Année universitaire 2015-2016

Projet d'Analyse, Conception, Validation
Editeur UML

BATAL Thibaut
BEULE DAUZAT Rémy
RAKOTOARISOA Jérémy
ROUSSEAU Jean-Beanjamin

Sommaire

Terminologie

Conteneur : Graphiquement, zone contenant un état initial et pouvant contenir des états/transitions. Un état composite et la fenêtre d'édition du diagramme sont des conteneurs du point de vue utilisateur.

Conteneur principal : Le conteneur qui n'est le fils d'aucun autre conteneur.

Editeur : L'application développée.

Editeur graphique : Emplacement où sont affichés les éléments du diagramme crée par l'utilisateur.

OS : Le système d'exploitation sur lequel est exécuté l'application.

Transition : Texte précédé par une flèche pour le relier à son état source, et suivi d'une autre flèche pour le relier à son état destination.

Utilisateur : Sous-entendu utilisateur final de l'application.

A) Documents d'analyse

Acteurs

Un seul acteur agit sur l'application : l'utilisateur.

Diagramme de cas d'utilisations

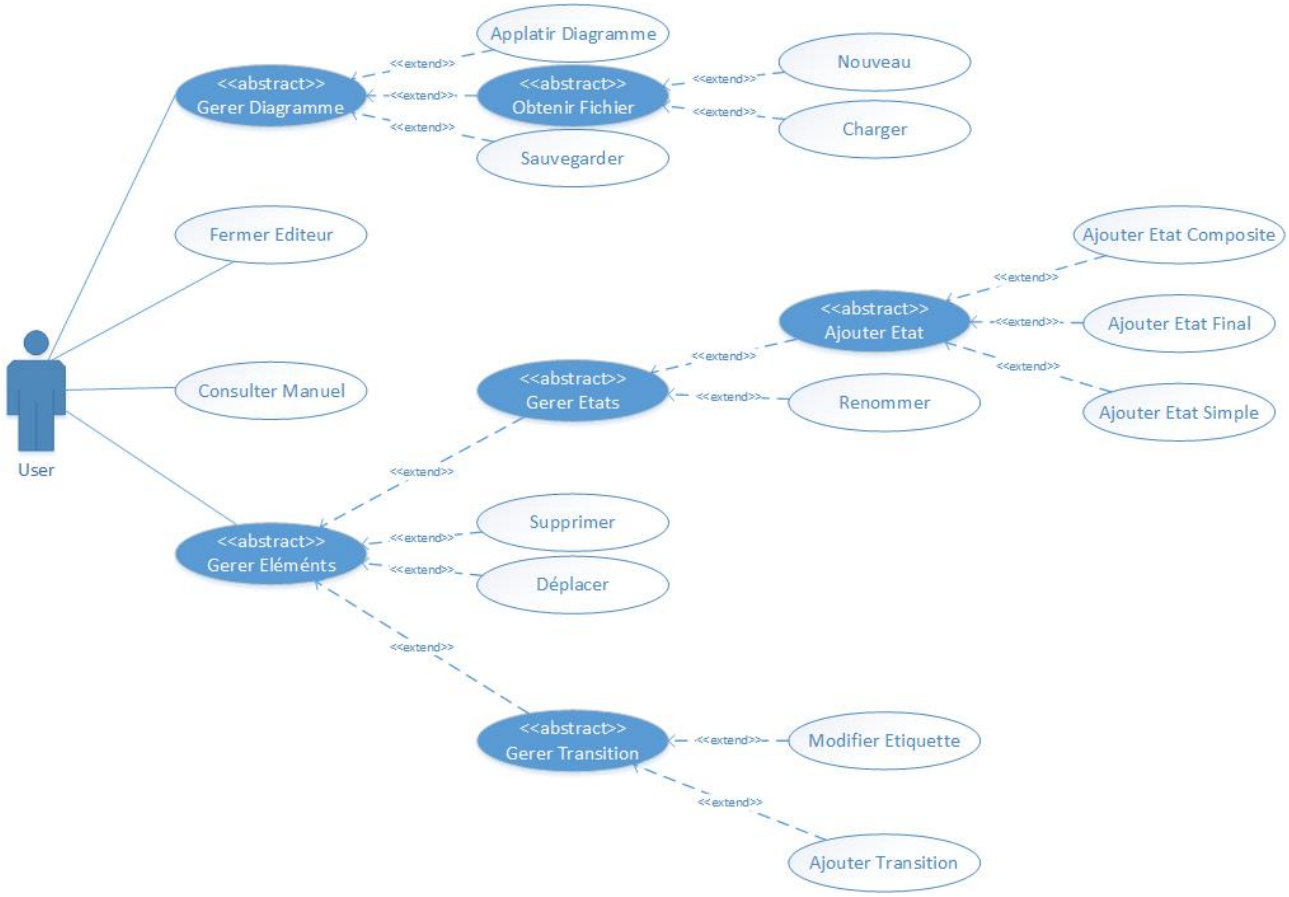


Illustration 1: Diagramme des cas d'utilisation

Voici l'ensemble des actions que l'utilisateur pourra effectuer sur notre application. Nous avons décidé de rester synthétique lors de la création de ce diagramme afin d'offrir une vision claire de nos objectifs.

Description des cas d'utilisation

Après chaque ajout, modification, suppression ou l'ouverture d'un diagramme, l'ensemble des contraintes imposées par le cahier des charges pour un diagramme d'état-transitions est vérifié. En cas d'erreur un message non bloquant s'affiche dans une barre d'erreur au dessus du diagramme. Ces contraintes sont :

- L'unicité des états (2 états ne peuvent pas avoir le même nom dans un même conteneur);
- Toutes les transitions doivent être déterministes (2 transitions d'un même état ne peuvent pas avoir la même garde et le même évenement) ;
- Aucun état ne doit être bloquant (chaque état doit avoir au moins une transition renvoyant vers un autre état sauf si c'est un état final).

Créer diagramme

Appel du cas

A l'ouverture de l'*éditeur* ou lorsque l'*utilisateur* clique sur Fichier->Nouveau.

Actions du cas

Un conteneur est crée, on le nommera dans ce document *conteneur principal*. Cela entraine la création d'un état initial associé, nommé "init_". Un état initial est affiché dans l'*éditeur graphique en haut à gauche*.

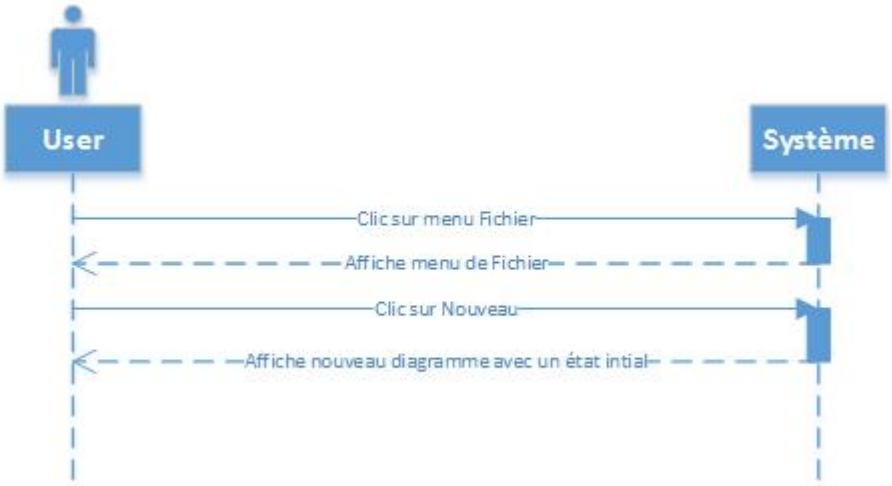


Illustration 2: Diagramme de séquence d'analyse de la création d'un nouveau diagramme

Charger diagramme

Appel du cas

Lorsque l'utilisateur clique sur Fichier->Ouvrir.

Actions du cas

Le gestionnaire de fichiers de l'OS s'ouvre, l'utilisateur y sélectionne son fichier. Les données qui y sont écrites sont alors chargées et affichées dans l'*éditeur graphique*. L'utilisateur peut ensuite modifier et sauvegarder son diagramme.

Sauvegarder diagramme

Appel du cas

Lorsque l'utilisateur clique sur Fichier->Sauvegarder

Actions du cas

Si aucun fichier n'est associé au diagramme (celui-ci a été créé dans l'éditeur par le cas "Créer un diagramme" plutôt que "Charger un diagramme"), le gestionnaire de fichiers de l'OS s'ouvre pour

demander à l'utilisateur de choisir un chemin et nom de fichier. Un fichier est alors crée et associé au diagramme.

Le contenu du diagramme est écrit dans le fichier qui lui est associé. Si un fichier avec ce nom existe déjà, il est écrasé.

Aplatir diagramme

Appel du cas

Lorque l'utilisateur clique dans "Fichier" puis que l'utilisateur clique sur "Aplatir".

Actions du cas

Le diagramme s'aplatit : tous les états composites et leurs états initiaux et finaux sont supprimés, les états intermédiaires remontent jusqu'au niveau le plus élevé. Si l'état composite n'a pas d'états final, ses états intermédiaires ne seront pas reliés aux états en sortie de l'état composite.

L'affichage est actualisé en conséquences.

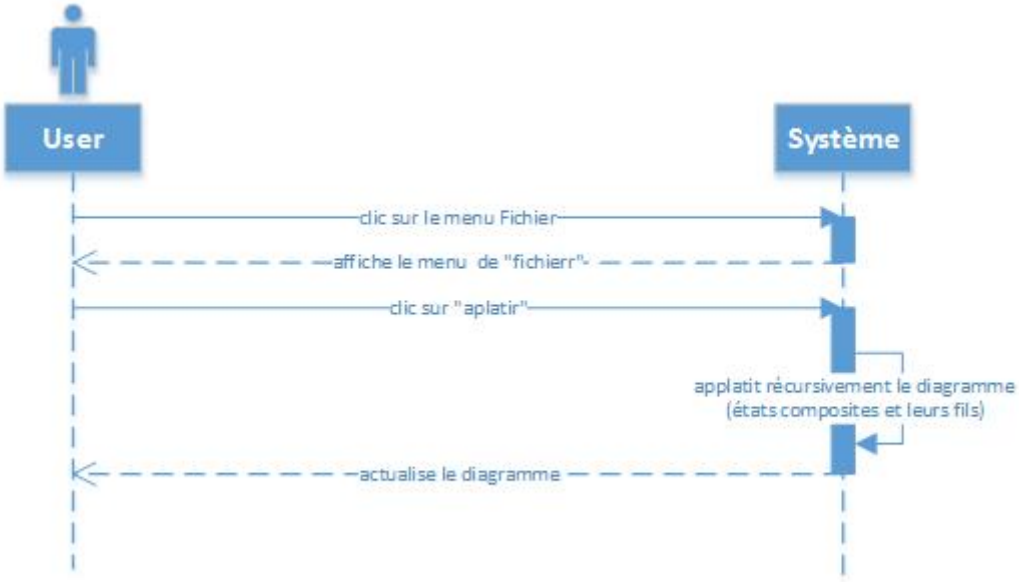


Illustration 3: Diagramme de séquence d'analyse pour aplatir le diagramme

Fermer éditeur

Appel du cas

Lorsque l'utilisateur clique sur l'icône fermer spécifique à l'OS dans l'angle supérieur-droit de l'application.

Actions du cas

L'application est quittée.

Consulter manuel

Appel du cas

Lorsque l'utilisateur clique sur "Aide", "Consulter Manuel".

Actions du cas

Ouverture du fichier Manuel.pdf par l'application par défaut de l'OS.

Ajouter état simple

Appel du cas

Lorsque l'utilisateur clique droit sur un conteneur dans l'*éditeur graphique* puis sur "Ajouter ->Etat->Simple".

Actions du cas

Affichage d'une textbox pour que l'utilisateur entre le nom du nouvel état.L'utilisateur ne peut pas ne rien mettre puis cliquer sur ok. Il doit obligatoirement rentrer un nom, sinon un message d'erreur apparaît.

L'état est alors crée avec ce nom, et comme conteneur parent celui sur lequel le clic droit a été effectué.

Affichage de l'état et de son nom dans l'*éditeur graphique* dans le conteneur parent.

Ajouter état Final

Appel du cas

Lorsque l'utilisateur clique droit sur un conteneur dans l'*éditeur graphique* puis sur "Ajouter ->Etat->Final".

Actions du cas

Affichage d'une textbox pour que l'utilisateur entre le nom du nouvel état. L'utilisateur ne peut pas ne rien mettre puis cliquer sur ok. Il doit obligatoirement rentrer un nom, sinon un message d'erreur apparaît.

L'état est alors crée avec ce nom, et comme conteneur parent celui sur lequel le clic droit a été effectué.

Affichage de l'état et de son nom dans l'*éditeur graphique* dans le conteneur parent.

Ajouter état Composite

Appel du cas

Lorsque l'utilisateur clique droit sur un conteneur dans l'*éditeur graphique* puis sur "Ajouter ->Etat->Composite".

Actions du cas

Affichage d'une textbox pour que l'utilisateur entre le nom du nouvel état.L'utilisateur ne peut pas ne rien mettre puis cliquer sur ok. Il doit obligatoirement rentrer un nom, sinon un message d'erreur apparaît.

L'état est alors crée avec ce nom, et comme conteneur parent celui sur lequel le clic droit a été effectué. Dans la zone de l'état composite est ajouté un état initial avec comme nom "init_nomDuComposite".

Affichage de l'état, de son état initial et de son nom dans l'*éditeur graphique* dans le conteneur parent.

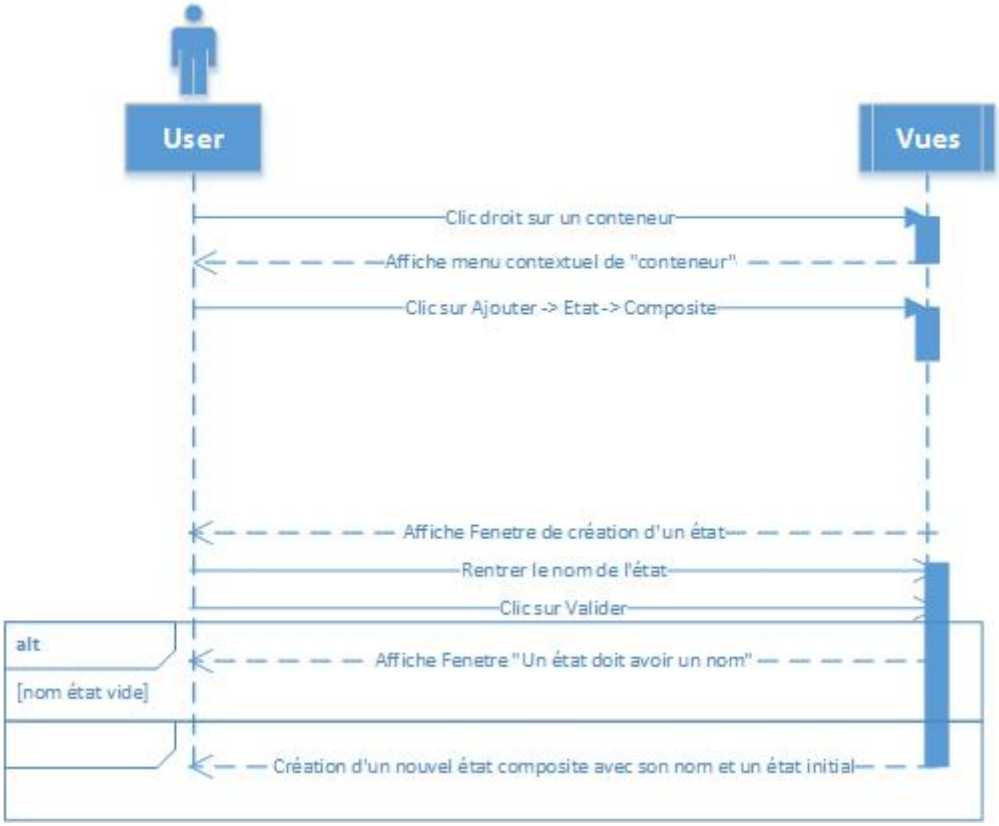


Illustration 4: Diagramme de séquence d'analyse de la création d'un état composite

Renommer état

Appel du cas

Lorsque l'utilisateur fait un clic droit dans l'*éditeur graphique* sur l'état qu'il souhaite renommer , un menu contextuel s'affiche, et qu'il clique alors sur "Renommer".

Actions du cas

Affichage d'une textbox pour modifier le nom. L'utilisateur ne peut pas ne rien mettre puis cliquer sur ok. Il doit obligatoirement rentrer un nom, sinon un message d'erreur apparaît.

Supprimer élément

Appel du cas

Lorsque l'utilisateur fait un clic droit dans l'*éditeur graphique* sur l'élément qu'il souhaite supprimer, un menu contextuel s'affiche, et qu'il clique alors sur "Supprimer".

Après sélection de l'élément, en cliquant sur "Supprimer" sur le clavier.

Actions du cas

Si c'est un état initial, affichage d'un message d'erreur. L'utilisateur doit cliquer sur ok pour revenir sur l'édition.

Si c'est un autre état, on supprime toutes les transitions associées, l'état lui-même, et tout son contenu si c'est un état composite.

Si c'est une transition, on la supprime.

Déplacer élément

Appel du cas

Lorsque l'utilisateur clique sur un élément. Le cas d'utilisation s'arrête lorsque l'utilisateur relâche le bouton de la souris (effet cliquer-déplacer).

Actions du cas

L'élément suit le curseur de la souris. Il ne peut pas changer l'élément de conteneur dans lequel il se trouve. Les conteneurs parents peuvent passer au dessus de conteneur fils mais ils ne sont pas inclus dedans.

L'affichage est actualisé de manière à ce que les liens entre états et transitions associés à l'élément déplacé soient redessinés.

Ajouter transition

Appel du cas

Lorsque l'utilisateur clique droit dans l'*éditeur graphique* sur un état puis sur "Ajouter Transition". Un état final ne peut pas avoir de transition qui partent de lui.

Actions du cas

Une liste déroulante s'affiche, contenant tous les états existants autorisés dans le même conteneur que l'état sur lequel l'utilisateur a fait un clic droit. L'utilisateur y sélectionne l'état de destination de la transition créée. Un état initial ne peut être relié directement à un état final et aucune liaison n'arrivera sur lui.

Si ce n'est pas une transition partant d'un état initial, 3 textbox sont également présentes pour que l'utilisateur entre la garde, l'événement et l'action à associer à la transition créée.

La transition est associée au conteneur de l'état sur lequel l'utilisateur a fait un clic droit. Ce dernier devient son état source.

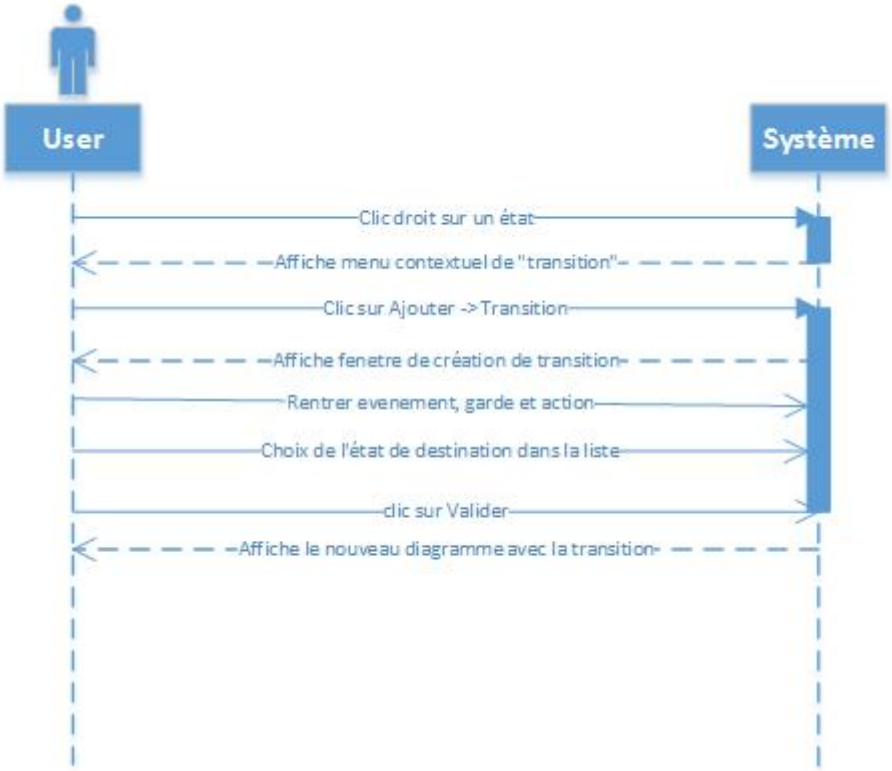


Illustration 5: Diagramme de séquence d'analyse pour l'ajout d'une transition

Modifier étiquette

Appel du cas

Lorsque l'utilisateur clique droit dans l'*éditeur graphique* sur une transition puis sur "Modifier Transition". Une transition partant d'un état initial ne peut être modifiée

Actions du cas

Une nouvelle fenetre s'affiche avec 3 textBox pour modifier la garde, l'évenement et l'action.

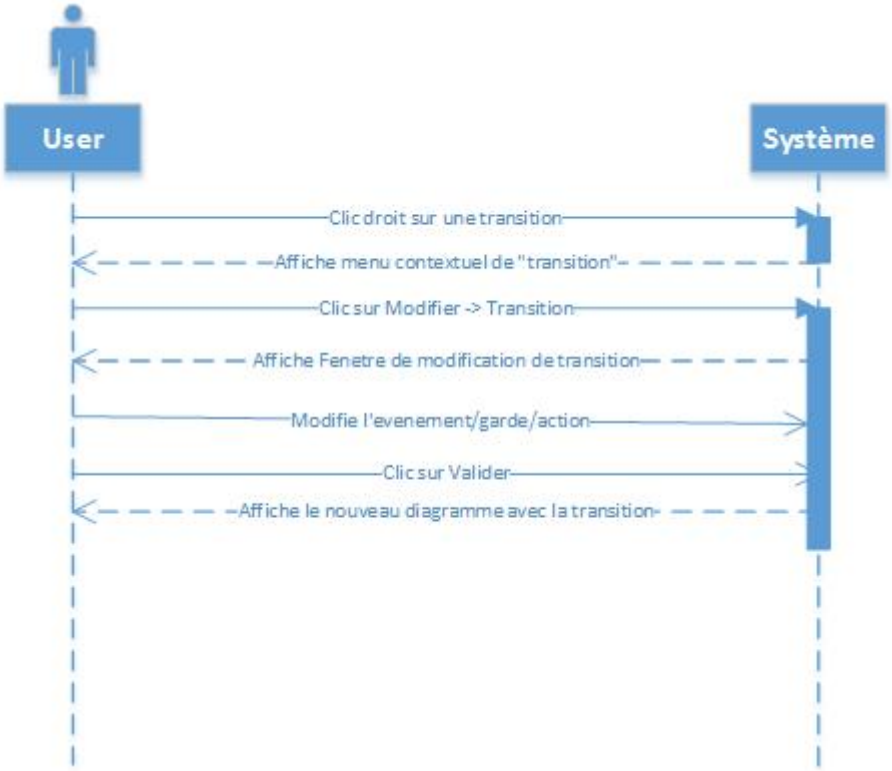


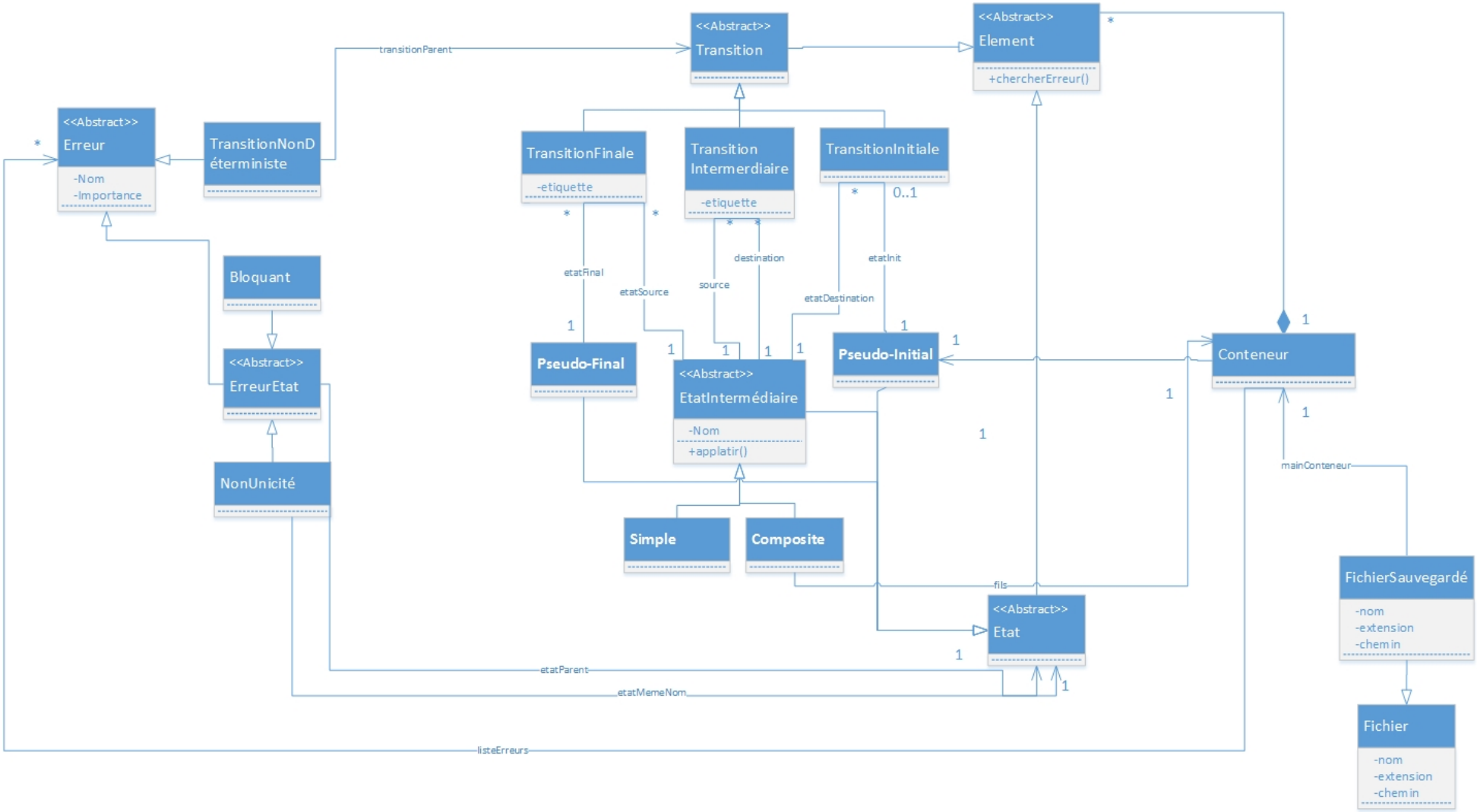
Illustration 6: Diagramme de séquence d'analyse pour la modification d'une transition

Diagrammes de classe d'analyse

Ce diagramme de classes d'analyse correspond à un premier plan de conception prenant en compte les besoins exprimés par le client. Nous avons ainsi pu détailler les principaux objets UML du modèle de la future application.

Le point le plus important de l'application, c'est l'interconnexion entre les états et les transitions. Nous avons décidé de créer trois types de transitions pour correspondre aux quatre types d'états demandés.

La notion de conteneur nous a paru pertinente afin de relier le cas du diagramme initial et celui des états composites. Nous avons ainsi pu effectuer des traitements génériques sur l'ensemble du diagramme.


<<Abstract>>
Transition
<<Abstract>>
Element
+chercherErreur()
transitionParent
<<Abstract>>
Erreur
-Nom
-Importance
TransitionNonDéterministe
TransitionFinale
-etiquette
Transition Intermerdiaire
-etiquette
TransitionInitiale
0..1
Bloquant
etatFinal
etatSource
source
destination
etatInit
etatDestination
Pseudo-Final
Pseudo-Initial
<<Abstract>>
EtatIntermédiaire
-Nom
+applatir()
Conteneur
<<Abstract>>
ErreurEtat
NonUnicité
Simple
Composite
fils
mainConteneur
<<Abstract>>
Etat
FichierSauvegardé
-nom
-extension
-chemin
etatParent
etatMemeNom
Fichier
-nom
-extension
-chemin
listeErreurs

B) Documents de conception

Voici l'architecture MVC que nous avons décidée de mettre en place à partir du diagramme de classes précédent

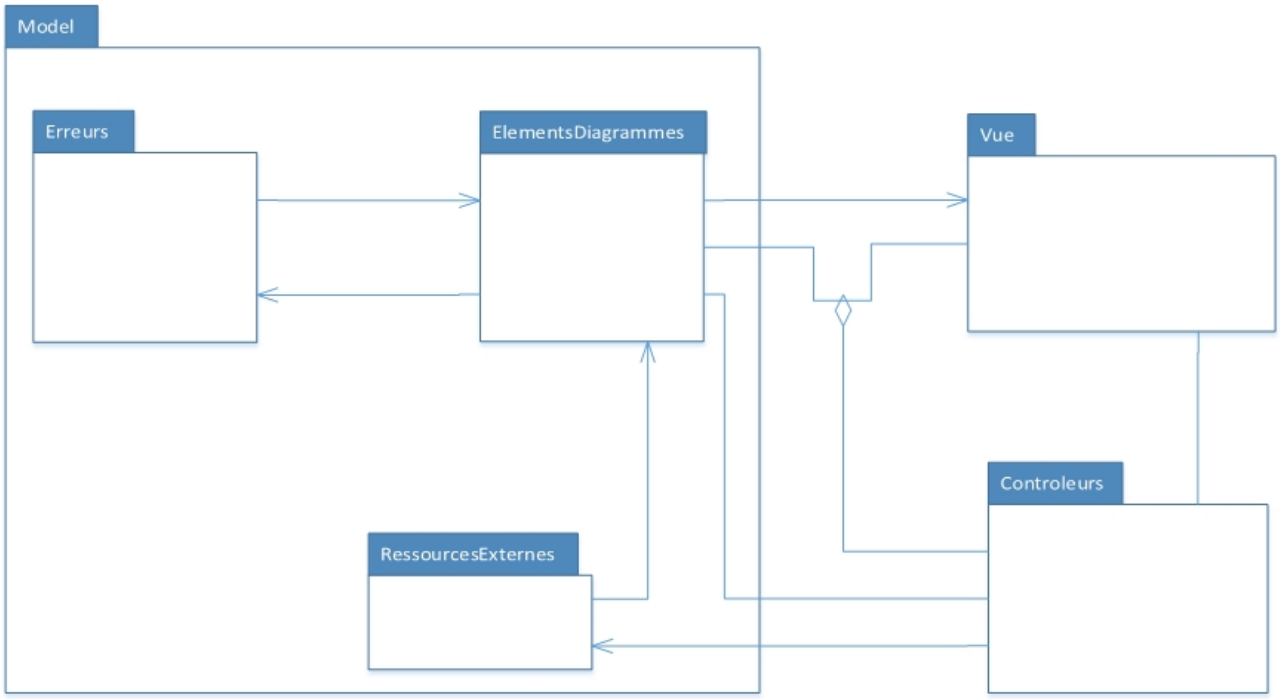


Illustration 8: Modèle MVC

Voici, ci-dessous, le premier diagramme de classes de conception que nous avons réalisé. Il met en avant nos choix architecturaux et nos choix de patrons de conception.

On peut remarquer l'utilisation de trois patrons :

- Le patron singleton autour des classes Ihm, ControleurDiagramme, ControleurFichier et EditeurGraphique afin d'y accéder facilement de n'importe où tout en garantissant leur unicité.
- Le patron Observateur sur les éléments du modèle afin de mettre à jour facilement l'IHM en fonction des modifications apportées par l'utilisateur.
- Le patron Composite afin d'appliquer des méthodes uniformes sur les éléments du diagramme et les éléments des états composites. Ce patron n'a pas été respecté à la lettre. Nous avons décidé de passer par un conteneur (qui contient des éléments, pouvant être eux-mêmes des états composites) pour utiliser ce patron dans notre architecture.

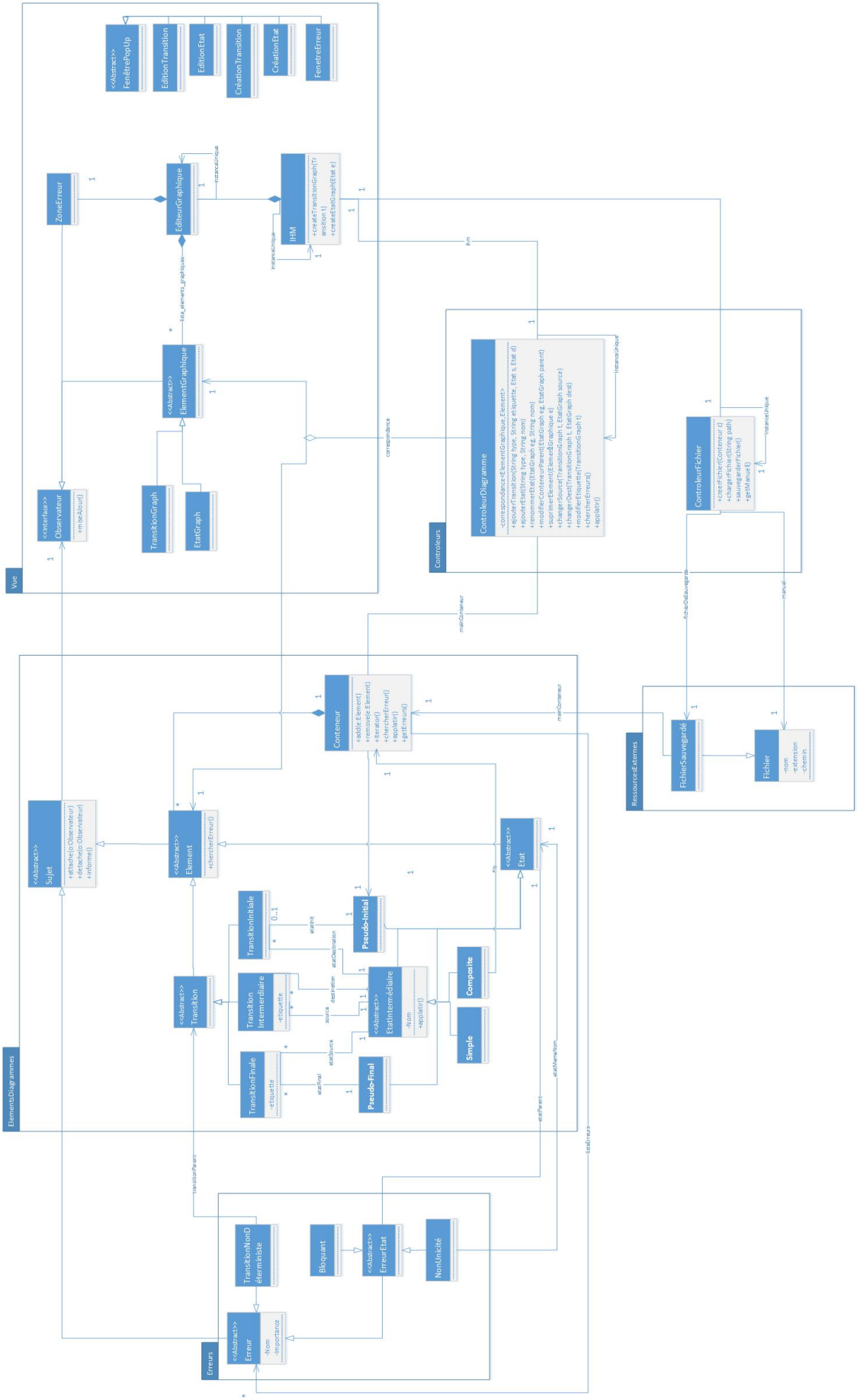


Illustration 9: Premier Diagramme de classe de conception

Afin de comparer, voici le package RessourcesExternes et Controleurs en fin de développement de l'application. Vous pouvez également trouver dans des fichiers à part le diagramme de classes de conception complet et les principaux diagrammes. Ces diagrammes ne peuvent pas être exposés de façon lisible ici.

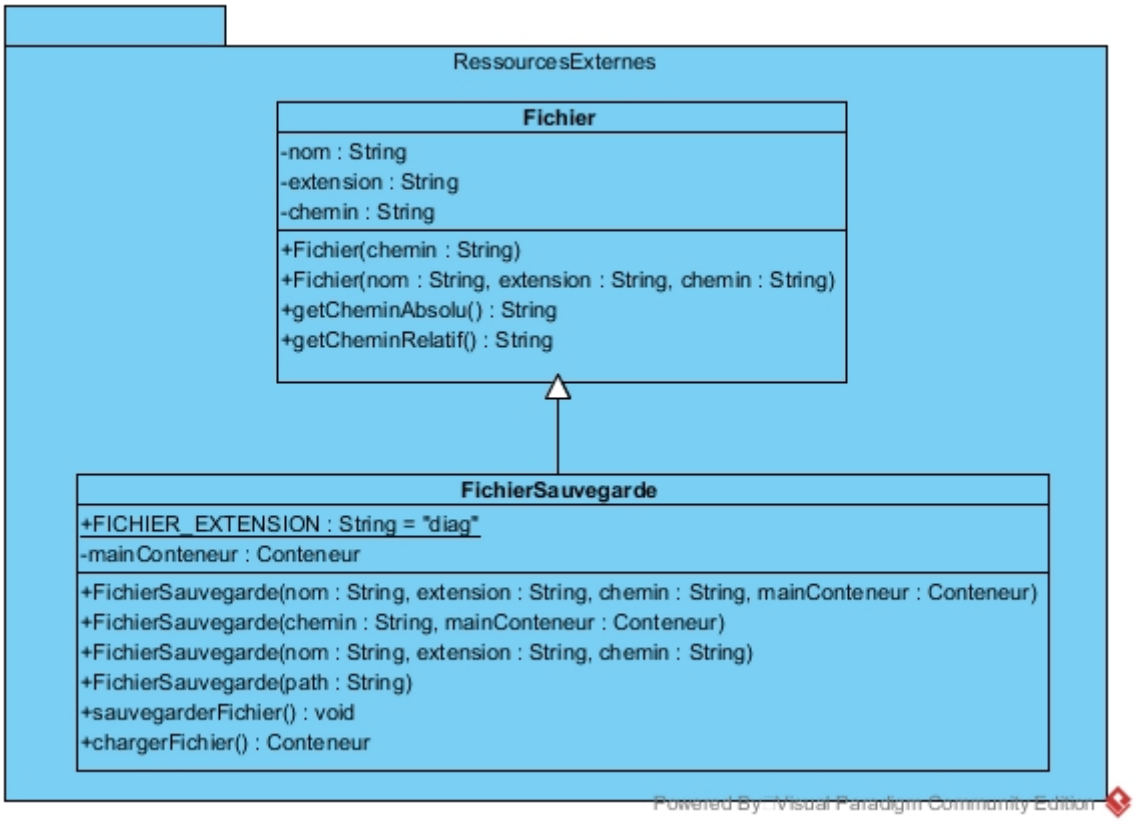


Illustration 10: Diagramme de conception final du package RessourcesExternes

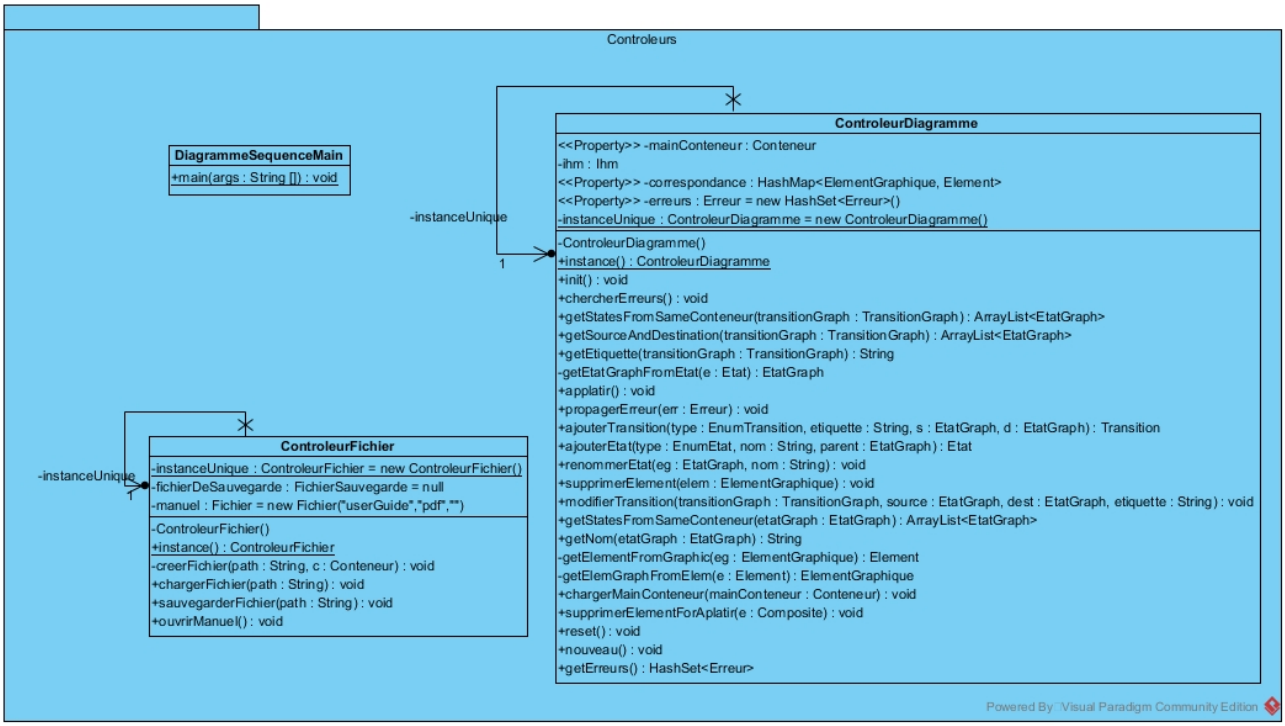


Illustration 11: Diagramme de conception final du package Controleur

Enfin, vous pouvez retrouver dans le dossier diagrammeSéquenceConception quelques diagrammes de séquences. Nous avons choisi les diagrammes de séquences d'Aplatir, des recherches d'erreurs, de l'ajout d'un état composite et d'une transition et de la supression car nous considérons que ce sont les plus intéressants.

C) Manuel utilisateur

Le manuel utilisateur est dans un fichier UserGuide.pdf que vous pouvez retrouver à coté de ce fichier.

D) Bilan sur les outils de modélisation utilisés

L'une de nos plus grandes difficultés fut la mise en commun des différentes idées au niveau de la phase d'analyse du projet. En effet, souvent nous n'avions pas les même compréhensions des différents concepts du diagramme d'états-transitions UML, ainsi qu'une représentation différente des problèmes et des méthodes de réalisation d'une solution. On a résolu ces problèmes en poussant parfois la formalisation des détails, comme la mise en place de définitions et des descriptions textuelles poussées.

Application développée

Faisons un tour des contraintes explicites du cahier des charges.

- L'utilisateur peut effectivement choisir le type de l'état qu'il désire créer lorsqu'il en ajoute un via le menu contextuel, puis le nommer.
- Il peut également ajouter des transitions, en leur donnant une étiquette qui est composée d'une chaîne de caractères avec un événement, une garde, ainsi qu'une suite d'actions à exécuter. L'utilisateur est libre d'écrire ce qu'il souhaite dans chacun de ces champs, tant qu'il n'y a ni '[', ni ']', ni '/', qui permettent de délimiter les trois champs dans le modèle.
- L'unicité des états est vérifiée à chaque modification du diagramme, via la méthode ControleurDiagramme.chercherPluriciteEtats() qui retourne des erreurs relatives à des états qui portent le même nom.
- Le fait qu'il existe toujours un état initial est géré par le fait qu'à la création d'un nouveau diagramme ou d'un état composite, un état initial associé est automatiquement généré.
- Détecter les transitions non-déterministes comme les états bloquants est effectué à chaque modification du diagramme, via les méthodes respectives ControleurDiagramme.chercherTransitionsNonDeterm() et ControleurDiagramme.chercherEtatsBloquants() qui retournent des erreurs relatives aux transitions non déterministes et états bloquants. Celles-ci sont affichées dans la zone d'erreur, permettant de ne pas bloquer l'utilisateur, qui est donc libre de les ignorer.
- Aplatir un automate est permis en cliquant sur Fichier->Aplatir.

L'ensemble des contraintes explicitement énoncées dans le cahier des charges est donc respecté.

Outils utilisés

Concernant les outils de modélisations que nous avons utilisés, nous nous sommes d'abord tournés vers Microsoft Visio. Ce logiciel a l'avantage de ne pas être contraignant pour l'utilisateur. Cela permet de pouvoir aisément personnaliser un diagramme, mais en apportant un inconvénient: les dépendances étant faibles entre les éléments, dès que nous souhaitons modifier une partie de ce diagramme, les modifications sont trop peu répercutées autour. Par conséquent, il faut réajuster chaque élément environnant pour s'accorder avec la modification faite. Cela est complexe en phase d'étude et de réflexion, où le diagramme est constamment modelé.

Nous avons alors utilisé Visual Paradigm, dont la prise en main peut rebuter, notamment pour l'ajout d'éléments particuliers (tels que des cadres "Alternatif" ou "Option" dans les diagrammes de séquence par exemple). Mais imposant un minimum de formalisation, cela permet de garder une cohérence entre les différents diagrammes. Par conséquent, ces restrictions uniquement au niveau UML apportent plus de simplicité, renforçant les liens entre les différents éléments du diagramme (comparé à Visio). Il est également possible de lier nos sources Java au logiciel, dans l'objectif d'assurer une cohérence entre nos diagrammes et celles-ci.