ENSIMAG

ACVL

2e année alternance

Éditeur diagrammes états/transitions

Auteurs:
Arthur De Kimpe
Maxime Hagenbourger
Nathanaël Couret
Thomas Bianchini

Enseignant:
Akram Idani

November 12, 2015

Ce document est un rapport rendant compte de notre travail fourni sur le projet ACVL. Il contiendra notemment l'analyse et la conception de notre application. Cette application a pour vocation de créer et d'éditer des diagrammes états transitions en UML.

Contents

1 Analyse **4**
1.1 Cas d'utilisation 4

2 Conception **7**
2.1 Validation d'un diagramme état transition 7

3 Manuel utilisateur **8**

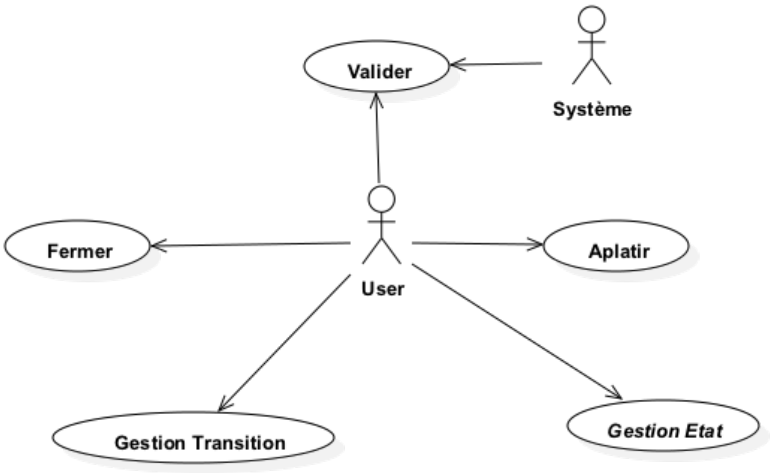
4 Conception **8**

1 Analyse

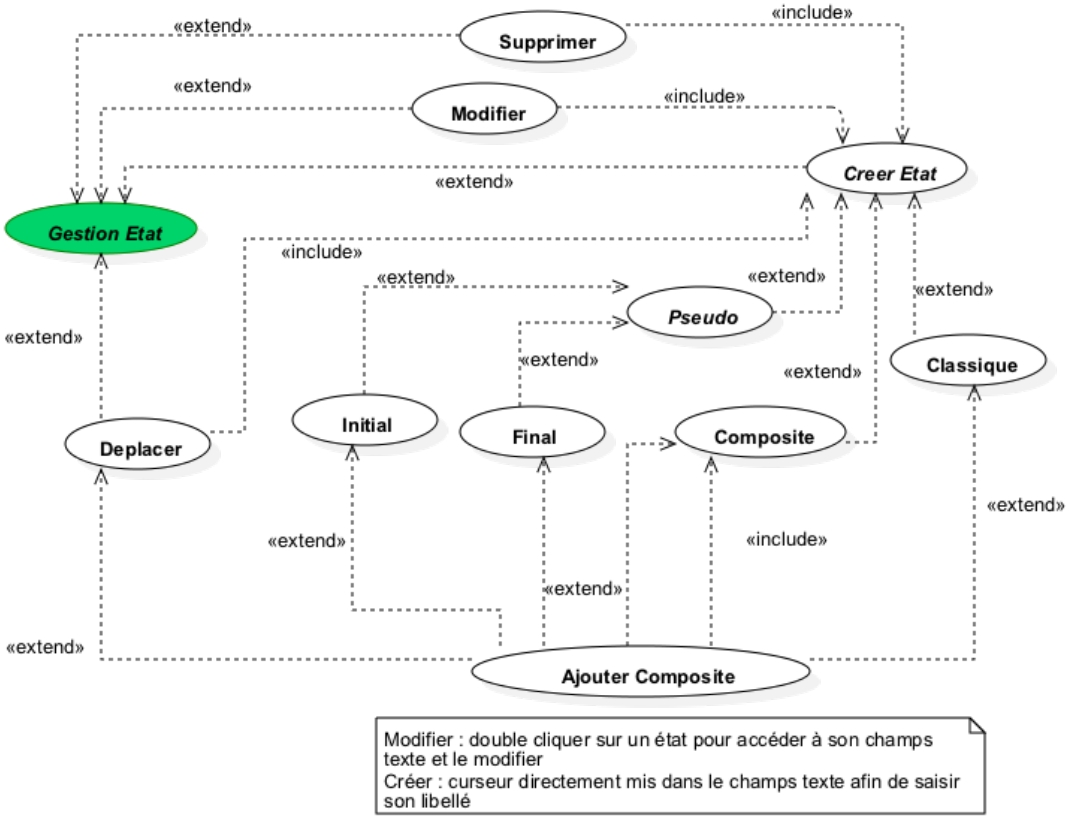
L'analyse de l'application se découpe en deux parties. Tout d'abord l'analyse des cas d'utilisation puis les diagrammes de séquence des cas d'utilisation qui méritent à notre sens un éclaircissment sur l'utilisation de l'application.

1.1 Cas d'utilisation

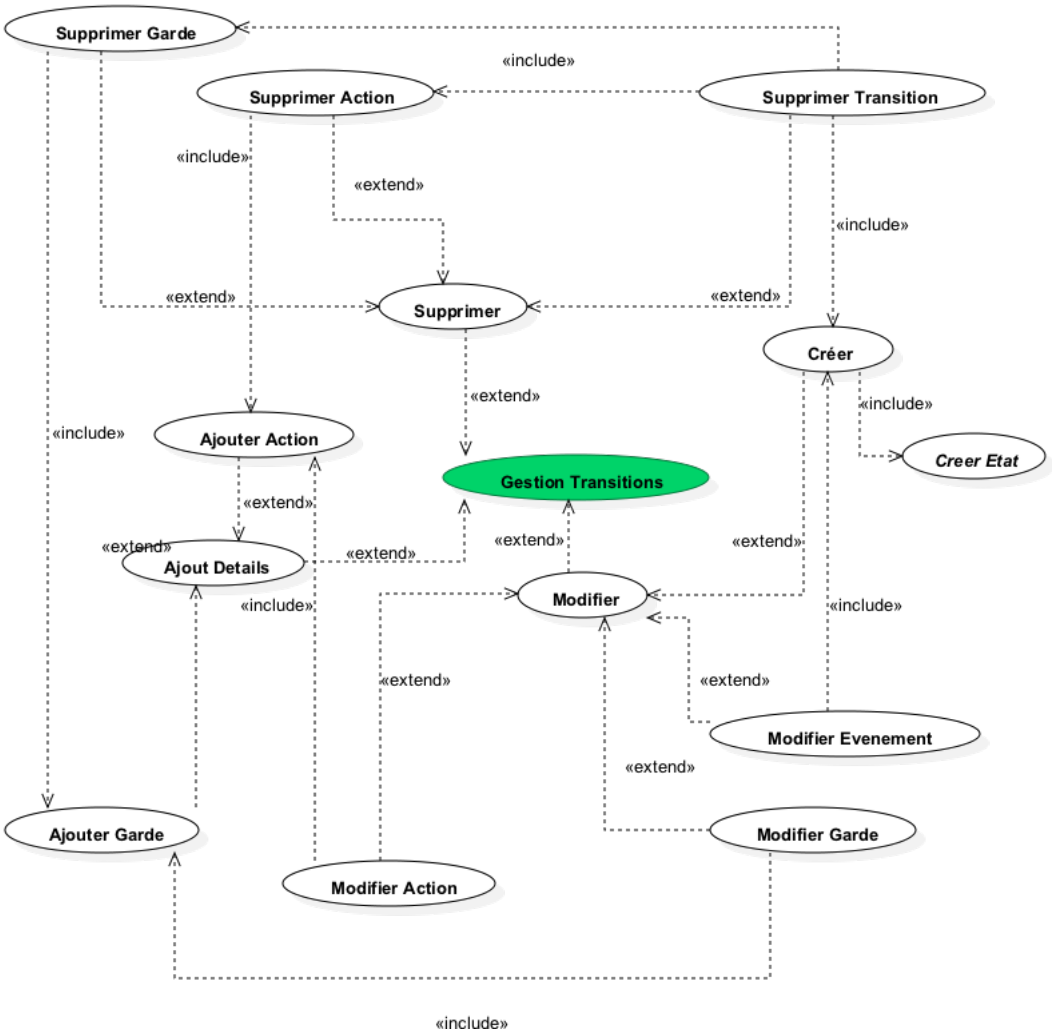
La vision générale des cas d'utilisation est décrite ci-après :



Dans un soucis de clarté nous avons découpé le cas “Gestion état ” ...



... et le cas “Gestion des transtions”



2 Conception

2.1 Validation d'un diagramme état transition

La validation d'un diagramme dépend évidemment du modèle. Ainsi nous considérons que la validation est un controlleur. La classe DiagramValidator contient donc un ensemble de méthode qui vérifient l'ensemble des conditions identifiées par l'équipe. Nous vérifions donc : tous les états sont atteignables, un état n'est pas un puit (on ne peut pas en sortir sauf pour les états finaux), un diagramme contient forcément un état initial et au moins un état final, un diagramme doit être déterministe selon les transitions (2 transitions identiques sortants d'un même état est interdit)...
Nous avons identifié que chaque état a des contraintes de validité qui lui sont propres. Nous avons donc choisi d'utiliser le patron visiteur afin de créer une méthode de validation par type d'état. Ainsi la condition "un état initial doit avoir seulement une transition sortante" n'est vérifiée que dans la méthode du visitor qui prend en paramètre un InitialState par exemple. Le code source du ValidVisitor présente l'ensemble des conditions vérfiiées pour chaque type d'état.
De plus, nous souhaitons que l'utilisateur soit au courant des erreurs présentes dans son diagramme. Pour cela nous avons créer une classe DiagramError qui permet notamment de d'avoir une liste d'erreurs dans l'objet DiagramValidator. À chaque erreur détectée, on ajoute un objet DiagramError dans la liste de l'objet DiagramValidator. Une fois le traitement de toutes les conditions accomplies, la liste des erreurs est récupérée par le controleur liée à la vue. Ce dernier se charge alors de transmettre cette liste à la vue qui regarde simplement si la liste est vide (pas d'erreur) ou pas. Dans ce dernier cas, elle affiche dans une popup l'ensemble des erreurs.

3 Manuel utilisateur

Pour ajouter des états normaux, composites, initiaux ou finaux il suffit de cliquer sur le bouton correspondant dans la barre de menu affichée en haut de l'application.

Les états sont déplaçables en les glissant/déposant quand le curseur de la souris indique que cela est possible.

Pour tracer une transition vers un autre état, il faut placer la souris au centre d'un état, celui-ci passe en surbrillance et à partir de ce moment il est possible de maintenir le clic gauche et tracer une transition vers un autre état.

Pour supprimer état ou transition, il faut sélectionner la cible et presser la touche Suppr. (ou sur Mac : fn + backspace).

Pour savoir si un graphe est valide, il convient de cliquer sur le bouton Validate qui informe l'utilisateur de la validité du graphe.

L'aplatissage d'un graphe se fait en pressant le bouton flatten. Si le graphe est invalide alors l'opération n'est pas effectuée.

Le double clic sur un état normal ou composite permet d'éditer son nom.

Concernant les transitions c'est pas encore FAIT.

4 Conception