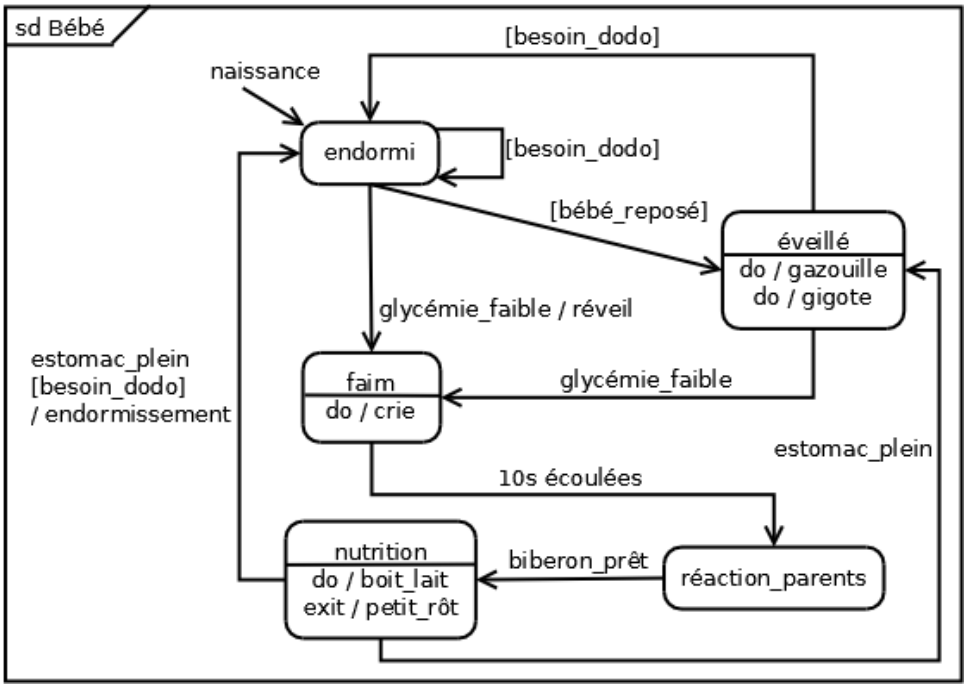
Licence Informatique 3e année

Modélisation Objet

TD3 - Modèle de conception

1- Arreuh

Le diagramme d'états ci-dessous est censé modéliser le fonctionnement d'un bébé. Indiquez les erreurs de modélisation commises (il ne s'agit pas de remettre en question sur le fond les états et transitions mais les erreurs relatives au langage UML).



2- Téléphone

On veut modéliser l'utilisation d'un téléphone par un diagramme d'états (il s'agit des états logiques du téléphone, non des états physiques).

a. Par défaut, le téléphone est raccroché. Si un appel est reçu, il se met à sonner. Si l'appelant raccroche avant que le téléphone soit décroché, le téléphone passe dans l'état raccroché. Si quelqu'un décroche, le téléphone arrête de sonner et une communication a lieu. La communication s'arrête lorsque l'une des deux personnes appelante ou appelée raccroche. À tout moment, la ligne peut être coupée et le téléphone passe alors dans l'état raccroché. Construisez un diagramme d'états modélisant ce fonctionnement.

b. On veut maintenant que le téléphone puisse gérer des appels en attente. Au cours d'une communication, si un nouvel appel survient, il est soit mis en attente si l'utilisateur appuie sur "Attente", soit ignoré si l'utilisateur appuie sur "Ignorer". Si une communication prend fin parce que l'un des interlocuteurs raccroche, et qu'il existe un appel en attente, une nouvelle communication commence. Si une communication prend fin par coupure de la ligne, les appels en attente sont tous annulés.

3- Epidémie

Construire le diagramme d'états correspondant au texte suivant : *Si une personne entre en contact avec un virus, elle a 75% de risques d'être infecté. Si elle est infectée, elle peut devenir malade au terme d'une période d'incubation de 5 jours. Si elle est vaccinée contre ce virus, elle a 90% de chance de ne pas tomber malade, et 10% de risque de développer la maladie. Une fois la personne infectée, elle va devenir contagieuse 3 jours après l'infection. La période de contagiosité dure 4 jours dans le cas où la personne ne développe pas la maladie, et jusqu'à 5 jours après la guérison dans le cas contraire. Une fois malade, l'état de santé de la personne commence à se dégrader chaque jour. Si rien n'est fait, au bout d'un moment, elle meurt. Si la personne reçoit un traitement adéquat, elle entre en rémission, et son état de santé s'améliore graduellement jusqu'à ce qu'elle soit guérie.*

4- Dans quel état j'erre

Construire le diagramme d'activités correspondant au texte suivant extrait d'un mode d'emploi d'un meuble en kit à monter soi-même : *L'assemblage n'est possible que si vous disposez d'un tournevis cruciforme. Commencez par assembler ensemble les pièces A et B. Ajoutez ensuite la pièce C à l'ensemble. Si les repères R1 et R2 ne sont pas bien alignés, désolidarisez C du reste et recommencez l'assemblage de C. Si vous avez le modèle M1, ajoutez ensuite la pièce D1, sinon ajoutez la pièce D2. Installez ensuite le tiroir et les éléments décoratifs dans l'ordre que vous voulez.*

5- MVC

On veut représenter le fonctionnement d'un logiciel construit selon le pattern MVC (Modèle Vue Contrôleur). Ici, la partie Modèle gère uniquement un entier et peut ajouter 1 à l'entier ou retirer 1 à l'entier. Une modification de l'entier entraine une mise à jour de la Vue. La partie Contrôleur comporte deux boutons : un bouton Plus, qui, lorsqu'il est appuyé, déclenche l'ajout de 1 dans le modèle. Un bouton Moins, qui, lorsqu'il est appuyé, déclenche la décrémentation de 1 dans le modèle.

a. Construire le diagramme d'activités correspondant à un fonctionnement séquentiel du programme (les traitements des clics se font les uns à la suite des autres).

b. Construire le diagramme d'activités correspondant à un fonctionnement parallèle du programme.

6- Les deux faces d'une pile

On veut modéliser le comportement d'une pile (au sens informatique) avec une capacité maximum. Une pile offre les méthodes *add(e)* pour ajouter un élément *e* et *remove()* qui renvoie l'élement du dessus de la pile (et le retire de la pile). *add(e)* n'ajoute *e* sur la pile que si la capacité maximum n'est pas atteinte (on suppose qu'il existe une méthode *count()* qui renvoie le nombre d'éléments empilés et un attribut *max* qui représente la capacité de la pile). *remove()* entraine le retrait d'un élément uniquement si la pile n'est pas vide.

a. Construire un diagramme d'états correspondant à ce fonctionnement.

b. Construire un diagramme d'activités correspondant à ce fonctionnement.