Editeur d'automates: "AutomatesLab"

Dossier de spécification

Référence	:	SPC-8-0C
Fournisseur		
Date	:	15/12/2022
Version/Édition	:	0C
État	:	Préliminaire
Type de diffusion	:	Diffusion propriétaire
Autre référence	:	

Historique des révisions

Date	Description et justification de la modification	Auteur	Pages / Chapitre	Edition / Révision
15/12/2022	Création		Toutes	0A
06/01/2023	Ajout des règles métiers et IHMs, autres corrections		Toutes	0B
12/01/2023	Remaniement PRJ-001 et correction mineures		Toutes	0C

Table des matières

1. Terminologie

1.1. Abréviations

SGF	**S**ystème de **G**estion de **F**ichiers
UML	**U**nified **M**odeling **L**anguage
MVC	**M**odel **V**iew **C**ontroller
IDE	**I**ntegrated **D**evelopment **E**nvironment
JVM	**J**ava **V**irtual **M**achine
JDK	**J**ava **D**evelopment **K**it
JRE	**J**ava **R**untime **E**nvironment

1.2. Définitions des termes employés

Use case	Cas d'utilisation du système, par extension il représente également une technique de modélisation mise en oeuvre dans UML
Classe	Association de données et de traitements modélisant un élément du système
XML	e**X**tended **M**arkup **L**anguage: langage de balisage pouvant aisément se créer dans une application et s'enregistrer sur disque.
JVM / JRE	Logiciel exécutant du code java compilé, il fait l'interface entre le logiciel et le matériel

Alphabet

Ensemble (fini) de caractères, aussi appelés lettres ou symboles. Exemple: V = {a, b, c} est un alphabet de 3 caractères.

Mot

Aussi appelé chaîne de caractère, ou chaîne. C'est une suite de caractères. Par exemple, "abba" est un mot sur V.

Automate fini

Un automate (fini) est un modèle mathématique utilisé dans la conception de programmes de compilation et d'interprétation. Il s'agit d'une collection d'états (cercles) et de transitions (flèches), pouvant être représenté par un graphe (exemple en Fig.1). L'automate a un état initial (symbolisé par une flèche entrante en Fig. 1), et un ou plusieurs états finaux (cercles doublés).

Un automate peut analyser un mot (par exemple "bab"). Il commence dans son état initial (état 0 sur la Fig.1), analyse la première lettre "b": il va suivre une des transitions existantes possibles. Pour l'automate de l'exemple, il peut soit aller boucler sur l'état 0, soit aller dans l'état 1. Supposons qu'il passe dans l'état 1. Il analyse la seconde lettre "a", et passe dans l'état 2. Il passe à la lettre suivante, "b", et passe dans l'état 3. Il n'y a plus de lettres à analyser, et l'automate est dans un état final : le mot est donc reconnu ("return True"). Sinon, il n'est pas reconnu.

Un automate possédant des "choix" (comme à l'analyse de la première lettre) est dit non déterministe, et inversement. Lorsqu'un automate est non déterministe, on considère qu'il reconnaît un mot s' il existe au moins une succession d'état menant à reconnaître ce mot.

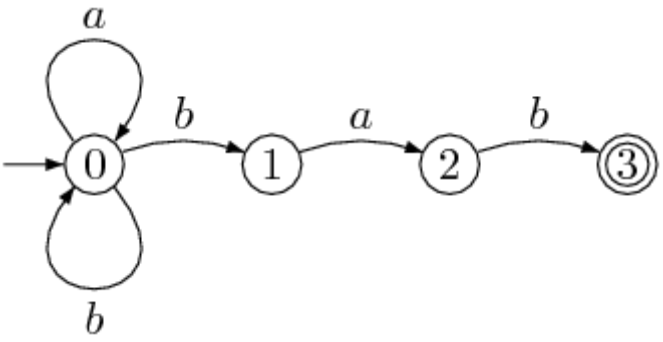

Fig. 1: Représentation d'un automate fini sous forme de graphe

Langage rationnel

Ensemble des mots reconnus par un automate fini. Peut être défini de manière équivalente par une expression régulière (Regex).

2. Introduction

2.1. Objet du document

Les spécifications du logiciel permettent de préciser :

- le besoin exprimé par le Client,
- le détail des fonctions satisfaisant ce besoin, leurs liens et leur logique d'enchaînement,
- les interactions du système avec les utilisateurs,
- la prise en compte des interfaces externes du logiciel,
- les contraintes de réalisation,
- la prise en compte des exigences de qualité.

Les spécifications permettent de déterminer ce que le Client attend du système, c'est à dire le *quoi*. Dès lors que la description fait appel au *comment*, il ne s'agit plus de spécification mais de conception.

Il faut présenter succinctement la structure du document :

- présentation du système global et de ses fonctions,
- description de la position dans le système du logiciel décrit dans cette spécification,
- présentation des fonctionnalités concernées par ce document de spécification,
- description détaillée de chaque fonctionnalité sous la forme d'un ou plusieurs cas d'utilisation, chaque cas d'utilisation comprenant une description du cas d'utilisation et des différents scénarios

2.2. Description du projet

Le projet Editeur d'Automate a pour objectif de créer une application (appelée AutomatesLab) permettant de définir des automates finis et de les sauvegarder dans un format XML adapté. Un automate peut être chargé dans l'éditeur depuis son fichier XML.

Le projet doit respecter les règles métiers suivantes:

- Un fichier XML représentant un automate est valide si et seulement si il suit les règles de représentation définies dans l'annexe 1: "Représentation XML.pdf".
- Un fichier est considéré comme corrompu s'il est illisible par le système d'exploitation hôte (donc cela ne concerne pas le cas où le fichier est ouvert par un autre processus).
- Enregistrer un automate sur disque nécessite que le XML en mémoire soit valide, c'est-à-dire qu'il suive les règles de représentation définies dans le document correspondant cité ci-dessus.
- Un état porte comme nom un entier naturel unique (comme sur l'exemple en Fig. 1).
- Lors de l'attribution d'un numéro à un état, on choisit le plus petit entier naturel disponible.
- Une transition est obligatoirement associée à deux états: l'état d'origine et l'état d'arrivée. L'état d'origine peut être aussi l'état d'arrivée, mais les états ne doivent pas être nuls.
- Un état peut être initial ou final. Il peut y avoir autant d'états finaux que l'on veut. Il y a exactement zéro ou un état initial dans un automate. Si l'utilisateur définit un état comme étant initial alors qu'un précédent état était déjà initial, le précédent perd cette propriété.

2.3. Responsabilités

La rédaction des spécifications est de la responsabilité du Chef de projet.

Il juge de son état complet et décide de sa présentation en revue de spécifications.

Les spécifications sont toujours livrables, elles permettent de décrire de façon complète le travail à réaliser. C'est pourquoi l'approbation du client est indispensable.

2.4. Options non incluses dans le projet

Le projet n'inclut pas les fonctionnalités optionnelles suivantes : la génération d'un automate à partir d'une expression régulière, retrouver l'expression régulière décrite par l'automate, déterminiser les automates, ainsi que sauvegarder, charger et éditer les automates sous forme matricielle.

2.5. Outils utilisés

Les documents de conception sont rédigés avec Google Doc.

Les schémas UML sont créés avec le logiciel StarUML. Les schémas présentés respectent les conventions UML.

L'application sera développée en Java en utilisant l'IDE IntelliJ Idea, ainsi que Maven pour la compilation. Les interfaces graphiques seront implémentées à l'aide de JavaFX.

GitHub sera utilisé pour le versioning du projet.

3. Exigences

3.1. Présentation de la mission du produit logiciel

3.1.1. Fonctions générales du logiciel

AutomatesLab est un logiciel d'édition d'automate. L'utilisateur doit pouvoir concevoir graphiquement un automate grâce à une boîte à outils. Les automates peuvent aussi être conçus en XML. Les automates peuvent être sauvegardés et chargés sous format XML. Il est nécessaire d'avoir des connaissances concernant les automates afin d'utiliser l'application.

3.1.2. Position du logiciel dans le système

AutomatesLab va s'exécuter sur des machines clientes uniquement. Il n'y a pas de système d'exploitation cible, le logiciel vise à être multiplateforme car nous utilisons Java (JDK 19.0.1). Il y a une interface externe du logiciel, avec le gestionnaire de fichiers du système d'exploitation de la machine. Les ressources partagées avec d'autres logiciels sont le temps CPU alloué, l'espace mémoire RAM et l'espace disque. AutomatesLab ne fait pas d'accès réseau et est uniquement contrôlé par l'utilisateur de la machine via ses IHM.

3.2. Exigences fonctionnelles

Le fonctionnement général du logiciel est résumé par le schéma UML use cases en Figure 2 ci-dessous.

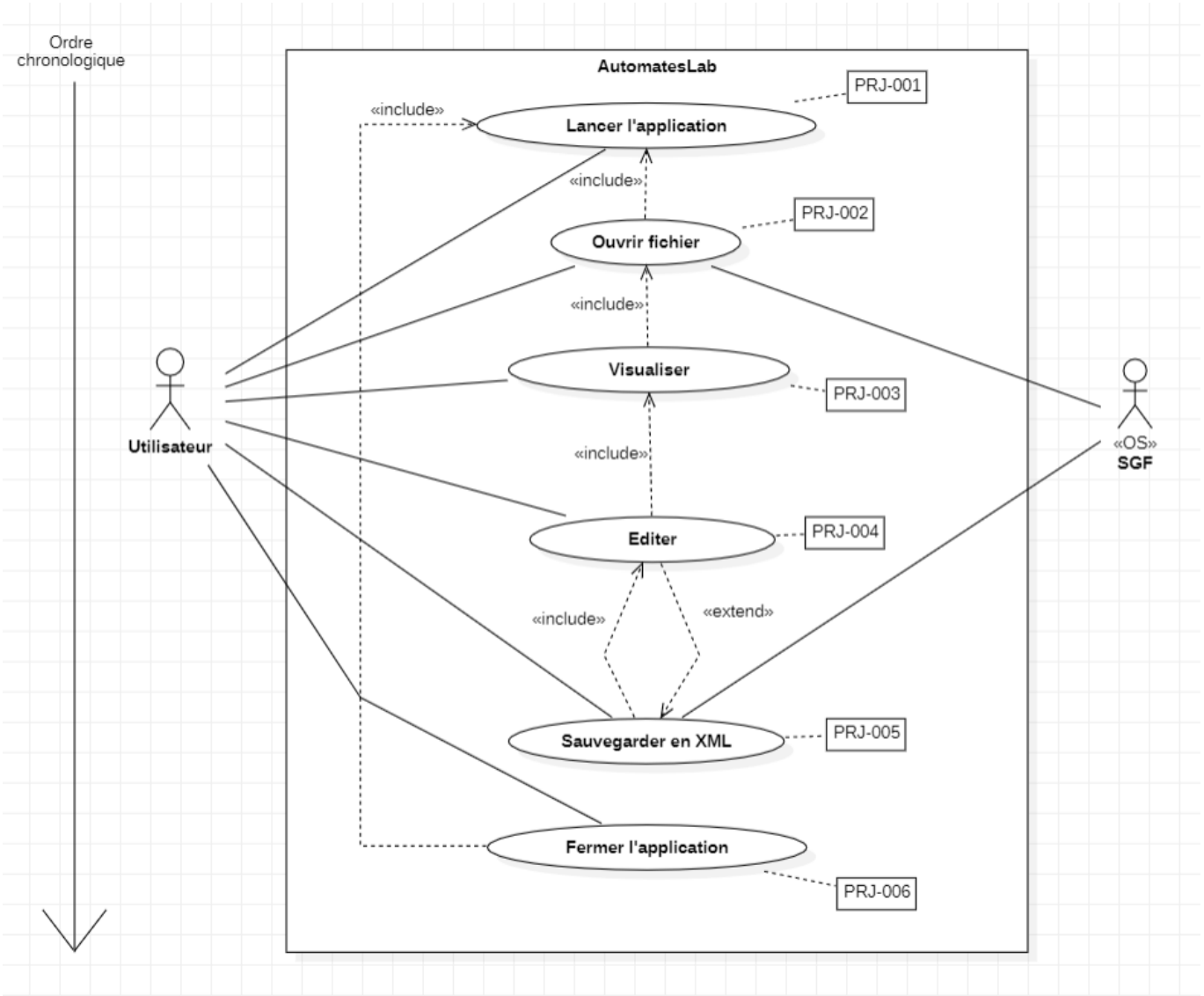

Fig. 2: Schéma use-cases général du logiciel

Les acteurs sont:

- Un utilisateur du logiciel AutomatesLab (acteur principal)
- Le SGF (acteur secondaire)

3.2.1. Ouverture/fermeture de l'application (PRJ-001 et PRJ-006)

3.2.1.1. Ouverture (PRJ-001)

A l'ouverture de l'application, la vue graphique vide s'ouvre.

3.2.1.2. Fermeture (PRJ-006)

La fermeture de l'application s'effectue lorsque l'utilisateur clique sur la croix en haut de la fenêtre (à gauche sous Windows et Linux, à droite sur macOS).

Si il y a un fichier ouvert dans l'éditeur, et que celui-ci contient des modifications qui ne sont pas enregistrées sur disque, une pop-up apparaît à l'écran: "Il y a des modifications non enregistrées. Souhaitez-vous enregistrer avant de fermer l'application ?" Et deux boutons "Oui" et "Non". Si l'utilisateur clique sur oui, le use case PRJ-005 "Enregistrer" se lance.
Puis l'application se ferme. Il n'y a pas de ressources temporaires conservées en mémoire ou sur disque.

3.2.2. Ouvrir un fichier (PRJ-002)

3.2.2.1. Description

Lorsque l'utilisateur a ouvert l'application, il a atterri sur l'interface graphique vide. En fonction de s'il veut créer un automate de zéro, ou charger un automate existant, le scénario correspondant va se lancer.

3.2.2.2. Scénario 1: Charger le fichier par défaut (PRJ-002-1)

Lorsque l'utilisateur sélectionne "Nouvel automate" dans le menu déroulant "Fichier", le logiciel charge le template XML d'Automate vide depuis le disque (la version la plus récente de ce fichier est consultable dans le document "Représentation XML.pdf", grande partie 4).
L'espace d'édition graphique s'ouvre sur l'interface du logiciel.

3.2.2.3. Scénario 2: Charger un automate depuis un fichier XML (PRJ-002-2)

L'utilisateur sélectionne l'option "Ouvrir…" dans le menu déroulant "Fichier". Une fenêtre contextuelle du SGF s'ouvre, permettant à l'utilisateur de choisir un fichier à charger. L'utilisateur sélectionne le fichier. Puis le scénario correspondant se lance.

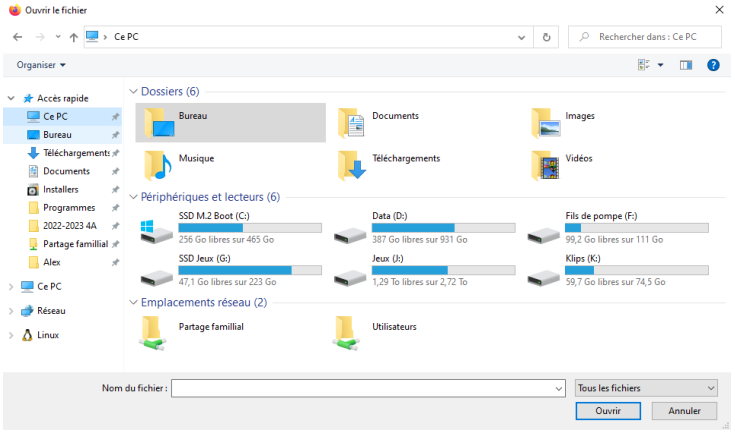

Fig. 3: Fenêtre d'ouverture de fichier, exemple de Windows 10

- Si le fichier que l'utilisateur tente d'ouvrir est corrompu, AutomatesLab fait apparaître une pop-up d'erreur "Fichier corrompu" et l'opération s'arrête.
- Si le fichier XML s'ouvre, mais ne correspond pas à la représentation XML d'un automate (spécifiée en Annexe "Représentation XML.pdf"), un pop-up d'information "Fichier ne représente pas un automate" s'affiche afin de prévenir l'utilisateur (la vue graphique ne sera pas disponible). Puis la vue XML s'ouvre afin que l'utilisateur puisse essayer de corriger le ou les problèmes.

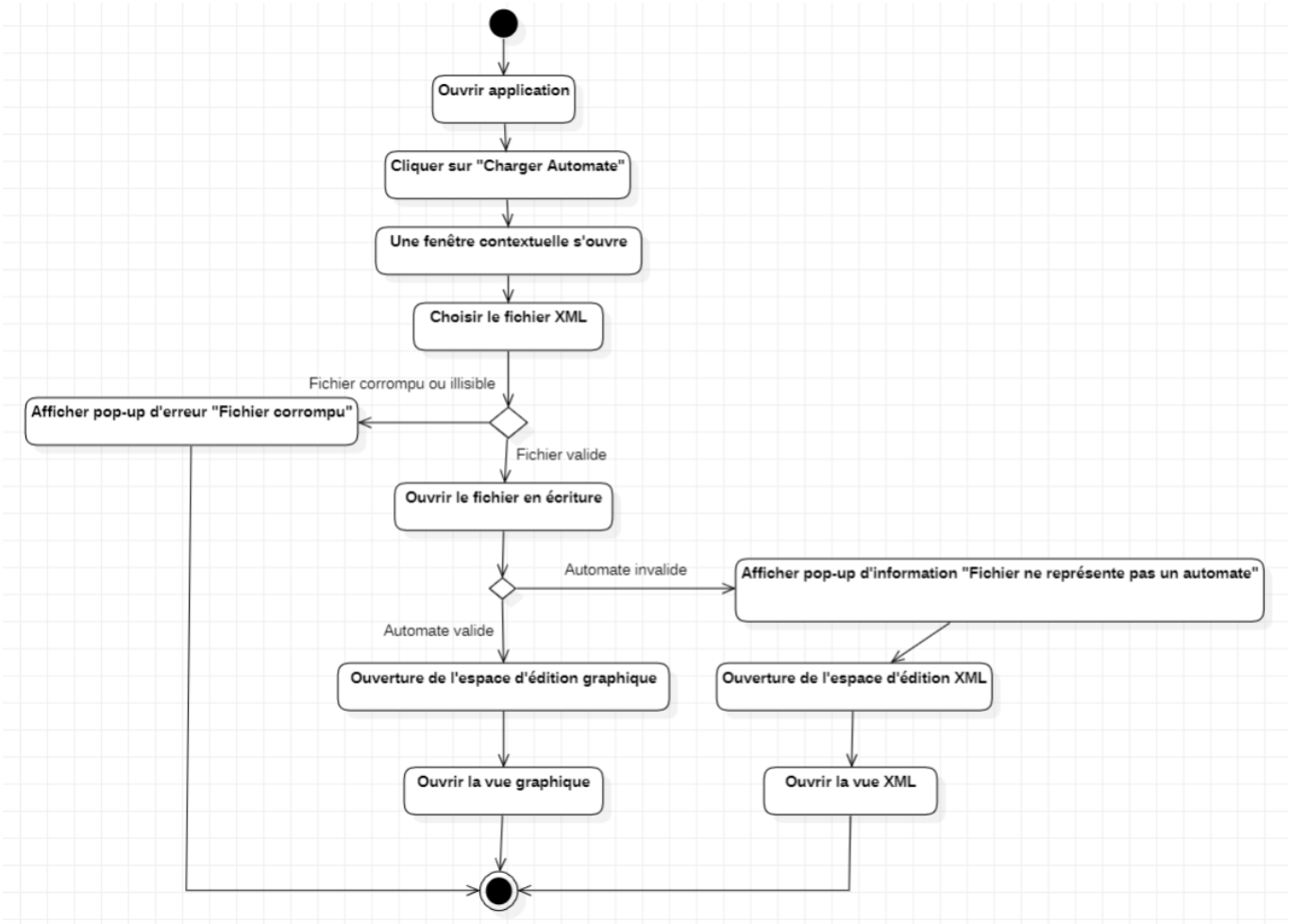

Fig. 4: Schéma détaillé du PRJ-002-2, charger un automate depuis un fichier XML

3.2.3. Visualiser un automate (PRJ-003)

3.2.3.1. Description

Lorsque l'utilisateur a choisi un automate à visualiser, le logiciel va l'afficher. L'utilisateur peut choisir entre plusieurs vues : une vue graphique et une vue XML.
L'utilisateur peut consulter ces affichages à l'aide d'onglets, positionnés sur la droite de la fenêtre de l'application. La vue représente la majeure partie de l'affichage, le reste étant la barre de menu contextuel. Cet écran est schématisé en Figure 5.
L'affichage commence lorsque l'utilisateur charge ou crée un automate, et se termine lorsqu'il ferme ledit automate ou l'application.

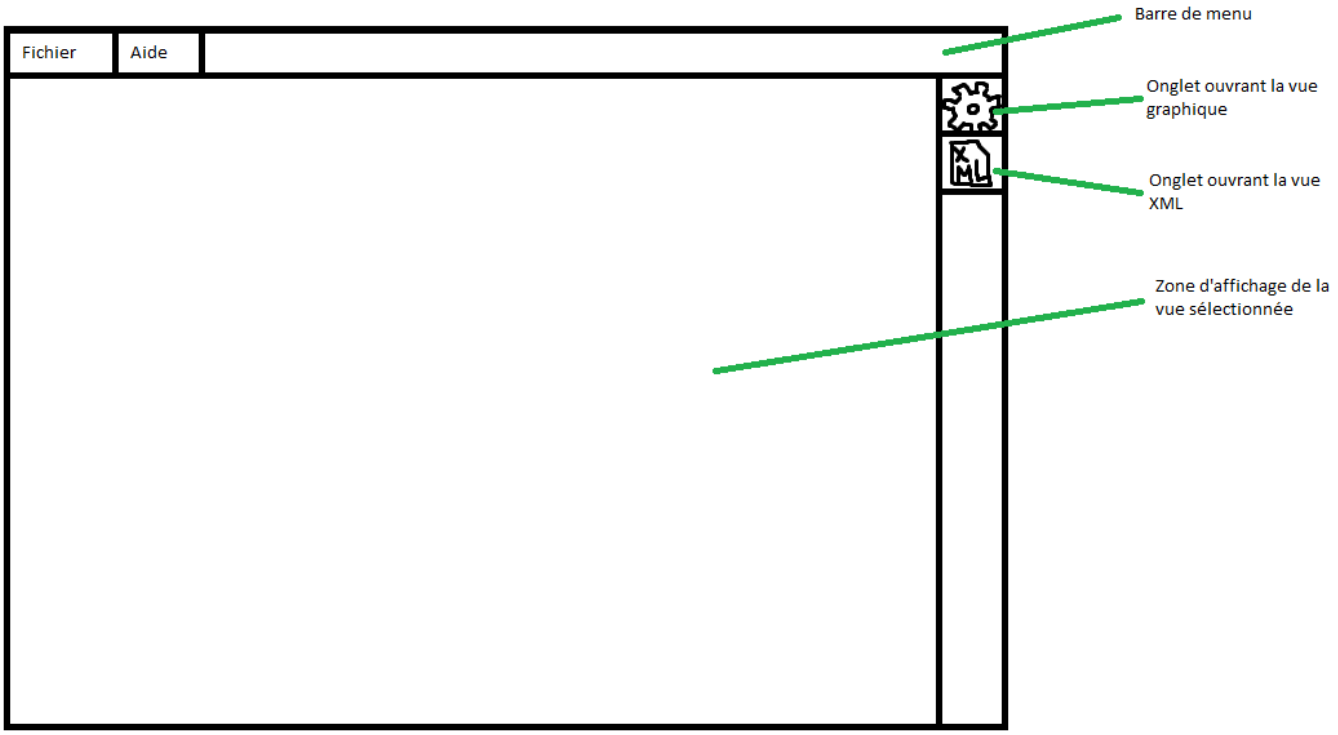

Fig 5: Schéma IHM de l'interface

3.2.3.2. Scénario 1: Vue graphique

Le logiciel affiche l'automate de manière graphique: un cercle pour chaque état, et des flèches associées de symboles correspondant aux transitions. Un état final est signifié par deux cercles concentriques, et l'état initial par une flèche spécifique. Une boîte à outils ("Toolbox") est située dans un panneau contextuel à gauche de la vue, et l'automate est représenté dans la partie centrale.

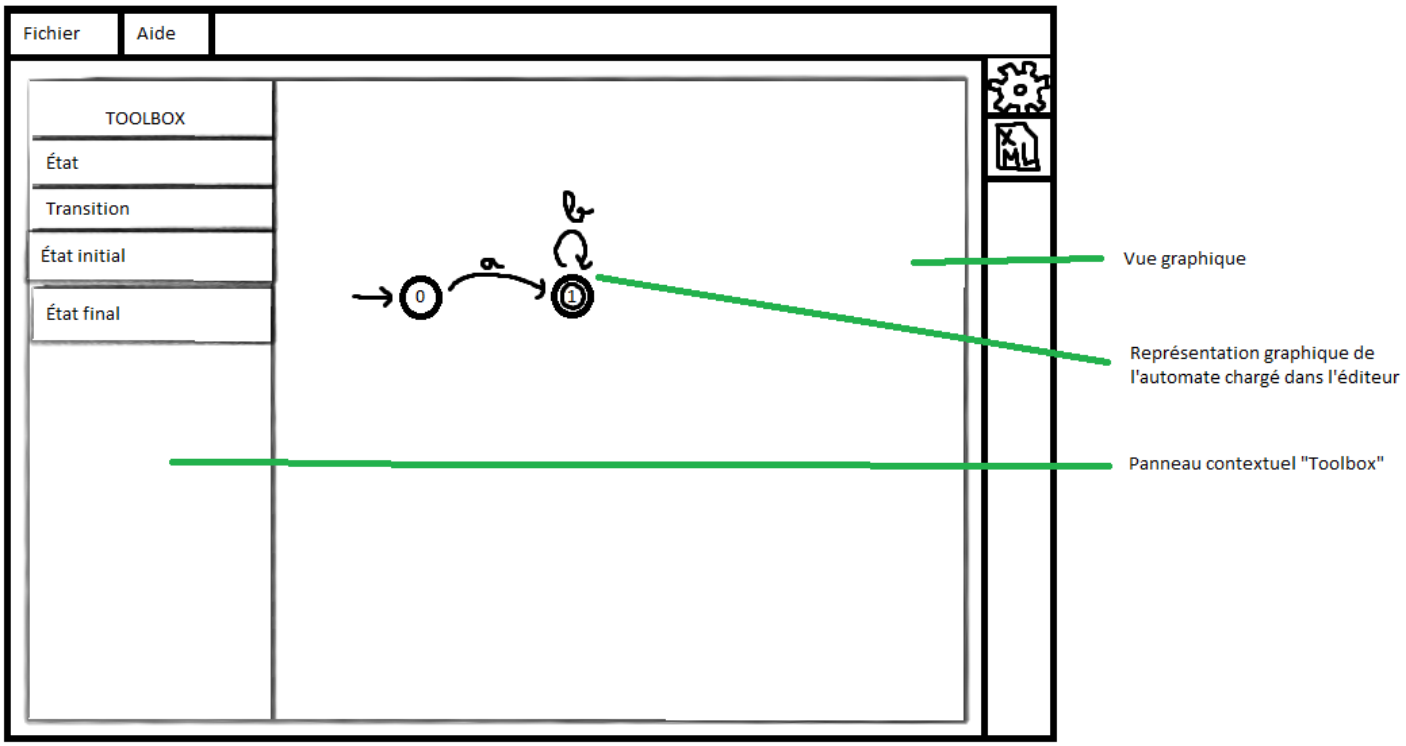

Fig. 6: Schéma IHM de l'interface, vue graphique

3.2.3.3. Scénario 2: Vue XML

Le logiciel affiche l'automate sous format XML: il montre le code du fichier XML de l'automate dans la zone d'espace dédiée à la vue. Cette zone correspond à une "inputBox".

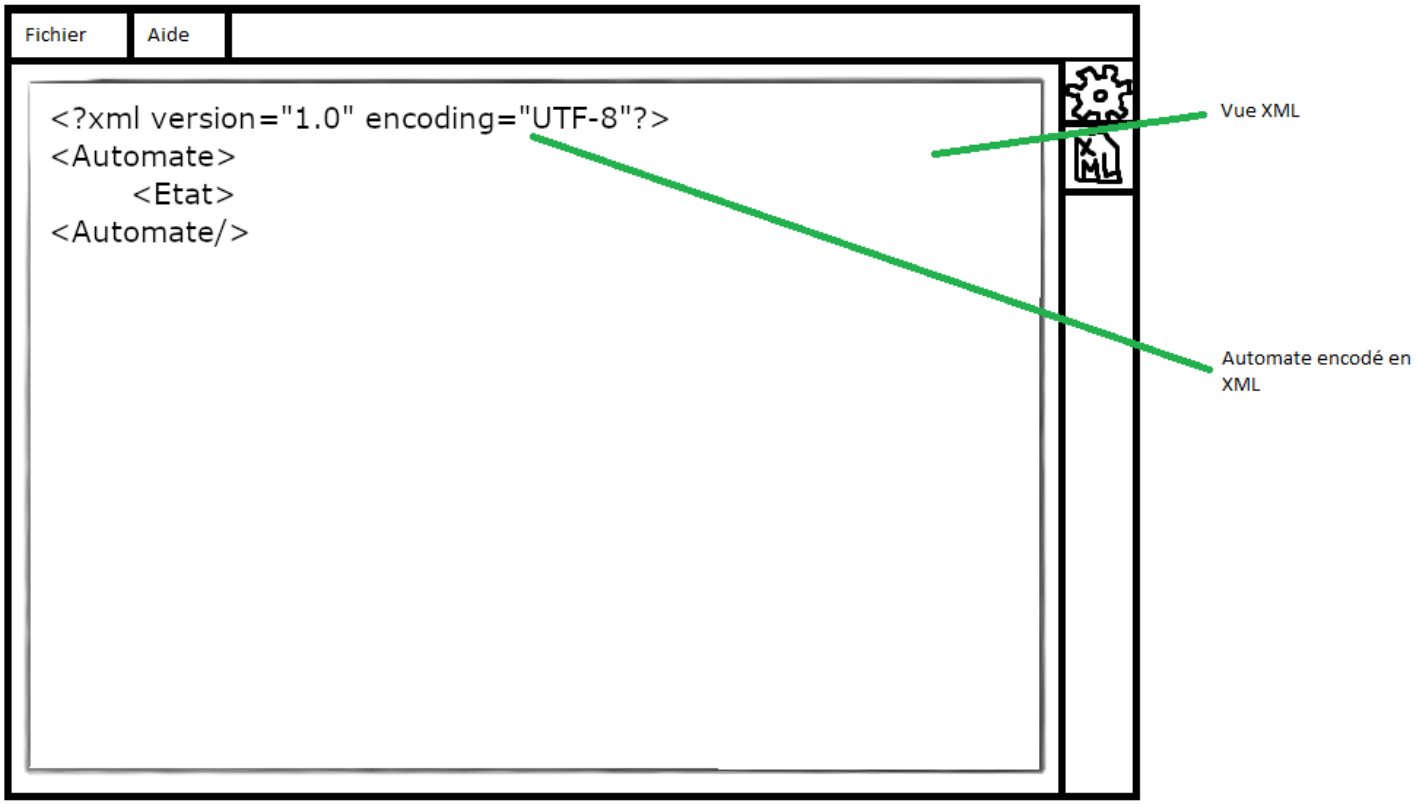

Fig. 7: Schéma IHM de l'interface, vue XML

Le fichier XML affiché ici ne correspond pas au format utilisé par l'application. Pour cela, se référer à l'annexe "Représentation XML.pdf"

3.2.4. Editer un automate (PRJ-004)

3.2.4.1. Description

L'utilisateur a créé ou chargé un automate dans le logiciel. En fonction de la vue qu'il utilise, il va pouvoir modifier l'automate de différentes façons.
Le traitement d'une opération demandée correspond à l'exécution de cette opération par le logiciel, qui se traduit par la modification correspondante sur le XML en mémoire représentant l'automate chargé.

3.2.4.2. Scénario 1: Modification en vue XML

L'utilisateur édite à l'aide de la vue graphique. Comme présenté auparavant en Figure 7, il a donc une interface de modification de texte. La boîte contient donc le XML représentant l'automate, et le logiciel se comporte comme un éditeur de texte.

3.2.4.3. Scénario 2: Modification en vue graphique

3.2.4.3.1. Description

L'utilisateur édite à l'aide de la vue graphique. Il dispose d'une boîte à outils, affichée dans une barre de menu contextuel sur la gauche de l'écran (voir IHM Fig. 5 présentée plus haut). Cette boîte à outils met à disposition de l'utilisateurs plusieurs fonctionnalités:

- Créer un état
- Créer une transition
- Définir un état comme initial
- Définir un état comme final

Les états et transitions sont des éléments sélectionnables. Sélectionner un élément le démarque des autres en l'encadrant à l'aide de la couleur d'accentuation de l'interface. Lorsqu'un élément est sélectionné, on peut:

- Le déplacer à l'aide d'un glisser déposer
- Le supprimer en appuyant sur la touche "Suppr", ou sur la touche "Backspace".

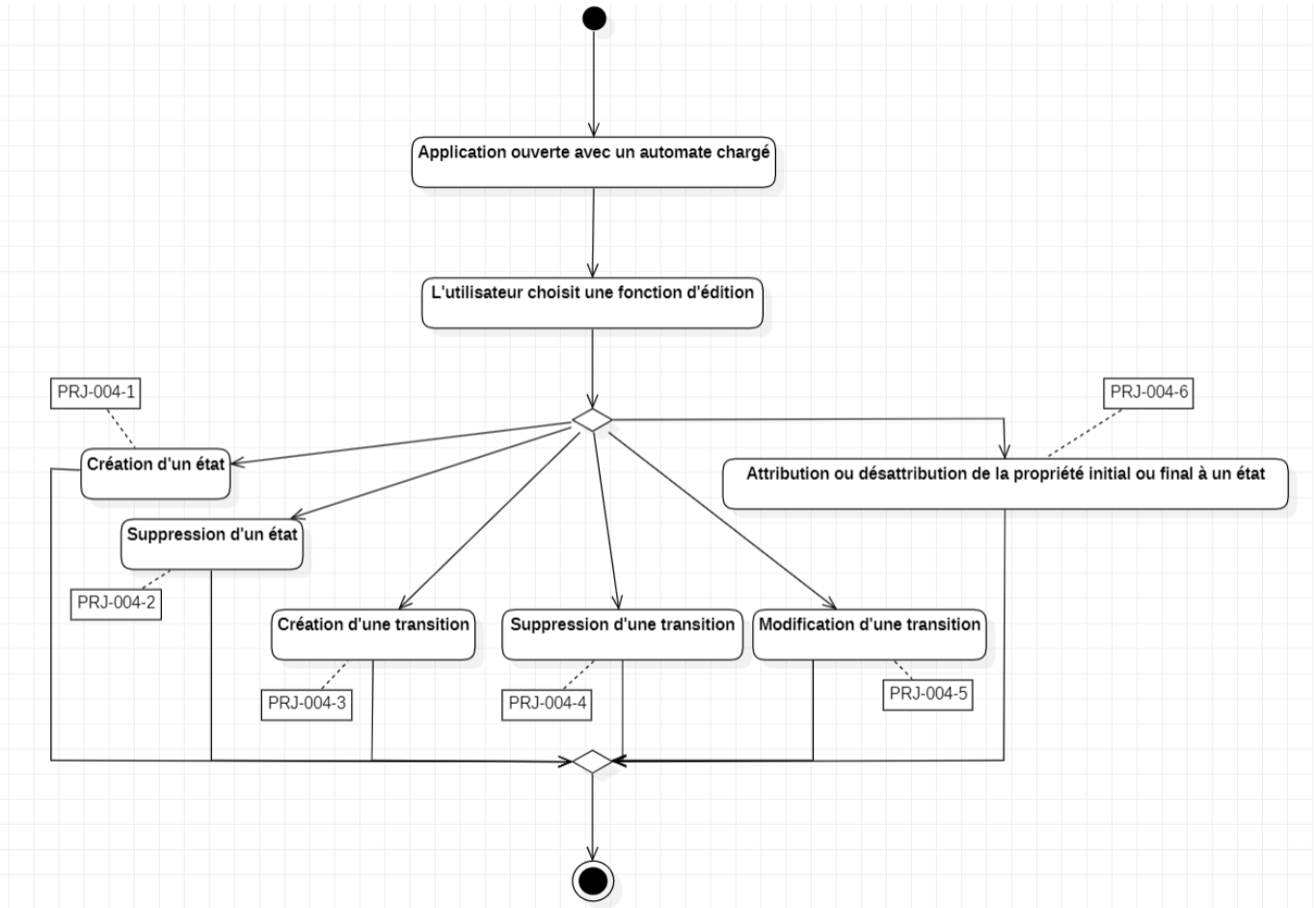

Fig 8: Schéma général du PRJ-004, édition d'un automate

3.2.4.3.2. Création d'un état (PRJ-004-1)

L'utilisateur sélectionne l'outil "État" de la boîte à outils. Il clique à l'emplacement où il souhaite créer l'état. L'état est créé (l'application modifie le XML chargé en mémoire), avec comme numéro le plus petit entier naturel disponible (non pris par un autre état).

3.2.4.3.3. Suppression d'un état (PRJ-004-2)

L'utilisateur sélectionne un état à supprimer, et il appuie sur la touche "suppr" ou "backspace". Le logiciel supprime alors l'état du XML de l'automate.

3.2.4.3.4. Création d'une transition (PRJ-004-3)

L'utilisateur sélectionne l'outil "Transition" de la boîte à outils. Il entre en mode création de transition. Il cliquer sur un premier état, qui sera l'état de départ de la transition, sur un second état (qui peut être le même que le premier) qui sera l'état d'arrivée de la transition. Puis une fenêtre contextuelle s'ouvre, contenant une zone de texte permettant à l'utilisateur d'entrer les symboles que doit accepter la transition. Si il ne rentre rien, la transition est créée en acceptant le mot vide.

La création ne peut s'effectuer qu'à condition que la transition demandée (d'un état donné vers un autre donné, pour un ou des symboles donnés) n'existe pas déjà. Si la transition d'un état vers un autre existe déjà, mais qu'elle ne comprend pas un des symboles de la transition à créer, le logiciel se contente de modifier la transition existante pour ajouter les nouveaux symboles.

Double cliquer sur une transition ouvre directement la fenêtre d'édition des symboles acceptés.

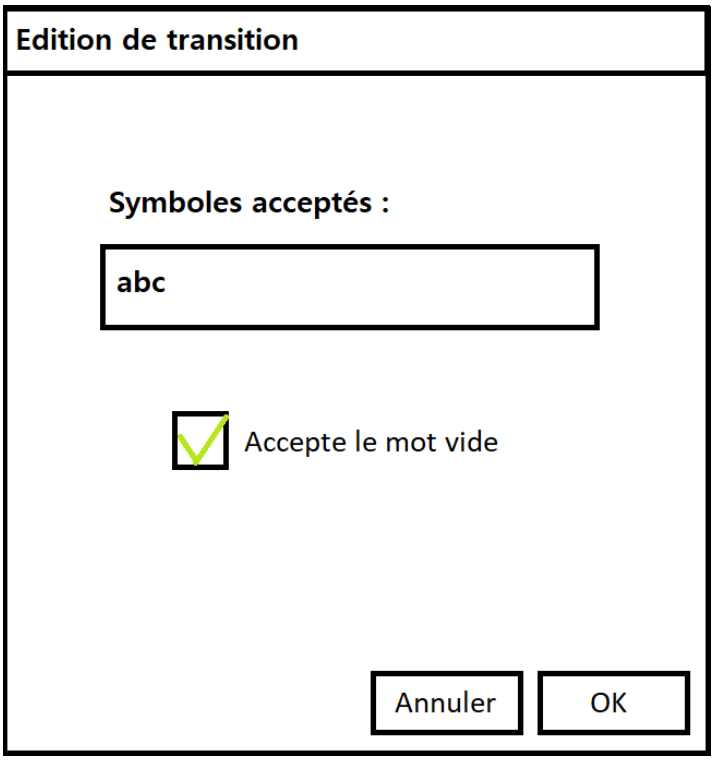

Fig. 9: Fenêtre d'édition de transition

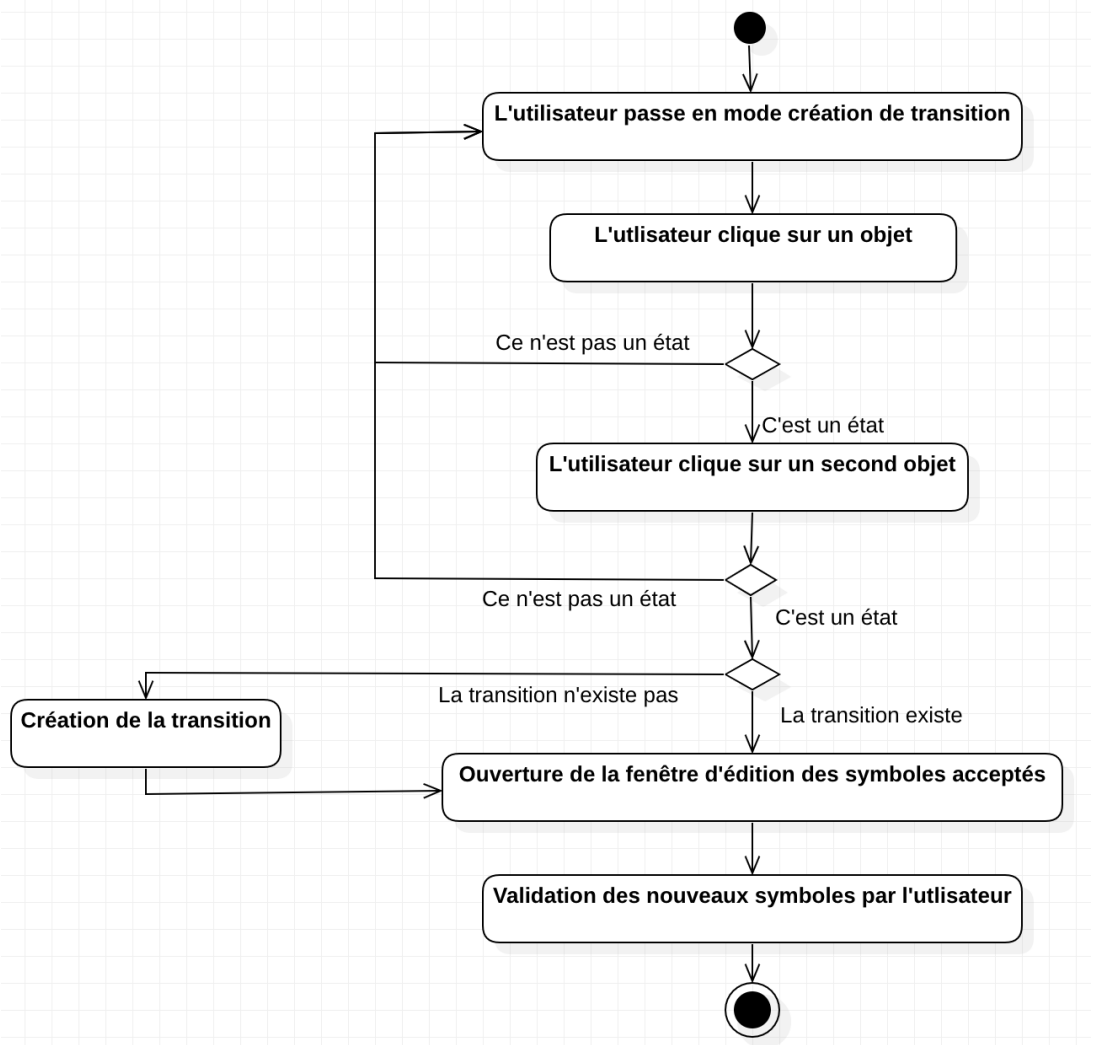

Fig 10: Schéma détaillé du PRJ-004-3, création d'une transition

3.2.4.3.5. Suppression d'une transition (PRJ-004-4)

L'utilisateur sélectionne une transition à supprimer, et il appuie sur la touche "suppr" ou "backspace". Le logiciel supprime alors l'état, ainsi que les transitions reliées à cet état, du XML de l'automate.

3.2.4.3.6. Modification d'une transition (PRJ-004-5)

Pour modifier une transition, l'utilisateur double-clique dessus. La fenêtre d'édition s'ouvre et il peut modifier les propriétés de la transition.

3.2.4.3.7. Attribution ou désattribution de la propriété initial/final à un état (PRJ-004-6)

L'utilisateur sélectionne l'outil correspondant à l'aide de la boîte à outils, puis clique sur l'état auquel il souhaite ajouter/enlever la propriété. L'application modifie alors le XML en conséquence.
Dans le cas où il existe déjà un état initial dans l'automate, celui-ci perd cet attribut, qui est réattribué à l'état demandé par l'utilisateur. Pour les états finaux, il peut y en avoir plusieurs, le problème ne se pose pas.

3.2.5. Sauvegarder un automate en fichier XML (PRJ-005)

3.2.5.1. Description

L'utilisateur a fini d'éditer son automate.
L'utilisateur sélectionne l'option "Enregistrer" (ou "Enregistrer sous") dans le menu déroulant "Fichier".
Le scénario correspondant se lance.
Avant d'enregistrer un automate sur le disque, on vérifie qu'il correspond bien la représentation spécifiée dans le dossier en annexe correspondant "Représentation XML.pdf".
Si ce n'est pas le cas, il sera tout de même enregistré mais ne pourra être ouvert qu'en vue XML.

3.2.5.2. Scénario 1: Enregistrer (PRJ-005-1)

Lorsque l'utilisateur sélectionne cette option, le logiciel utilise l'emplacement du fichier XML de l'automate courant pour enregistrer l'automate (au format XML) en mémoire vers le disque. Si il n'y a pas d'emplacement de fichier pour l'automate courant, on lance le scénario 2.

3.2.5.3. Scénario 2: Enregistrer sous (PRJ-005-2)

L'utilisateur a sélectionné l'option "Enregistrer sous".
Une fenêtre contextuelle d'enregistrement de fichier du SGF s'ouvre.
L'utilisateur sélectionne l'emplacement où enregistrer le fichier et le nomme.
Le logiciel tente d'enregistrer le fichier à l'emplacement fourni. Tant que l'opération échoue (ou jusqu'à ce que l'utilisateur annule l'opération), une pop-up "Erreur lors de l'enregistrement" le signifie à l'utilisateur ; on retourne à l'étape précédente.
La fenêtre contextuelle se referme.

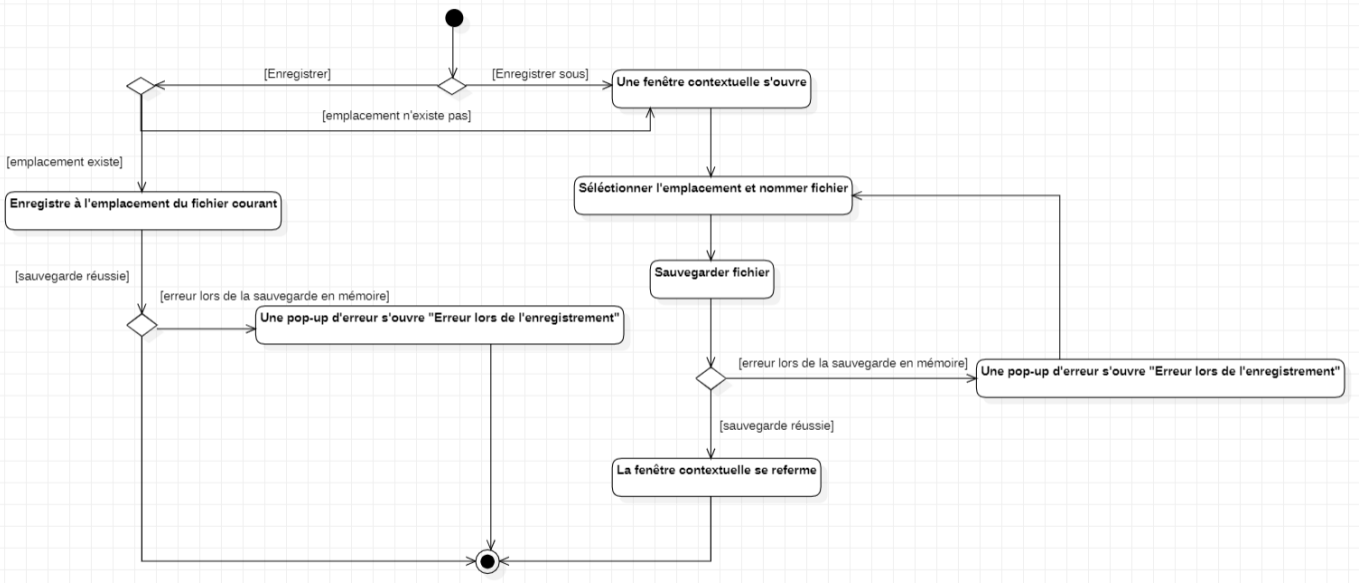

Fig 11: Schéma détaillé du PRJ-005, sauvegarder un automate au format XML

3.3. Exigences opérationnelles

3.3.1. Environnement

3.3.1.1. Environnement matériel

Nous utilisons nos ordinateurs personnels. Nous considérerons l'ordinateur le moins puissant comme le minimum requis.

Ses spécifications sont :

- 8 Go de mémoire
- CPU 4 cœurs, 8 threads @2.00 GHz

Le stockage nécessaire pour faire fonctionner l'application est de 30 Mo.

3.3.1.2. Environnement logiciel

L'implémentation se faisant en Java, notre logiciel interagit avec la JVM. Étant donné les fonctionnalités de cette dernière, il s'agit du seul logiciel avec lequel il interagit. La JVM interagit uniquement avec le système d'exploitation, pour les IHM et les accès au SGF. Ainsi, il est impératif que le JRE version 19 soit installé.

3.3.2. Sécurité

Les accès aux fichiers du SGF sont pris en charge par la JVM.
Seul l'utilisateur est responsable de l'emplacement de sauvegarde de ses fichiers. Le logiciel n'a pas le droit de créer ou supprimer de fichier (par la JVM) sans que l'ordre en soit donné par l'utilisateur.

3.4. Interfaces

3.4.1. Interfaces avec des fichiers

AutomatesLab a les mêmes droits d'accès qu'un utilisateur lambda. Il a accès à l'ensemble des fichiers que l'utilisateur choisit d'ouvrir à l'aide du logiciel, en lecture et en écriture. Cependant, il sera uniquement capable de lire des fichiers XML.

3.4.2. Interface Homme / Machine

Un utilisateur d'AutomatesLab doit connaître les principes de base des automates, les différents types possibles d'automates et potentiellement les algorithmes applicables à ces automates.
L'utilisateur n'a pas besoin de maîtriser intégralement le système d'exploitation, des connaissances basiques sur comment l'utiliser pour lancer un logiciel sera tout ce dont il a besoin. Il n'y a pas besoin d'autorisations particulières du système d'exploitation (il n'est pas nécessaire d'exécuter le logiciel en mode administrateur par exemple).

L'application est composée d'une fenêtre redimensionnable, pouvant être passée en arrière-plan. C'est une fenêtre standard (fenêtre classique avec fonctions "réduire", "agrandir", et "fermer"). La taille par défaut de cette fenêtre est de 1080x720, se lançant au centre de l'écran actif. Des pop-up d'information (détaillées dans les use-case) pourront apparaître au cours de l'utilisation du logiciel.

En ce qui concerne la charte graphique, les couleurs se voudront foncées afin de constituer un thème sombre. L'interface sera donc constituée d'une palette de plusieurs gris foncés et les textes seront écrits en blanc. Il y aura aussi une couleur d'accent rouge-orangé. Elle servira à appuyer certaines fonctionnalités de l'interface. Par exemple, quel onglet d'un menu est sélectionné ou quel bouton est activé.

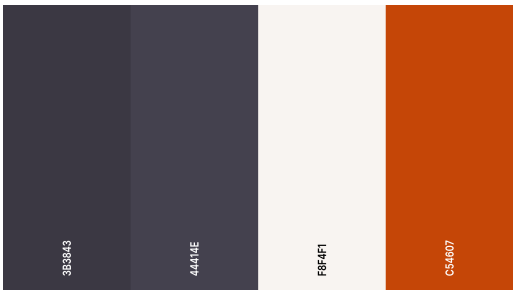

Fig. 12: Palette de couleur

Les interfaces ressembleront aux schémas déjà présentés aux sections précédentes (les couleurs ne sont pas les couleurs finales).

- Fig 5: Schéma IHM de l'interface
- Fig. 6: Schéma IHM de l'interface, vue graphique
- Fig. 7: Schéma IHM de l'interface, vue XML

L'interface comporte une barre de menu en haut de la fenêtre. Ce menu est décrit par le schéma suivant:

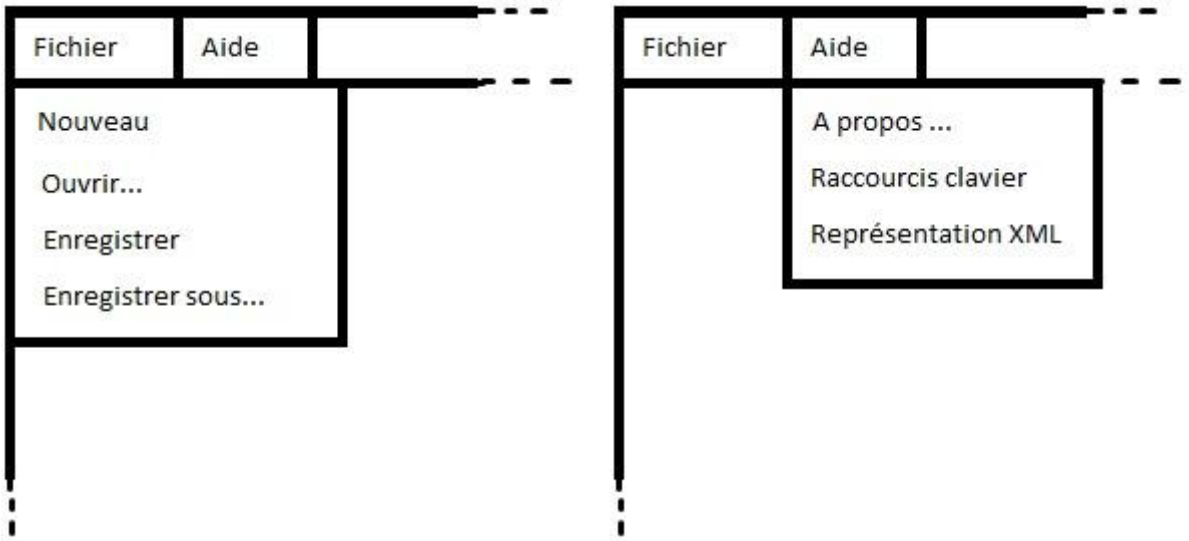

Fig. 13: Schéma de la barre de menu

Ces boutons ont pour action:

- "Menu fichier"
  - "Nouveau": Créé un nouvel automate (PRJ-001)
  - "Ouvrir…": Charge un automate depuis un fichier XML (PRJ-002)
  - "Enregistrer": Sauvegarde un automate sur disque (PRJ-005-1)
  - "Enregistrer sous…": Sauvegarde comme un nouveau fichier (PRJ-005-2)
- "Aide"
  - "A propos": Ouvre une fenêtre donnant des informations concernant le logiciel (nom, version, date, auteurs)
  - "Raccourcis clavier": Ouvre une fenêtre donnant les raccourcis clavier de l'application
  - "Représentation XML": Ouvre le fichier de documentation "Représentation XML.pdf"

Le menu A propos ressemblera à cela:

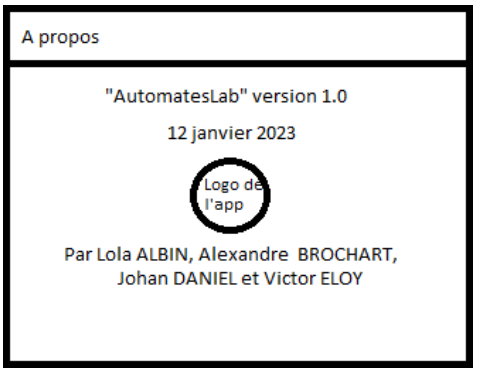

Fig. 14: Fenêtre "A propos"

Et le menu Raccourcis clavier à ceci:

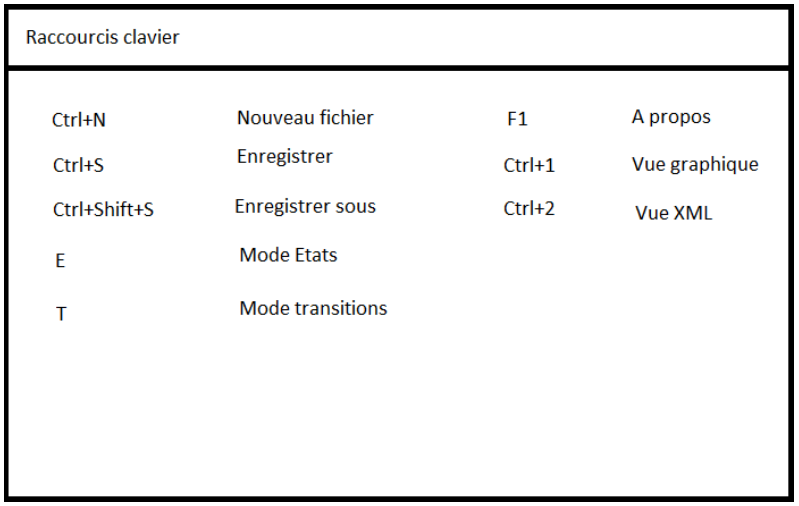

Fig. 15: Menu "Raccourcis clavier"

3.5. Exigences concernant la conception et la réalisation

Notre application se doit d'être stable. Il faut également qu'elle soit réactive et performante. Lorsque l'on interagit avec l'interface utilisateur, l'action demandée doit être réalisée rapidement (moins d'une demi-seconde): le plus proche possible de l'instantané.
Enfin, elle doit être portable, c'est-à-dire fonctionner sur toute machine personnelle, quelques soient ses caractéristiques. L'utilisation de Java comme langage de programmation garantit cette fonction ; AutomatesLab fonctionnera sur toutes les machines ayant le JRE version 19 ou supérieur d'installé, et ce quel que soit le système d'exploitation.