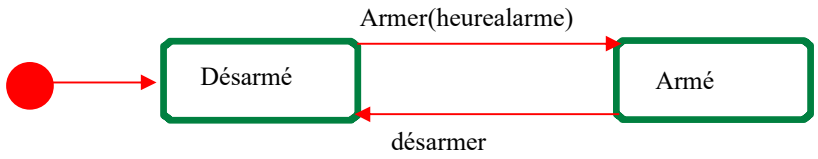
Considérons un réveille-matin simplifié :

- on peut mettre l'alarme « on » ou « off » ;
- quand l'heure courante devient égale à l'heure d'alarme, le réveil sonne sans s'arrêter ;
- on peut interrompre la sonnerie.

Question 1 : Dessinez le diagramme d'états correspondant.

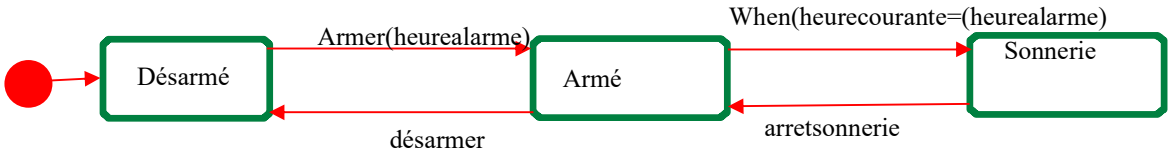
Solution

Voyons tout d'abord la première phrase :
1. On peut mettre l'alarme « on » ou « off ».
Le réveil a clairement deux états distincts : *Désarmé* (alarme « off ») ou *Armé* (alarme « on »). Une action de l'utilisateur permet de passer d'un état à l'autre. On suppose que le réveil est bien désarmé au départ. Notez le paramètre *heureAlarme* de l'événement *armer*.



Considérons maintenant les deux autres phrases :
2. Quand l'heure courante devient égale à l'heure d'alarme, le réveil sonne sans s'arrêter ;
3. On peut interrompre la sonnerie.
Le fait de sonner constitue un nouvel état pour le réveil. Il s'agit bien d'une période de temps durant laquelle le réveil effectue une certaine activité (sonner) qui dure jusqu'à ce qu'un événement vienne l'interrompre.

Le passage de l'état *Armé* à l'état *Sonnerie* est déclenché par une transition due à un changement interne, représenté au moyen du mot-clé « when ». En revanche, d'après l'énoncé, le retour de l'état *Sonnerie* à l'état *Armé* ne s'effectue que sur un événement utilisateur.



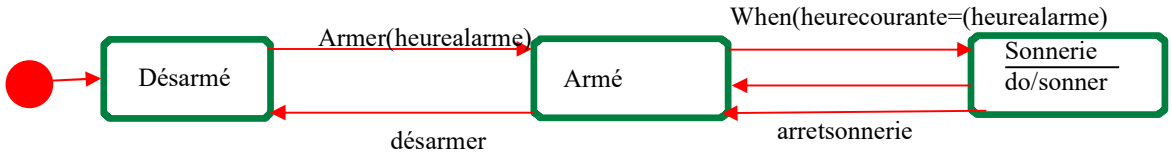
Question 2 : Complétez le diagramme d'états précédent pour prendre en compte le fait que la sonnerie du réveil s'arrête d'elle-même au bout d'un certain temps.

Solution

ACTIVITÉ CONTINUE OU FINIE – TRANSITION AUTOMATIQUE

Une activité à l'intérieur d'un état peut être soit :
– « continue » : elle ne cesse que lorsque se produit un événement qui fait sortir l'objet de l'état ;
– « finie » : elle peut également être interrompue par un événement, mais elle cesse de toute façon d'elle-même au bout d'un certain temps, ou quand une certaine condition est remplie.
La transition de complétion d'une activité finie, aussi appelée transition automatique, est représentée en UML sans nom d'événement ni motclé.

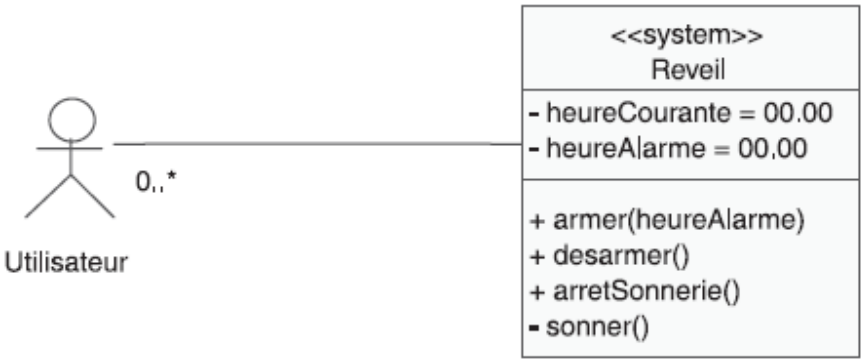
Dans notre exemple, il suf.t donc d'ajouter une activité *sonner* à l'état *Sonnerie* et une transition automatique en sortie de cet état. Le diagramme d'états complété est représenté sur le schéma suivant.



Il convient aussi de se demander si l'utilisateur a le droit de désarmer le réveil pendant qu'il sonne. Dans ce cas, il faudrait ajouter une transition déclenchée par *désarmer* et allant directement de *Sonnerie* à *Désarmé* .

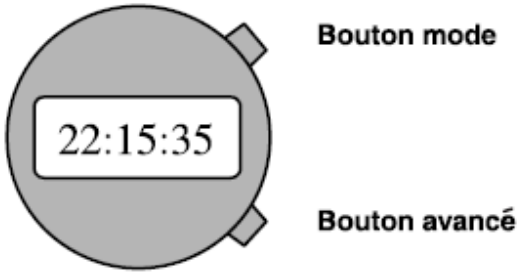
Question 3 : **Déduisez-en le diagramme de contexte statique étendu du réveil**
.

Solution



Problème 2 :

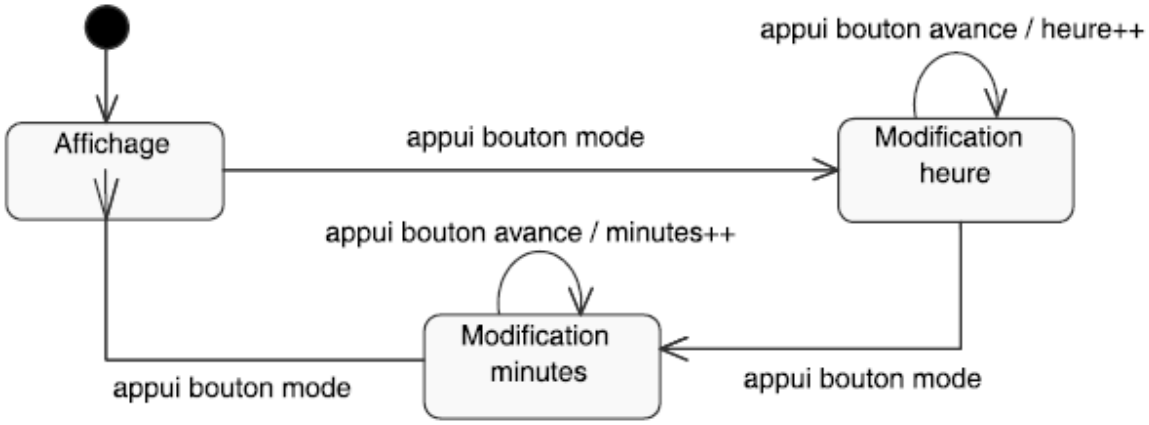
Considérons une montre à cadran numérique simplifiée :



1. Le mode courant est le mode « Affichage ».
2. Quand on appuie une fois sur le bouton mode, la montre passe en « modification heure ». Chaque pression sur le bouton avance incrémente l'heure d'une unité.
3. Quand on appuie une nouvelle fois sur le bouton mode, la montre passe en « modification minute ». Chaque pression sur le bouton avance incrémente les minutes d'une unité.
4. Quand on appuie une nouvelle fois sur le bouton mode, la montre repasse en mode « Affichage ».

Question1 : **Dessinez le diagramme d'états correspondant.**

Solution :
On obtient sans difficulté particulière ce diagramme d'états typique, qui est présenté sur le schéma suivant.

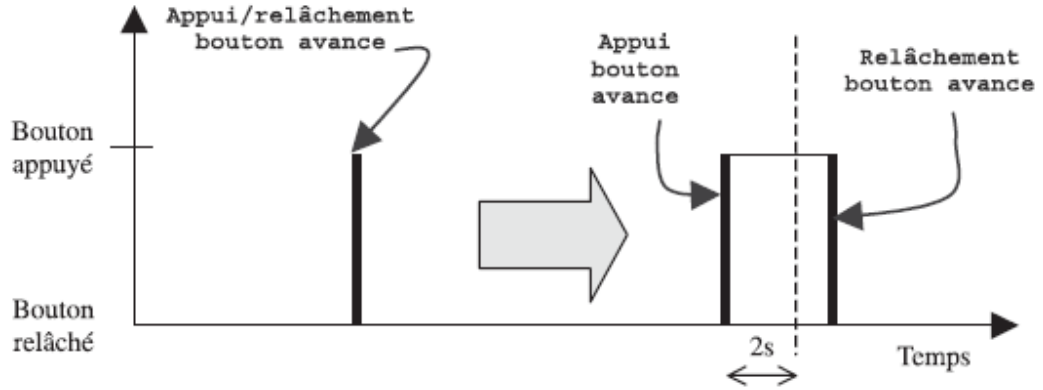


On remarquera les notations en style C++ ou Java pour les actions : « heure++ » et « minutes++ ». UML ne propose pas encore de langage d'action, nous pouvons donc exprimer le détail des actions comme nous le souhaitons : texte libre, pseudo-code, etc.
Nous obtenons des transitions propres sur les états de modification et pas sur l'état d'affichage. Cela veut-il dire que l'événement « appui bouton avance » est impossible dans l'état « Affichage » ? Non, bien sûr. Cela signifie plutôt que, comme cet événement n'a aucun effet dans cet état, il ne déclenche aucune transition. L'événement est purement et simplement perdu.

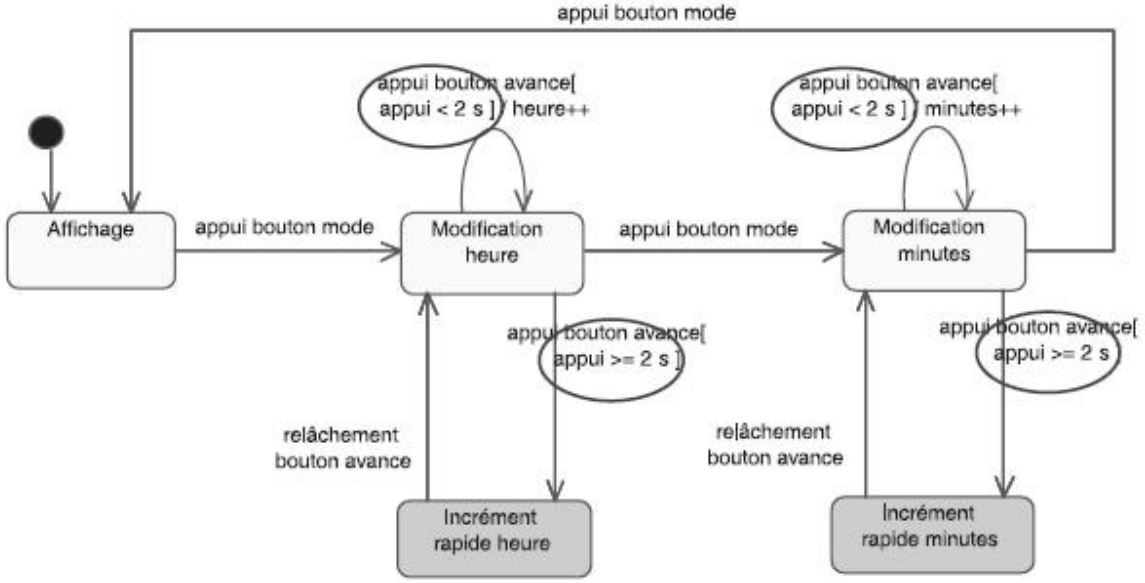
Question2 : Événement temporel

Ajoutez le comportement suivant : quand on appuie sur le bouton avance plus de deux secondes, les heures (ou les minutes) s'incrémentent rapidement jusqu'à ce qu'il se produise un relâchement dans la pression du bouton.
Envisagez plusieurs solutions possibles.

Solution :
Dans l'exemple précédent, les événements d'appui sur les boutons correspondaient en fait au couple indivisible « pression » et « relâchement ». Nous avions considéré que la durée de pression sur chaque bouton était négligeable par rapport aux durées des états ou, en tout cas, non significative. Avec le nouvel énoncé, ce n'est plus le cas, puisque la durée de pression sur le bouton avance influe sur le comportement de la montre. La bonne approche consiste à introduire un nouvel événement : « relâchement bouton avance », afin de pouvoir gérer le temps de pression.



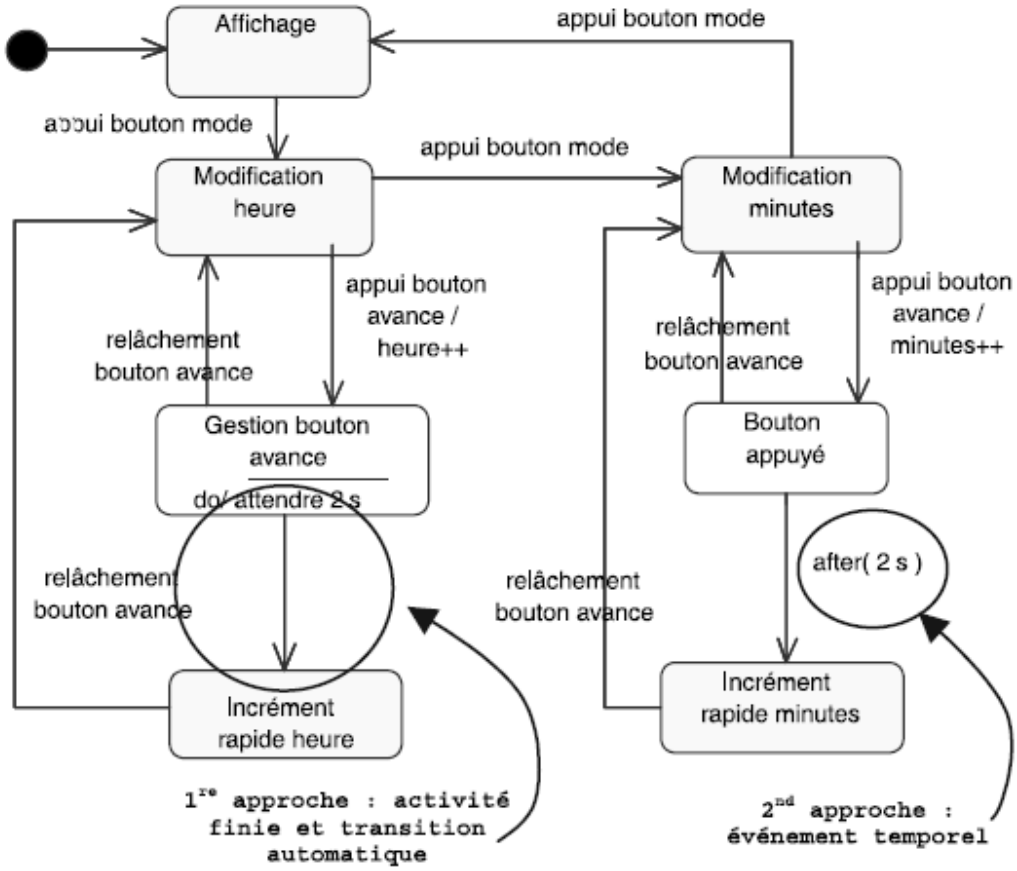
Une première solution, tentante, consiste à introduire une condition sur la durée de pression, ainsi qu'un nouvel état « Incrémentation rapide », comme cela est illustré sur la figure suivante :



Pourtant, cette solution qui semble évidente n'est pas acceptable en UML.
En effet, un événement (comme une transition et une action) est par convention instantané, ou en tout cas insécable (atomique). Il est donc tout à fait inapproprié de tester sa durée ! Les seuls concepts dynamiques en UML pour lesquels la notion de durée est significative sont l'état et l'activité. Il faut donc s'en servir pour résoudre cet exercice. Deux solutions sont possibles : toutes les deux nécessitent que l'on ajoute un

état intermédiaire pour que l'on puisse tester la durée de pression sur le bouton avance, mais elles diffèrent quant à la façon de réaliser ce test :
• La première approche consiste à introduire une activité finie « attendre 2 s » dans l'état intermédiaire et une transition automatique qui représente le fait que le bouton est appuyé plus de deux secondes.
• La seconde approche consiste à utiliser un autre mot-clé proposé par UML : le pseudo-événement « after », suivi d'une durée en argument représentant l'échéance d'un timer.

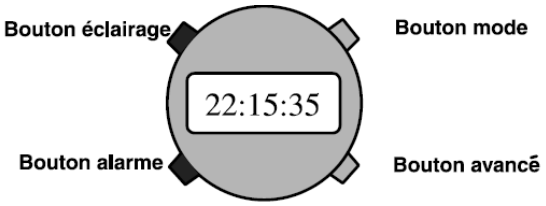
Pour illustrer les deux solutions, nous les avons représentées ensemble sur le diagramme suivant mais, dans la réalité, il faudrait bien sûr en choisir une seule et l'appliquer aux deux états de modification. Pour notre part, nous recommandons la seconde solution qui nous semble plus simple et plus facile à lire.



On notera que le comportement initial est conservé : si le bouton avance est relâché en moins de deux secondes, les heures (ou les minutes) sont bien incrémentées d'une unité. En fait, la transition propre qui existait sur chaque état de modification a pu être coupée en deux suite à la séparation des deux événements « appui » et « relâche », et à l'ajout de l'état intermédiaire.

Question3 : **Dessinez le diagramme d'états complet incluant tous les comportements de la montre.**

Reprenons notre exemple de montre à cadran numérique tel qu'il était présenté au début de l'exercice, et ajoutons maintenant à cette dernière deux autres boutons :
• un bouton éclairage ; en le pressant, on éclaire le cadran de la montre, jusqu'à ce qu'on le relâche ;
• un bouton alarme, qui ajoute à la montre digitale une fonctionnalité classique d'alarme, comme cela a été décrit lors du premier exercice (réveille-matin).

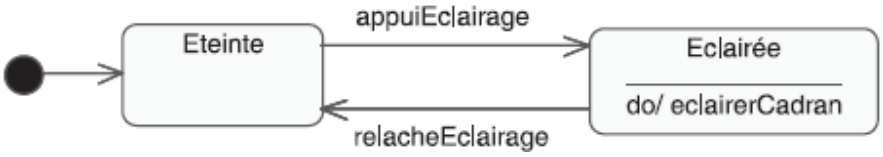


solution

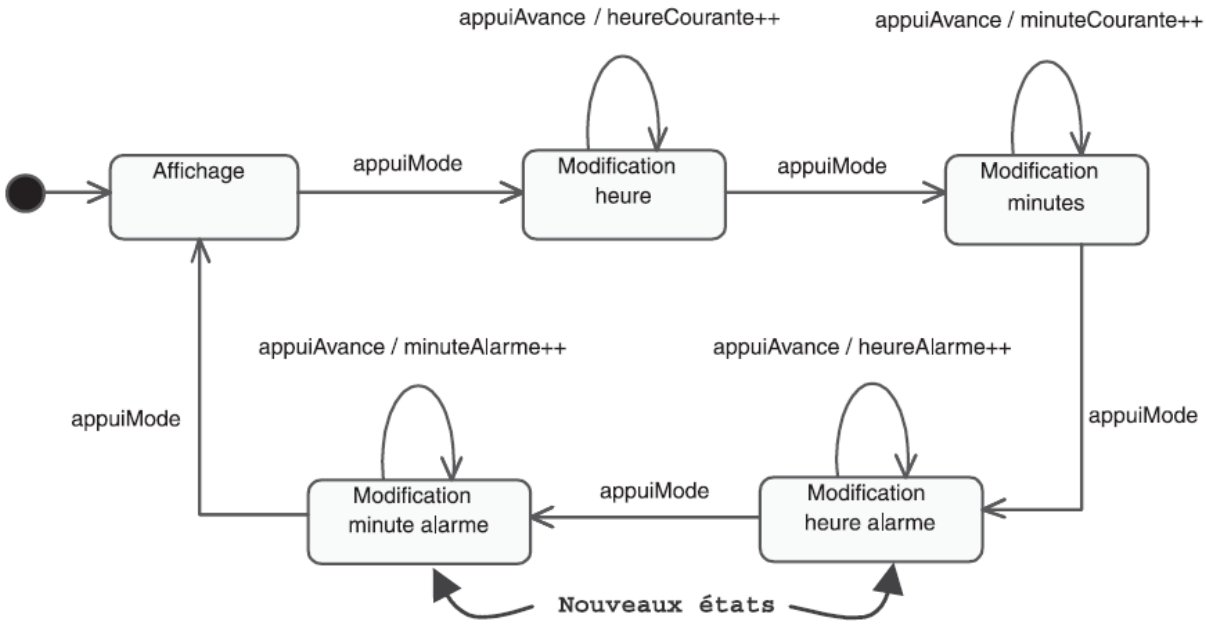
Nous sommes clairement en présence de trois comportements concurrents :
• la gestion de l'affichage ;
• la gestion de l'alarme ;
• la gestion de l'éclairage.

Commençons par le plus simple d'entre eux, qui concerne la gestion de l'éclairage. Cette dernière peut se modéliser très simplement par un automate à deux états, comme nous l'illustrons sur le schéma suivant.

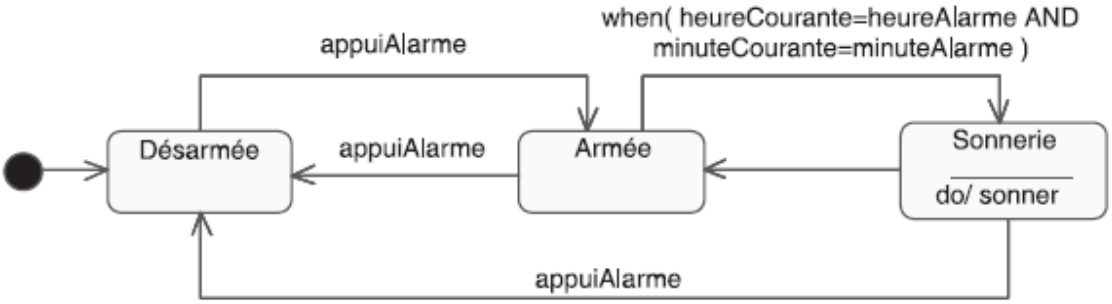
Diagramme d'états de la gestion de l'éclairage



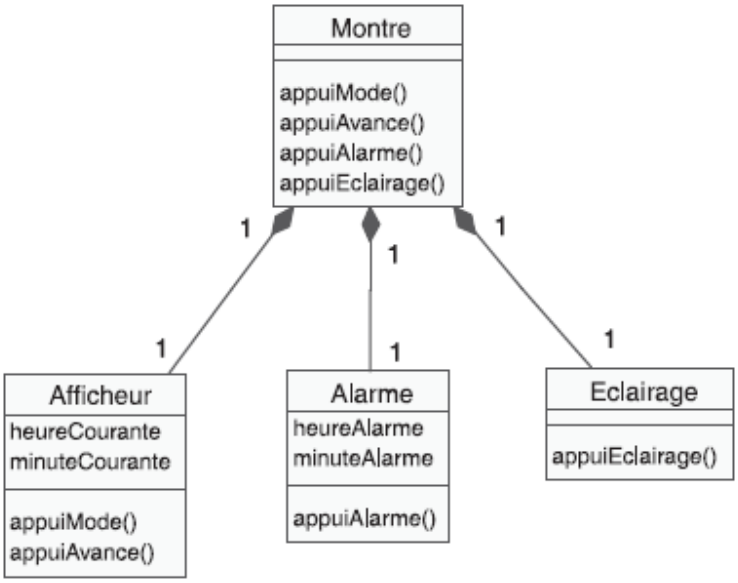
Si la gestion de l'éclairage peut tout à fait se modéliser à part, il n'en est pas de même pour l'affichage et l'alarme. En effet, il faut maintenant pouvoir aussi modifier l'heure d'alarme et la minute d'alarme, ce qui ajoute deux nouveaux états au diagramme de la figure 6-6, comme cela est indiqué ci-après.



Il reste maintenant à modéliser la gestion de l'alarme. Nous pouvons nous inspirer du diagramme d'états du réveil pour obtenir le schéma suivant. On y notera la dépendance avec la gestion de l'affichage *via* le test effectué par la gestion de l'alarme sur les attributs (« when »…).

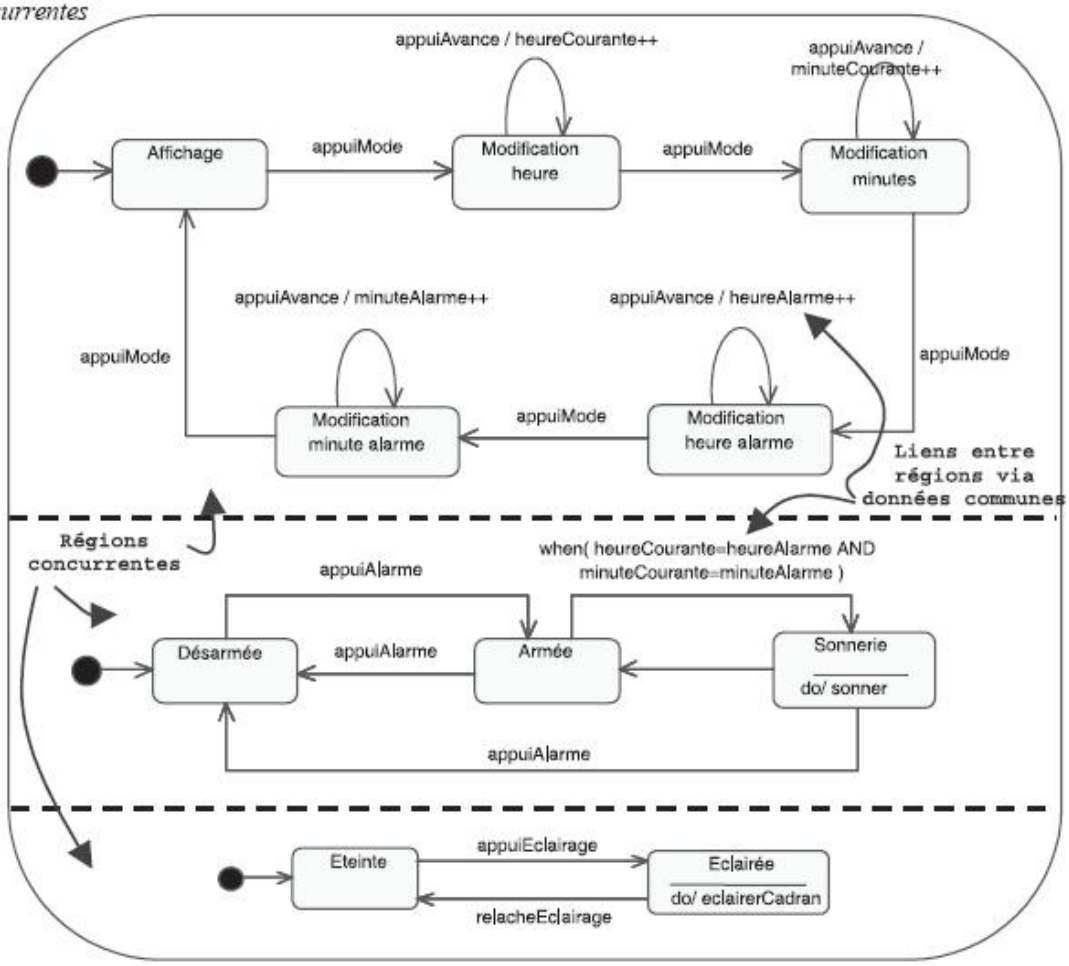


Nous avons donc obtenu trois diagrammes d'états. Comment faire en sorte que ces trois diagrammes séparés décrivent le comportement de la montre à cadran numérique ?
Là encore, deux solutions sont possibles :
• Considérer que toute instance de *Montre* contient en fait trois instances et que chacune gère un des trois comportements décrits précédemment. Toute montre délègue ainsi une partie de sa dynamique à une instance d'afficheur, d'éclairage ou d'alarme, suivant le cas. On peut représenter cela au moyen d'une relation de composition dans un diagramme de classes.



• Décrire des « régions concurrentes » au sein du diagramme d'états de la classe *Montre* . C'est une solution moins usitée que la précédente (principalement car certains outils UML ne la proposent pas), mais tout aussi légale. L'état courant de la montre devient alors un vecteur à trois lignes : état de l'affichage, état de l'alarme, état de l'éclairage. Une montre peut simultanément être en affichage de modification minute, être en train de sonner et être éclairée. Le diagramme d'états de la montre serait alors tel qu'il apparaît sur le schéma ci-après.

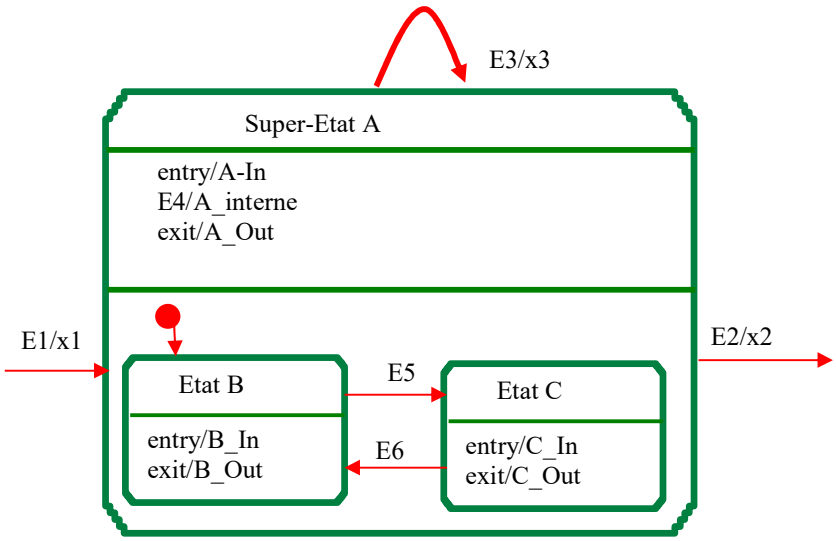
Diagramme d'états de la montre avec régions concurrentes



On notera que chaque « région » doit être initialisée puisque, si les états sont bien exclusifs à l'intérieur d'une région concurrente, ils existent simultanément dans les trois régions.

Exercice 2 : Diagramme d'états hiérarchique

Considérons le fragment de diagramme d'états suivant, qui contient de nombreuses actions.



Nota : ACTION D'ENTRÉE (OU DE SORTIE) – ENTRY (EXIT)
Une action d'entrée (introduite par le mot-clé « entry » à l'intérieur du symbole d'un état) représente une action qui est exécutée chaque fois que l'on entre dans cet état. Cela permet de factoriser une même action qui sera déclenchée par toutes les transitions qui entrent dans l'état. L'action de sortie (introduite par le mot-clé « exit ») est l'action symétrique en sortie de l'état.

Le diagramme de l'énoncé comprend donc :
• une transition propre sur le super-état (E3/x3) ;
• une transition interne dans le super-état (E4/A_Interne) ;
• des transitions d'entrée et de sortie dans le super-état et chacun des sous-états.
Nous allons étudier l'ordre temporel d'exécution des actions en complétant le tableau suivant. Nous partirons de l'état à gauche du diagramme symbolisé par « … », et nous prendrons comme nouvel état de départ l'état d'arrivée de la ligne précédente.

État de départ	Événement	Actions	État d'arrivée
	E1		
	E5		
	E4		
	E6		
	E3		
	E5		
	E3		
	E2		

Question : **Complétez le tableau précédent.**

Solution
Dans l'état de départ, symbolisé par « … » à gauche du diagramme, l'événement E1 déclenche l'action x1, puis conduit au super-état A. Cette entrée dans le super-état A déclenche l'action d'entrée A_In, puis l'entrée dans le sous-état B (à cause du symbole du sous-état initial), et donc l'action d'entrée B_In.

État de départ	Événement	Actions	État d'arrivée
…	E1	x1, A_In, B_In	B (dans A)

Dans l'état B, l'événement E5 fait sortir de l'état et déclenche donc l'action B_Out, puis conduit à l'état C et déclenche en conséquence l'action C_In.

État de départ	Événement	Actions	État d'arrivée
B	E5	B_Out, C_In	C (dans A)

Dans l'état C, l'événement E4 est-il possible ? Oui, car les transitions internes sont héritées du super-état. L'événement E4 fait donc rester dans l'état C et déclenche simplement l'action A_Interne.

État de départ	Événement	Actions	État d'arrivée
C	E4	A_Interne	C (dans A)

Dans l'état C, l'événement E6 fait sortir de l'état et déclenche donc l'action C_Out, puis conduit à l'état B et déclenche en conséquence l'action B_In.

État de départ	Événement	Actions	État d'arrivée
C	E6	C_Out, B_In	B (dans A)

Dans l'état B, l'événement E3 est-il possible ? Oui, car les transitions propres sont héritées du super-état. L'événement E3 fait d'abord sortir de l'état B et déclenche l'action B_Out, puis fait sortir du super-état A et déclenche A_Out, déclenche ensuite l'action x3, puis fait entrer dans le super-état A et déclenche A_In, enfin fait rentrer dans l'état B et déclenche l'action B_In.

État de départ	Événement	Actions	État d'arrivée
B	E3	B_Out, A_Out,x3,A_In,B_In	B (dans A)

Nous avons déjà considéré l'arrivée de E5 dans l'état B :

État de départ	Événement	Actions	État d'arrivée
B	E5	B_Out, C_In	C (dans A)

Attention, il y a un piège ! Dans l'état C, l'événement E3 fait d'abord sortir de l'état C et déclenche l'action C_Out, puis fait sortir du super-état A et déclenche A_Out, déclenche ensuite l'action x3, puis fait entrer dans le super-état A et déclenche A_In, en.n fait rentrer dans l'état B (car c'est le sous-état initial !) et déclenche l'action B_In.

État de départ	Événement	Actions	État d'arrivée
C	E3	C_Out, A_Out,x3,A_In,B_In	B (dans A)

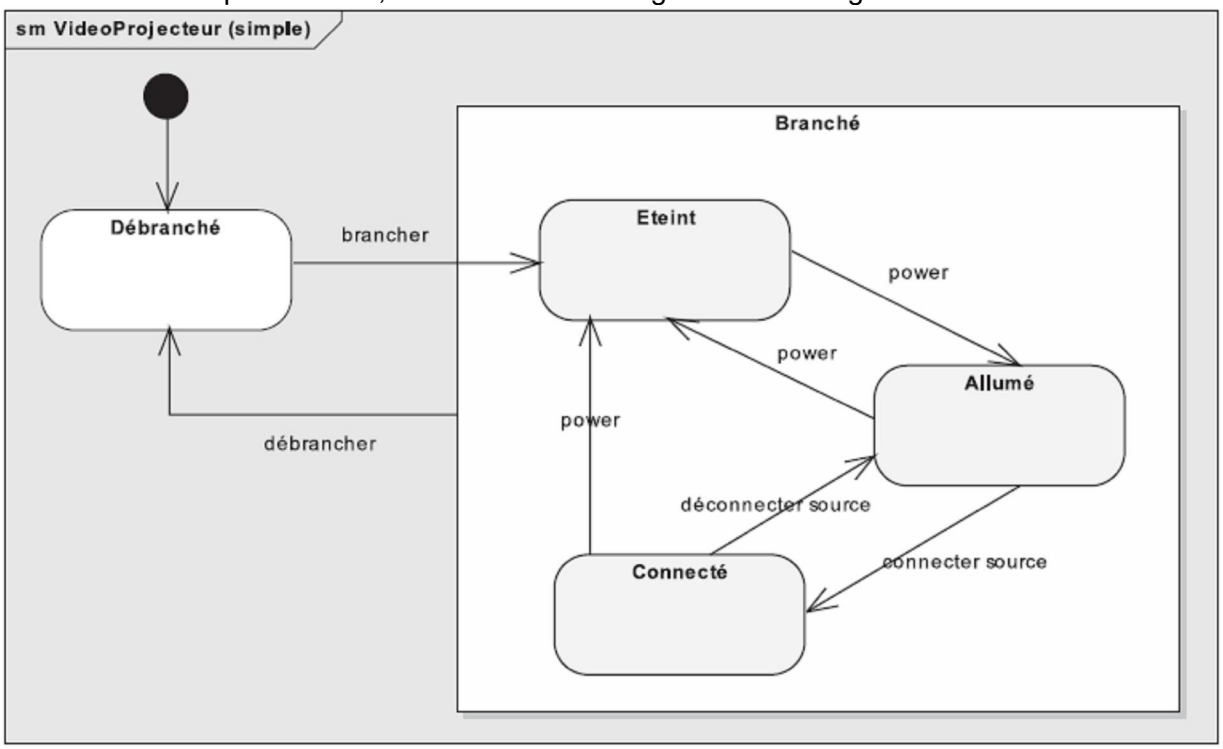
Dans l'état B, l'événement E2 fait d'abord sortir de l'état B et déclenche l'action B_Out, puis du super-état A et déclenche A_Out, et déclenche enfin l'action x2.

État de départ	Événement	Actions	État d'arrivée
B	E2	B_Out, A_Out,x2	…

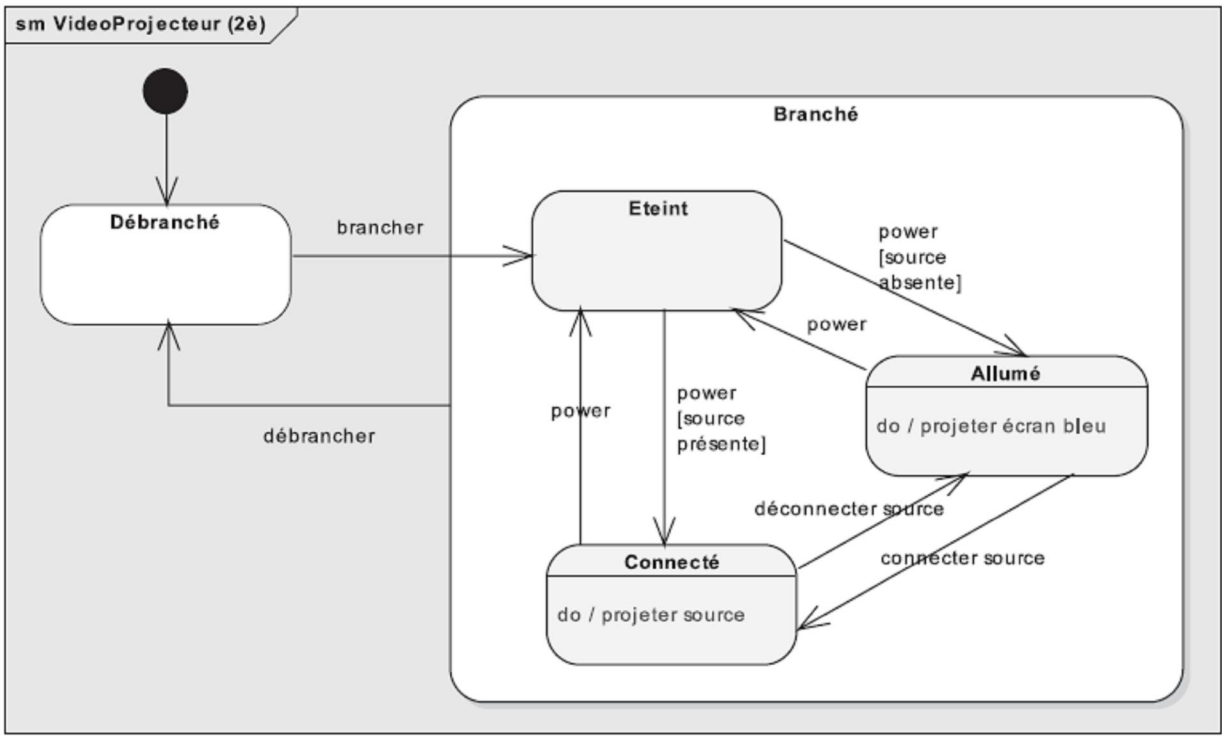
Exercice 3 : Modélisation

Modélisez le comportement du vidéoprojecteur lors d'une session de formation. Commencez par identifier les états et transitions « nominaux ». Ajoutez les périodes de préchauffage et de refroidissement de la lampe. Représentez ensuite le fait qu'il faut appuyer successivement deux fois en moins de 5 s sur le bouton power pour interrompre la vidéo-projection. Envisagez enfin la panne de la lampe…

Commençons par identifier le scénario nominal d'utilisation du vidéoprojecteur : le brancher, puis l'allumer (bouton power), puis connecter l'ordinateur. Ensuite, l'éteindre, puis le débrancher. Si nous ajoutons la possibilité de l'éteindre alors qu'il est allumé ou connecté, puis celle de le débrancher intempestivement, nous arrivons au diagramme de la figure ci-dessous.



Ajoutons le comportement suivant : si l'on éteint le vidéoprojecteur sans le déconnecter de sa source, il repassera directement dans l'état *Connecté* quand on le rallumera. Les deux états *Allumé* et *Connecté* intègrent chacun une activité durable : respectivement la projection d'un écran bleu ou de la source d'entrée. Le diagramme devient alors comme indiqué sur la figure ci-dessous.



Ajoutons maintenant les activités continues de préchauffage et de refroidissement de la lampe. Il est donc nécessaire d'introduire deux états supplémentaires autour de l'état *Éteint*.
Comment sortir de ces états supplémentaires : soit en utilisant un événement de changement (when) testant la température de la lampe, soit plus simplement une transition automatique. Nous utiliserons cette dernière solution, plus simple et n'obligeant pas à spécifier les températures visées.
Vérifions que nous avons pris en compte tous les scénarios :

d'abord le comportement nominal en début de session de formation :

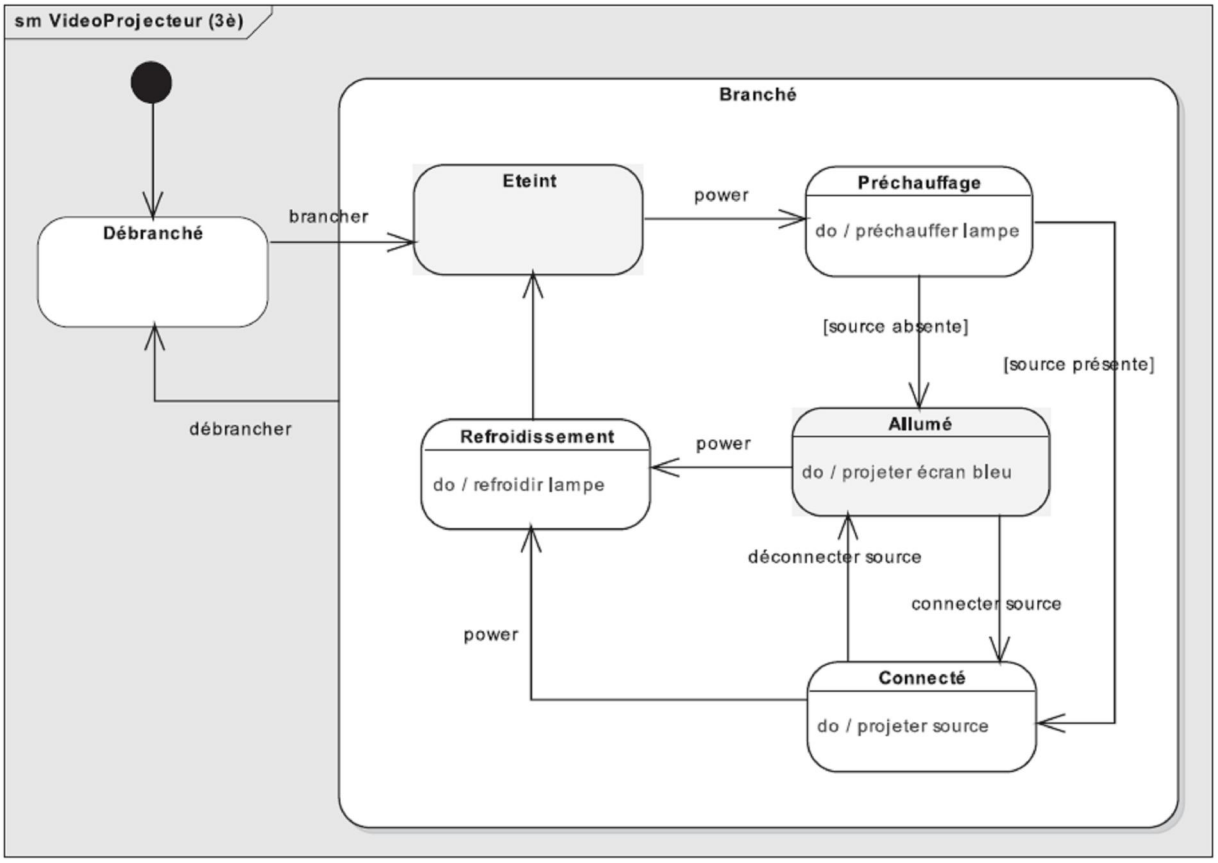
Débranché – brancher – *Éteint* – power – *Préchauffage* – [source absente] – *Allumé* – connecter source – *Connecté*.

Puis, à la pause, le formateur éteint le projecteur :

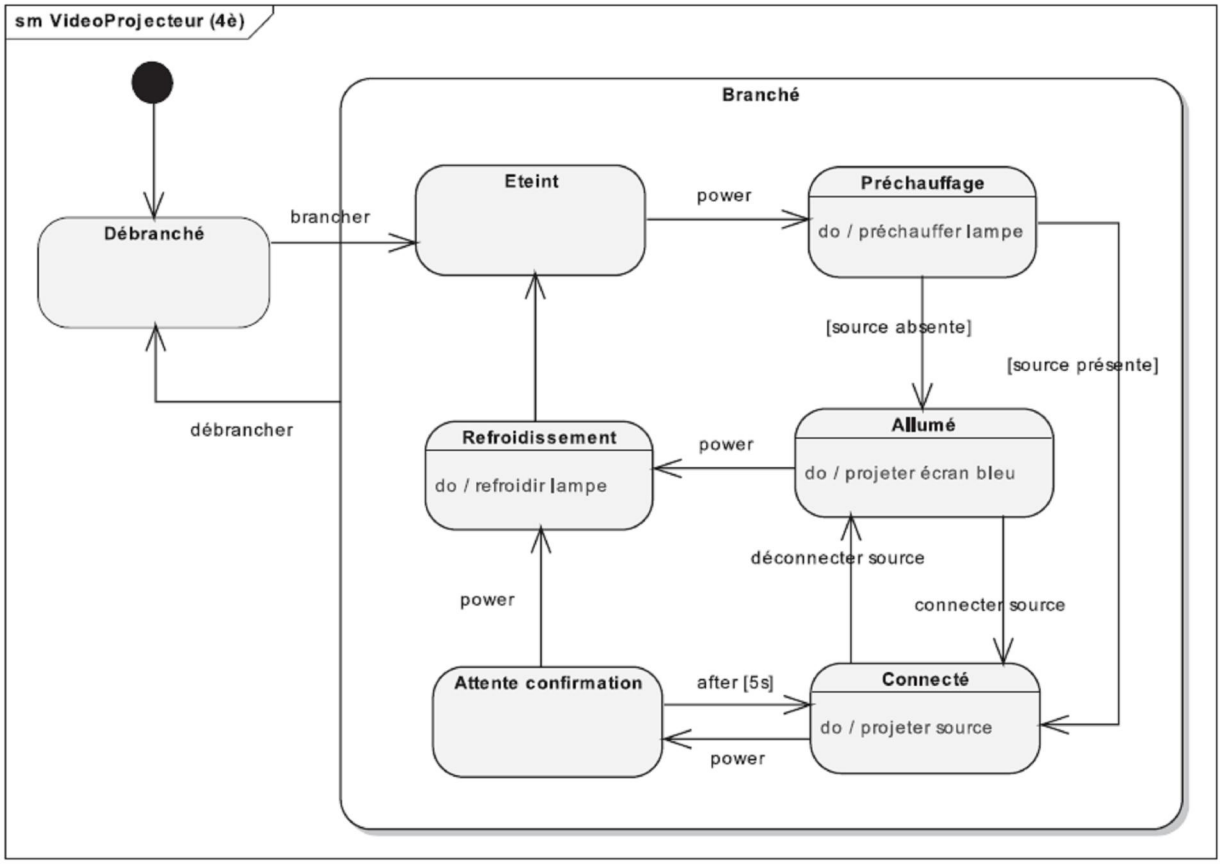
Connecté – power – *Refroidissement* – *Éteint*.

Ensuite, de retour dans la salle, il rallume le projecteur et n'a pas besoin de reconnecter la source :

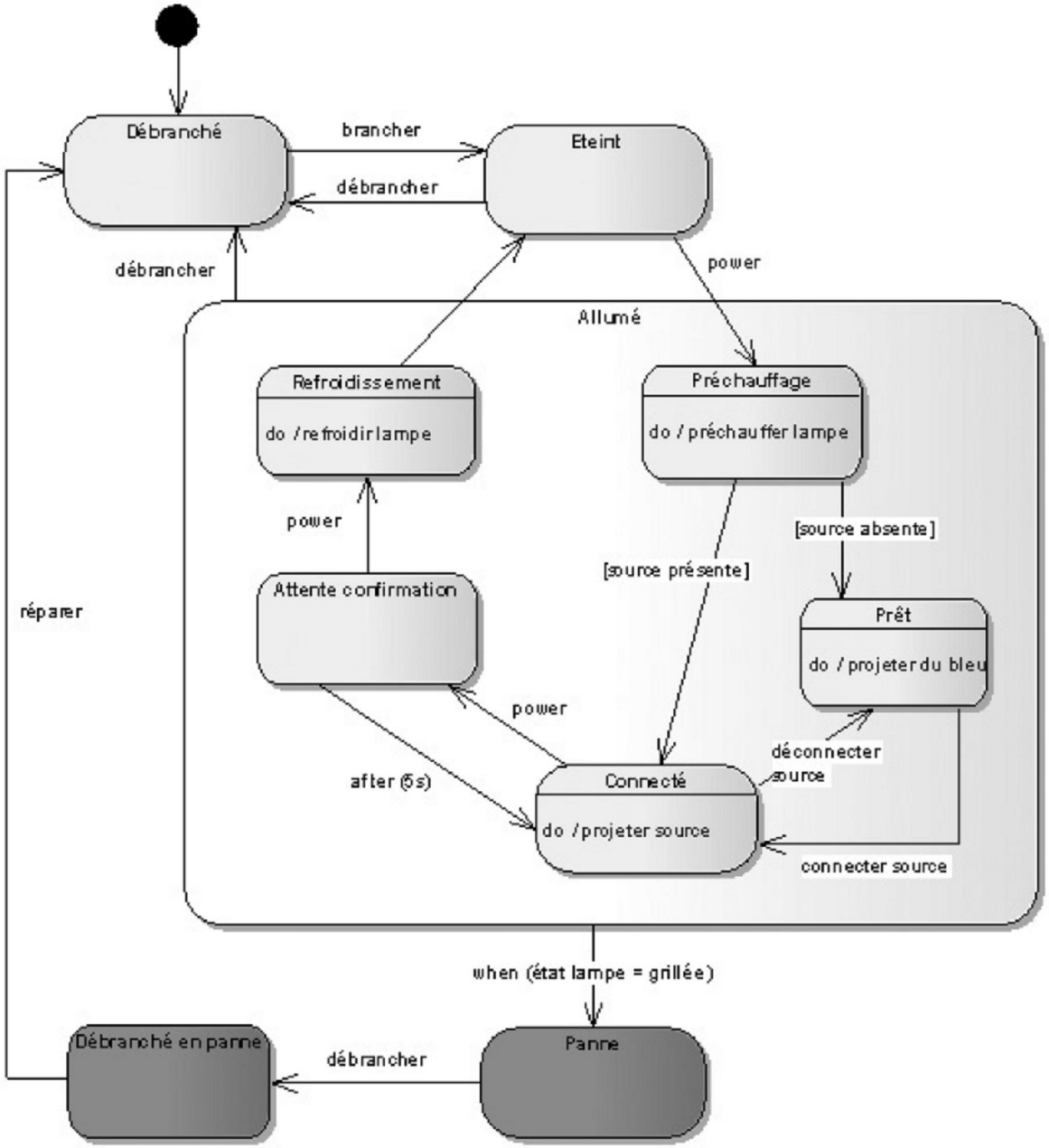
Éteint – power – *Préchauffage* – [source présente] – *Connecté*.



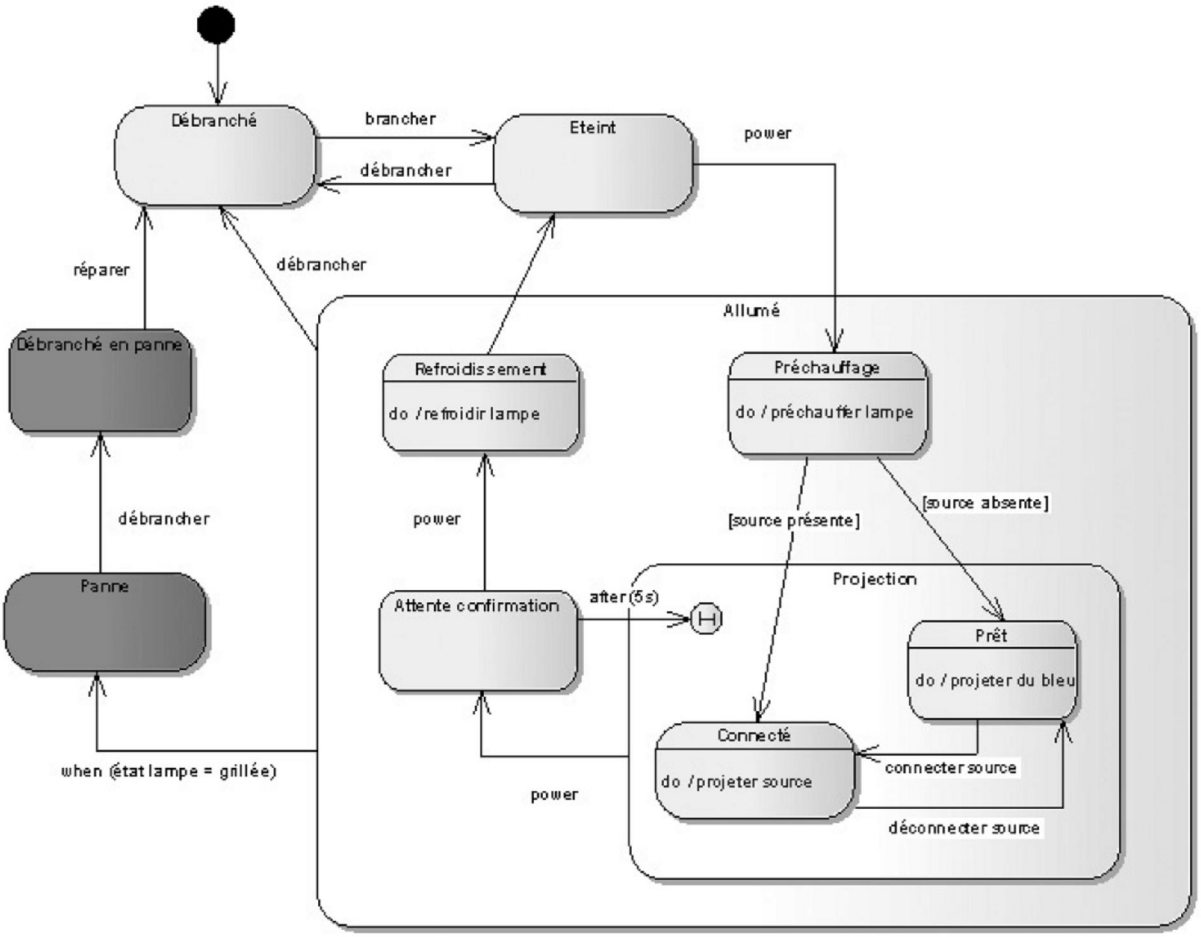
Pour éviter de perdre trop de temps lors d'un appui intempestif sur power dans l'état *Connecté*, les vidéoprojecteurs modernes demandent une confirmation sous la forme d'un deuxième appui sur power en moins de 5 s.
Nous avons vu lors de l'exercice 1 l'événement temporel after (délai)qui va nous servir ici, associé à un nouvel état transitoire d'attente de confirmation.



Ajoutons enfin l'événement redouté : la lampe peut griller dès lors que le projecteur n'est pas éteint. Le plus simple consiste à introduire un nouvel état composite à l'intérieur de *Branché* mais excluant *Éteint*. Il suffit alors d'introduire une transition factorisée déclenchée par l'événement interne de changement when (état lampe = grillée), qui amène vers un état de panne. Le diagramme complété est représenté sur la figure ci-dessous.



En fait, la confirmation pour éteindre le vidéoprojecteur est également obligatoire depuis l'état *Prêt*. On peut donc introduire un super-état englobant *Prêt* et *Connecté*, mais il faut alors utiliser un pseudo-état *History* (H) pour savoir dans quel sous-état revenir quand la temporisation tombe. Le schéma finalisé est montré sur la figure suivante.


Débranché
brancher
débrancher
Eteint
power
réparer
débrancher
Allumé
Débranché en panne
Refroidissement
do / refroidir lampe
Préchauffage
do / préchauffer lampe
débrancher
power
[source présente]
[source absente]
Projection
Panne
Attente confirmation
after (5s)
Prêt
do / projeter du bleu
Connecté
do / projeter source
connecter source
déconnecter source
power
when (état lampe = grillée)