INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Encadrants

Gildas AVOINE

Barbara KORDY

Logiciel d'analyse de sécurité

Application au réseau STAR

Rapport de pré-étude

Étudiants

Pierre-Marie AIRIAU

Valentin ESMIEU

Hoel KERVADEC

Maud LERAY

Florent MALLARD

Corentin NICOLE

Résumé

Est-il possible de paralyser le Service des Transports en commun de l'Agglomération Rennaise (STAR) ? L'analyse de la sécurité d'un tel système permet la mise en évidence de ses potentielles failles de sécurité. Dans cette optique, nous allons développer un logiciel destiné à faciliter la tâche des experts en sécurité. Ces derniers pourront ainsi visualiser facilement les différentes attaques possibles et déterminer si leurs défenses sont adaptées.

Ce rapport de pré-étude présente les objectifs et le contexte de notre sujet, dresse un bilan de l'existant, élabore notre cahier des charges, et aborde la planification de notre projet.

23 octobre 2014

Table des matières

1 Introduction

Un long chemin a été parcouru depuis le 18 mars 1662, où le brillant penseur Blaise Pascal réalise la première expérience de transports en commun urbains au monde. Il s'agit alors de sept carrosses publics qui sont mis en service entre la Porte Saint-Antoine et le palais du Luxembourg à Paris. Depuis lors, les transports en commun ont beaucoup évolué, et sont aujourd'hui indispensables au bon fonctionnement d'une ville. En témoignent ces chiffres impressionnants : selon les estimations du Groupement des Autorités Responsables de Transport (GART), 6.5 millions de trajets sont réalisés par jour en France [**?**]. À Rennes, l'agglomération sur laquelle portera notre étude, la société Keolis dénombre en moyenne 250 000 trajets par jour [**?**].

Nous comprenons donc aisément l'importance que revêt le bon fonctionnement de ces transports publics en métropole. Ils génèrent des retombées économiques, réduisent de manière significative les embouteillages urbains et participent à l'amélioration de la qualité de vie de la population. Par exemple, à Rennes, la ligne A du métro permet d'économiser 11 000 tonnes de CO_2 par an [**?**].

Une paralysie de ces transports publics aurait une incidence considérable sur l'ensemble de la métropole. Il serait donc intéressant de réaliser une étude de sécurité pour prévoir les risques les plus importants et mettre au point des défenses efficaces.

2 Objectifs

En se basant sur le concept des arbres d'attaque et de défense (concept des ADTrees) décrit dans la Section 4, nous allons développer une application destinée à faciliter la tâche des experts en sécurité. Nous partons du principe que les utilisateurs de notre application maîtrisent les ADTrees, ou du moins n'y sont pas totalement étrangers.

L'étude des transports publics urbains rennais nous permettra d'illustrer notre projet. Ceci nous servira d'exemple, mais notre logiciel pourra être utilisé de manière plus générale. Il pourra modéliser un certain nombre de problèmes de sécurité, en prenant en compte différents paramètres : profil de l'attaquant, cible visée, etc.

3 La sécurité des transports en commun

3.1 Un contexte mondial

Le bon fonctionnement des transports en commun dépend de nombreux facteurs. Qu'il soit humain ou bien technique, le moindre dysfonctionnement impactera le système entier très rapidement. Ainsi, un simple tour d'horizon de la presse internationale fait vite remonter à la surface de nombreux cas de paralysie des transports publics urbains dans le monde entier. Nous avons choisi d'évoquer ci-dessous les cas les plus marquants, ainsi que les plus fréquents, parmi ceux que nous avons trouvés.

Les cas de paralysie les plus notables sont ceux impliquant des dégâts humains parmi les passagers. Il s'agit pour la plupart d'attaques terroristes. Bien que relativement rares, le lourd bilan humain de ces attentats marque durablement les esprits. C'est le cas de l'attentat à la bombe

dans la gare Saint-Michel du train inter-urbain parisien, le 25 juillet 1995, qui provoqua la mort de 8 personnes [**?**]. L'attaque la plus marquante de l'histoire fut celle du 7 juillet 2005 dans la ville de Londres : quatre attentats-suicides dans différentes rames de métro provoquent la mort de cinquante-six personnes et en blessent sept cents autres [**?**].

Si ces attaques sont impressionnantes, elles restent minoritaires. Parmi les autres cas de paralysie que nous avons recensés, nous pouvons constater que la majorité sont provoqués par le personnel des compagnies gérant ces transports. En effet, lors de conflits sociaux, le personnel possède un énorme moyen de pression sur la hiérarchie car il peut bloquer l'intégralité du trafic en se mettant en grève. Ce fut le cas à Londres, où un plan de fermeture des guichets du métro a provoqué une grève massive le 28 avril 2014. Ce mouvement social a entraîné la fermeture des deux tiers des stations et la diminution du trafic de 50% [**?**]. Ce fut aussi le cas à San Francisco, en juillet et octobre 2013, où une grève du réseau ferré inter-urbain (fort de ses quatre cent mille passagers quotidiens) a paralysé la ville pendant plusieurs jours [**?**]. Des exemples peuvent aussi être cités en France, comme à Marseille en décembre 2013, où les conducteurs ont réussi à bloquer la quasi-totalité du réseau pendant plus de deux jours [**?**]. Plus marquant, à Brest en juillet 2014, le tramway fut attaqué dans la nuit par une douzaine de personnes à coups de cocktails Molotov et de jets de pierres. Bien que personne n'ait été blessé, l'incident a fortement choqué la population brestoise [**?**].

L'informatisation des systèmes de transports crée une vulnérabilité supplémentaire. Les cartes d'abonnement des voyageurs peuvent être détournées, dans le but de récupérer des informations personnelles ou de voyager gratuitement. Par exemple, en 2008, des étudiants de l'université d'Amsterdam ont réussi à percer la sécurité d'un nouveau système de carte mise en projet par le gouvernement néerlandais [**?**].

Les serveurs informatiques sont aussi vulnérables. Fin 2008, les serveurs du service de transport bruxellois, la STIB, ont été victimes d'une attaque informatique [**?**]. Cette attaque fut vite repoussée par les ingénieurs en informatique de la société. Comme de nombreux services vitaux du réseau de transports (tels que cartes d'abonnement, coordination des bus, etc.) dépendent de son système d'information, une attaque contre ce dernier pourrait provoquer de graves dysfonctionnements.

Pour illustrer notre projet, nous avons choisi de nous appuyer sur l'étude du réseau de transports publics de Rennes Métropole. Nous allons donc maintenant le décrire succinctement, afin de mieux comprendre par la suite le cadre de l'étude ainsi que les termes utilisés.

3.2 Le cas rennais

Le réseau de transports publics rennais est constitué d'un réseau de bus et d'une ligne de métro automatique [1]. Un système de vélos en libre-service a également été mis en place depuis quelques années. Tous ces dispositifs sont gérés par le Service des Transports en commun de l'Agglomération Rennaise (Star), qui dépend de la société Keolis Rennes. Les usagers peuvent accéder aux différents services par plusieurs moyens : en achetant des tickets à l'unité, ou en utilisant une carte d'abonné rechargeable (carte Korrigo).

Les bus sont équipés de huit cent soixante-dix écrans, diffusant entre autres le nom des prochains arrêts du bus, les stations de vélos Star à proximité, mais aussi quelques publicités. Des informations similaires sont visibles sur soixante-dix écrans dans les stations de métro. Les utilisateurs ont également à leur disposition cinquante bornes d'information aux voyageurs dans les

1. Une deuxième ligne de métro est actuellement en construction

abribus. Un système d'aide à l'exploitation et à l'information des voyageurs permet d'indiquer en temps réel le passage du prochain bus, les perturbations, les correspondances, la disponibilité des vélos STAR, etc [**?**]. Toute cette organisation n'est cependant pas à l'abri des incidents, et présente quelques failles. Voici un récapitulatif des exemples les plus importants que nous avons trouvés dans la presse.

En avril 2012, les premières attaques volontaires sont relevées : le réseau STAR a été entièrement paralysé en pleine heure de pointe suite à l'agression d'un chauffeur. À l'époque, la direction recensait dix-huit agressions depuis le début de l'année, et promettait un redéploiement de ses agents de médiation et de prévention. Une mesure insuffisante pour la Confédération Française Démocratique du Travail (CFDT), qui réclamait « une police dédiée aux transports » [**?**]. Environ un mois plus tard, en mai, la ligne de métro a été entièrement bloquée dès 5h30 pendant toute une matinée : un problème informatique entre le centre de commandement du métro et les rames empêchait la liaison qui permet de contrôler les rames à distance. Des bus relais ont été mis en place pour desservir les stations, jusqu'à ce que la circulation du métro reprenne vers 14h [**?**].

La ligne de métro a été paralysée en mai 2006 durant quatre heures, par deux chiens errants sur les voies. Le STAR ne s'explique pas comment les chiens ont pu pénétrer dans le tunnel [**?**]. Dans la même catégorie d'attaques non volontaires, en juillet 2009, la ligne de métro a été bloquée pendant près de vingt heures à la suite d'un violent orage ayant provoqué l'inondation des voies de circulation [**?**].

Depuis 2012, des incidents se produisent régulièrement sur le réseau mais ne conduisent pas à l'arrêt du système. Il s'agit principalement d'agressions du personnel, comme ce fut le cas le 5 octobre 2014 où un contrôleur a reçu un coup de poing pendant l'exercice de ses fonctions [**?**]. Ces quelques faits montrent bien l'importance pour le STAR de la mise en place d'un outil d'évaluation des risques.

4 Arbres d'attaque et de défense

4.1 Méthodologie

Le concept des arbres d'attaque a été introduit en 1999 par Bruce Schneier, un expert américain en sécurité informatique qui est parti du constat que des systèmes réputés « inviolables » se font inévitablement briser [**?**]. De plus, ces systèmes sont brisés par des méthodes d'accès qui n'avaient pas été imaginées par leurs concepteurs, car ces derniers n'avaient pas les outils nécessaires pour dresser une liste exhaustive des attaques possibles. Schneier a donc créé le concept des arbres d'attaque dans ce but : pouvoir réaliser un inventaire complet des méthodes d'attaque sur un système, quel qu'il soit, afin de pouvoir en concevoir la défense de la manière la plus adaptée. Schneier a lui-même conçu son modèle d'arbre d'attaque à partir du principe des « arbres de défaillances », une méthodologie datant du début des années soixante dont le but est d'évaluer l'impact de la défaillance d'un composant le système qui l'englobe [**?**].

Lors de ses recherches, Schneier a retenu une représentation des menaces sous la forme d'arbres, expliquée par la Figure 1. Ces arbres modélisent les pré-requis nécessaires à la réalisation de l'objectif principal (dans notre exemple, accéder au compte en banque d'une personne). Pour cela, on représente d'abord cet objectif à la racine de l'arbre, puis on le décompose en objectifs intermédiaires, appelés « nœuds ». Ces derniers sont ensuite re-décomposés de la même façon en répétant l'opération autant de fois que nécessaire. Les nœuds de l'arbre d'attaque sont par

convention représentés par des ovales reliés entre eux par des traits.

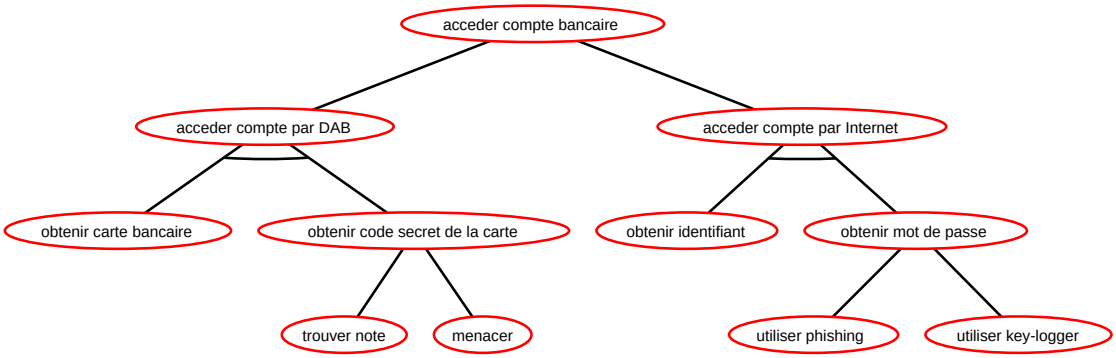

FIGURE 1 – Arbre d'attaque

La relation entre les fils d'un nœud peut prendre deux formes distinctes :
— la **disjonction**, qui correspond au cas où la validation d'un seul nœud fils suffit à valider le nœud père. Cette relation équivaut à l'opération logique « OU ». Dans notre exemple, accéder au compte en banque de quelqu'un peut se faire soit par un distributeur automatique de billets (DAB), soit par Internet.
— la **conjonction**, qui correspond au cas où la validation de l'ensemble des nœuds fils est nécessaire pour valider le nœud père. Cette relation équivaut à l'opération logique « ET ». On la représente par un arc de cercle reliant les relations père-fils. Par exemple, pour retirer de l'argent d'un compte bancaire depuis un DAB, il faut à la fois obtenir une carte bancaire et le code secret associé.

Enfin, le modèle des arbres d'attaque permet d'associer aux nœuds des paramètres représentant des informations quantitatives ou qualitatives sur l'action : coût, difficulté, probabilité, temps d'exécution, etc. Ces paramètres permettent de quantifier le poids du nœud dans l'arbre. Il est alors possible, en utilisant les fonctions appropriées, de propager ces paramètres au nœud père. Ceci permet à l'attaquant de choisir une stratégie d'attaque plutôt qu'une autre au vu de ses ressources. La Figure 2 illustre le coût financier minimum pour l'attaquant. On remarque dans ce cas précis que le coût d'un nœud disjonctif est le coût minimal de ses fils, tandis que le coût d'un nœud conjonctif en est la somme. Chaque quantification a un mode de calcul propre vis-à-vis du type de nœud (disjonctif ou conjonctif). Il est également possible de combiner plusieurs paramètres pour en produire de nouveaux. Par exemple, on peut obtenir le paramètre « risque » en multipliant la « probabilité de succès » de l'attaque par son « impact ».

Le concept des arbres d'attaque a évolué grâce à la contribution de personnes ayant amélioré le modèle de Schneier [**?**]. En particulier, le concept d'arbre d'attaque a été étendu à celui d'arbre d'attaque et de défense (« attack–defense tree » en anglais, ou « ADTree »). Les ADTrees représentent non seulement les (sous-)objectifs de l'attaquant mais aussi les défenses mises en place que l'attaquant aura besoin de désactiver pour atteindre son but [**?**]. Par convention, les défenses sont représentées par des rectangles. La Figure 3 représente un ADTree qui étend l'arbre d'attaque précédent avec des défenses.

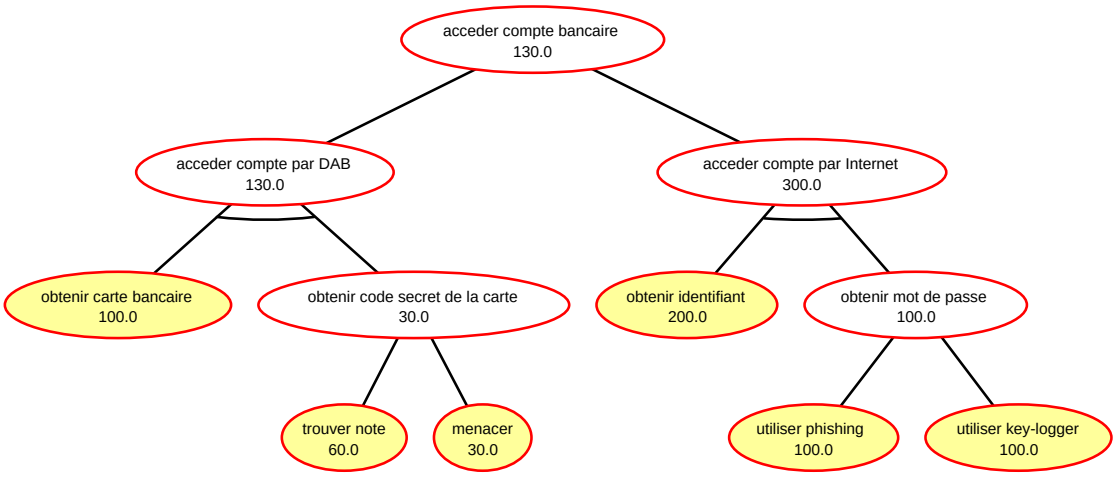

FIGURE 2 – Quantification du coût de l'attaque

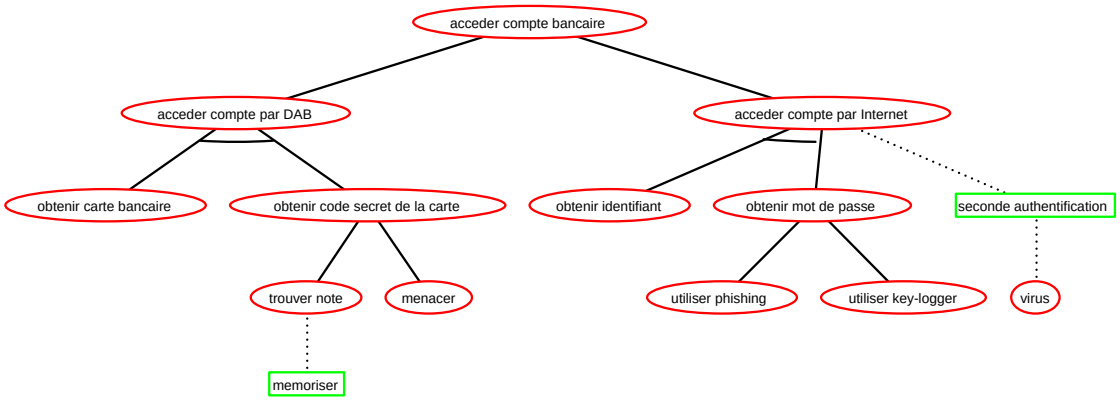

FIGURE 3 – Arbre d'attaque et de défense

4.2 Support logiciel des ADtrees

Aujourd'hui, les arbres d'attaque sont utilisés par de nombreuses entreprises pour réaliser leurs audits de sécurité. Bien qu'il existe différents logiciels les implémentant sur le marché (Securl-Tree [**?**], ATTACKTREE+ [**?**]), de nombreuses entreprises développent en interne leur propres outils de modélisation des arbres d'attaque. Ils peuvent ainsi l'adapter au mieux à leurs besoins.

ADTool [**?**] est le seul logiciel que nous avons identifié à implémenter les ADTrees. Il s'agit d'un logiciel libre développé par une équipe de chercheurs de l'université du Luxembourg. Selon nous, ce logiciel gagnerait à disposer de certaines fonctionnalités supplémentaires. Par exemple, ADTool ne prend pas en charge l'affichage simultané de plusieurs paramètres. Il ne permet pas non plus de faire du copier-coller d'arbres, ou encore de gérer dynamiquement la pondération des nœuds de l'arbre en fonction de l'attaquant.

Nous avons donc décidé de créer notre propre logiciel basé sur ADTool, en l'agrémentant de

fonctionnalités supplémentaires.

5 Cahier des charges

Comme indiqué dans la Section 2, notre principal objectif est de réaliser un logiciel aidant à l'analyse de la sécurité d'un système [2]. Destiné aux experts en sécurité des entreprises, notre logiciel utilisera le concept d'ADTrees décrit dans la Section 4. Il sera développé pour la plateforme Microsoft Windows.

Nous repartirons sur la base d'ADTool pour réaliser notre logiciel, afin de ne pas réécrire l'existant. Toutefois, diverses fonctionnalités que nous avons jugé utiles seront développées :

- trouver le chemin d'attaque optimal selon une fonction de synthèse, qui utilisera plusieurs paramètres (Section 5.1).
- filtrer l'arbre en fonction d'une fourchette de critères (Section 5.2).
- utiliser des modèles d'arbres généraux que l'utilisateur pourra ensuite modifier en fonction de sa situation (Section 5.3).
- améliorer l'édition des arbres avec ADTool, afin de la rendre plus souple et plus pratique (Section 5.4).
- créer un guide afin de proposer à l'utilisateur une méthode d'analyse (Section 5.5).

Pour finir, nous décrirons l'architecture de notre logiciel dans la Section 5.6.

5.1 Fonction de synthèse et chemin optimal

Dans la version actuelle d'ADTool, il est possible d'utiliser un seul paramètre à la fois sur un arbre. Pourtant, pouvoir faire une synthèse à partir de différents paramètres peut permettre de rechercher des compromis plus facilement. Par exemple, si on souhaite rechercher l'attaque la moins coûteuse financièrement, sans pour autant sacrifier le temps passé à réaliser notre attaque, on pourrait fournir une fonction définie par

$$synthese(cout, tps) = 2 * cout + tps.$$

Avec cette fonction, une nouvelle valuation de l'arbre sera calculée et on pourra ensuite l'élaguer pour garder uniquement les branches permettant de minimiser la fonction de synthèse. L'arbre élagué sera ensuite affiché, et l'utilisateur pourra prendre la décision de l'enregistrer comme un nouvel arbre.

Afin d'obtenir le plus grand choix de fonctions possibles, nous donnerons à l'utilisateur la possibilité de combiner les différents paramètres de façon linéaire, polynomiale, et aussi d'utiliser des fonctions mathématiques « standards », comme l'exponentielle, le logarithme, etc.

5.2 Filtre à critères

Une autre fonctionnalité que nous développerons sera le filtre à critères. Les critères consistent en un ensemble de valeurs autorisées pour nos paramètres, et seront gardés uniquement les nœuds

2. La nature du dit système peut être très vaste, allant d'un simple distributeur de billets au réseau de transport de toute une ville.

respectant ces règles. Ceci permettra d'obtenir rapidement les attaques réalisables en fonction des ressources de l'attaquant, nous permettant de modéliser plusieurs attaquants théoriques [3].

Supposons qu'il existe trois sous-arbres A, B et C pour atteindre un objectif, avec un coût respectivement de 2596 €, 70333 € et 200 €. Le temps passé à réaliser l'attaque dure 1, 2 et 3 semaines. On voudrait garder uniquement les attaques qui coûtent moins de 30000 €, et qui prennent moins d'une semaine. Notre filtre supprimera donc les sous-arbres B et C, pour garder uniquement le A.

L'arbre obtenu à l'issue du filtrage pourra très bien être identique à celui fourni en entrée (toutes les attaques respectent les critères), ou au contraire être vide (aucune attaque n'est réalisable avec les moyens alloués). Ce nouvel arbre sera ensuite affiché à l'utilisateur, qui pourra décider de l'enregistrer comme un nouvel arbre.

5.3 Modèles généraux

Commencer une nouvelle étude peut parfois être difficile. Pouvoir se baser sur des études déjà réalisées pour les adapter à sa situation simplifie grandement la tâche. Ainsi, nous donnerons à l'utilisateur la possibilité de partir de modèles d'attaques existantes, qu'il devra ensuite éditer pour que l'arbre corresponde à son système. Dans notre cas d'étude, nous pourrions imaginer avoir un modèle générique d'attaque de réseau de transport en commun, pour ensuite l'éditer afin de prendre en compte les spécificités de l'agglomération rennaise et du STAR.

Notre logiciel intégrera divers arbres génériques, mais l'utilisateur pourra aussi en importer depuis d'autres sources (Internet, clé usb, etc.). Notre logiciel étoffera sa collection de modèles à chaque nouveau modèle importé par l'utilisateur. L'utilisateur sera aussi capable de créer ses propres modèles, pour s'en resservir dans d'autres de ses analyses, ou encore pour les publier sur Internet. Ainsi, on pourrait imaginer à terme un site web regroupant des milliers de modèles réalisés par un réseau d'experts en sécurité.

5.4 Amélioration d'ADTool

Bien qu'ADTool soit déjà très complet, nous avons identifié plusieurs fonctionnalités qui rendraient la manipulation des arbres plus aisée :

- Une fonction de couper / copier / coller d'arbres et de sous-arbres.
- Réaliser des glisser / déposer d'arbres et de sous-arbres.
- Importer et exporter des arbres dans un grand nombre de formats différents.
- Permettre la représentation d'un arbre avec plusieurs paramètre à la fois.

5.5 Guide

L'utilisateur n'étant pas nécessairement familier avec les ADTrees, le guide lui permettra de comprendre rapidement comment réaliser ses arbres, et comment réaliser son analyse.

Notre guide lui fera suivre une méthode générique [**?**], découpée en différentes étapes. Une fois que l'une d'entre elles sera complétée, le guide passera à l'étape suivante et expliquera précisément

3. Un étudiant et un syndicat du crime organisé n'auront pas les mêmes moyens financiers par exemple.

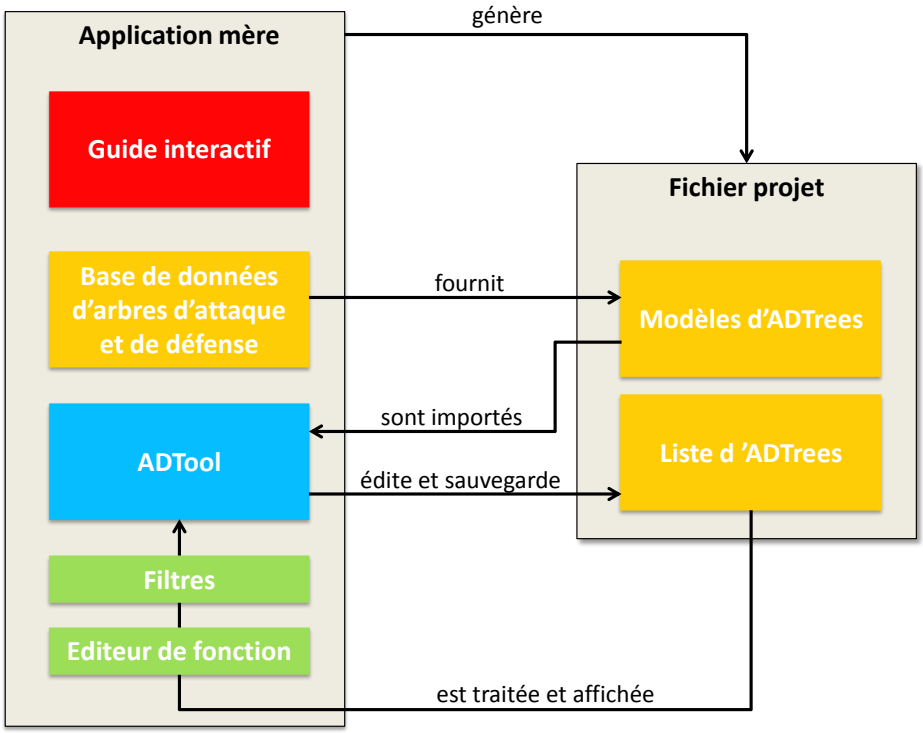

FIGURE 4 – Différents modules graviteront autour d'ADTool pour atteindre nos objectifs.

à l'utilisateur comment la réaliser. Le guide prendra la forme d'un « journal de quête », ce qui lui permettra de garder une trace des étapes déjà réalisées.

5.6 Architecture

L'architecture de notre logiciel est présentée sur la Figure 4.

Toutes les modifications ne seront pas intégrées directement dans ADTool (seules les modifications de la Section 5.4 y seront implémentées). À la place, nous auront une application mère, qui se servira d'ADTool comme étant un composant dédié à la visualisation et à l'édition des arbres.

L'analyse d'un système sera sauvegardée sous forme de projet. Celui-ci contiendra différents arbres utilisés pour l'analyse, ainsi qu'une bibliothèque de modèles que l'utilisateur pourra reprendre à tout moment pour créer un nouvel arbre (groupe "Fichier Projet"). À la création du projet, une série de questions sera posée à l'utilisateur. Les réponses obtenues permettront de sélectionner un sous-ensemble des modèles de la bibliothèque du logiciel, pour les précharger dans la bibliothèque du projet.

Les autres grandes parties de notre application (section 5.2, 5.3 et 5.5) seront implémentées sous

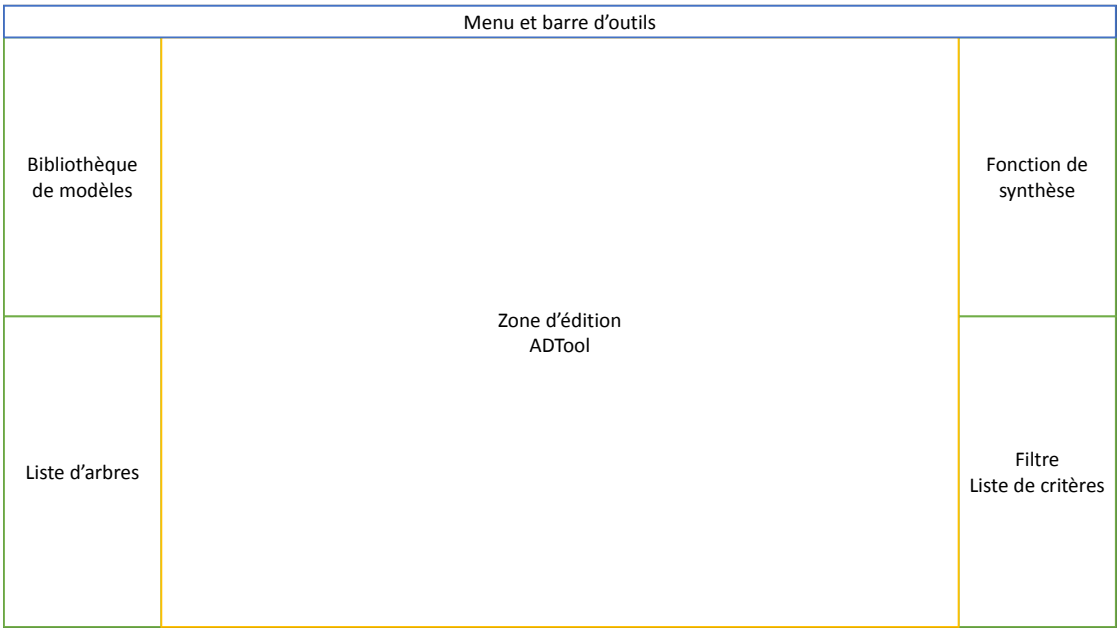

FIGURE 5 – L'interface sera gardée au plus simple.

forme de modules dédiés à leur tâche. La fonction de synthèse et le filtre seront deux modules qui prendront en entrée un arbre du fichier projet (et leurs autres paramètres respectifs), et enverront le résultat à ADTool pour affichage. L'utilisateur décidera ensuite de ce qu'il fera de l'arbre généré (sauvegarde, édition, export, etc.). Le guide supervisera l'ensemble de l'application, et devra fournir des informations et consignes pertinentes à l'utilisateur afin de l'aider dans son analyse.

Une ébauche de l'interface se situe Figure 5. Notons que le guide n'y figure pas, puisqu'il ne sera pas affiché en permanence pour ne pas surcharger l'interface. Il s'agira plutôt d'une fenêtre dédiée qui pourra être ouverte et fermée au besoin.

6 Outils

Nous avons choisi d'employer des outils répondant au mieux à notre cahier des charges tout en s'intégrant bien à notre formation. Ces différents outils sont décrits ci-après.

6.1 ADTool

ADTool est un logiciel libre et son développement a nécessité un travail conséquent. Il est donc pertinent de ne pas développer notre propre éditeur et afficheur d'ADTrees, mais plutôt d'utiliser ADTool pour qu'il remplisse ces fonctions. ADTool fera donc partie intégrante de notre logiciel, comme mentionné dans la Section 5.

6.2 Systèmes d'exploitation

Nous développerons notre application pour **Windows** en raison de sa très grande présence dans les entreprises. Nous souhaitons toutefois garder la porte ouverte à une compatibilité avec GNU/Linux et Mac OS, au cas où un futur portage serait envisagé. Ce portage n'est cependant pas notre priorité et reste purement optionnel. Ces considérations nous ont amené à choisir les langages de programmation décrits ci-dessous.

6.3 Langages de programmation et environnements de développement intégrés

Pour notre application, l'emploi du **C++** nous paraît pertinent. Bien que, de par sa nature, Java permette une compatibilité sur tous les systèmes d'exploitation, nous souhaitons éviter d'utiliser ce langage sur la totalité du projet. En effet, le C++ étant très répandu en entreprise, nous souhaitons nous y confronter. De plus, tant que nous utilisons des librairies standardisées, un portage sur GNU/Linux ou Mac OS reste possible. Par conséquent, nous utiliserons **Visual Studio** pour développer en C++, car il s'agit de l'environnement de développement intégré le plus puissant et le plus utilisé pour ce langage.

Enfin, pour les améliorations apportées à ADTool — qui est écrit en Java — nous emploierons **IntelliJ IDEA**.

6.4 Interfaces graphiques

Toujours dans un souci de compatibilité, notre choix s'est porté sur **Qt** pour l'interface graphique que présentera notre logiciel. En ce qui concerne ADTool, nous considérons que son interface est déjà suffisamment simple et intuitive. Nous ne prévoyons donc pas d'amélioration graphique pour ADTool, car ceci ne présenterait qu'un intérêt limité.

6.5 Outils de gestion de projet

Pour permettre la gestion de version de notre projet, nous avons choisi d'utiliser **Git** pour son efficacité et sa simplicité d'utilisation. De plus, il semble s'imposer progressivement dans le monde professionnel, au détriment de Subversion.

Pour le partage de fichiers lourds, la rédaction de notes de réunion, ou plus simplement pour mettre rapidement des idées en commun, nous avons décidé d'utiliser **Google Drive**, que nous préférons à Dropbox, car ce dernier ne permet pas d'écrire simultanément sur un même document.

Enfin, il nous est demandé d'employer **MS Project** comme outil de planification, que nous utiliserons donc pour tenir à jour le planning du projet.

7 Organisation du projet

7.1 Répartition des rôles

Étant donné que trois membres du groupe partent à l'étranger au second semestre, nous avons fixé une organisation précise pour que le projet se déroule de la meilleure manière possible. Les différents rôles ainsi décidés au sein du groupe sont décrits ci-dessous.

Coordinateur Le coordinateur s'occupe de planifier les réunions de projet et de les animer. Il est le contact privilégié des encadrants. Son rôle est aussi de s'assurer de l'avancée des rapports et de leur relecture. Il changera régulièrement afin que tout le groupe assure ce rôle. Hoel Kervadec est le premier coordinateur jusqu'à la livraison de ce rapport. Corentin Nicole sera le prochain coordinateur, puis Florent Mallard assurera ce rôle. De cette manière, chacun des étudiants partant en études à l'étranger aura tenu la place de coordinateur avant son départ.

Administrateur système L'administrateur système s'occupe de maintenir à jour les plates-formes et outils utilisés durant le projet (GitHub, GoogleDrive, etc.). Valentin Esmieu est en charge de ces plates-formes.

Scribe Un scribe est volontaire à chaque début de réunion afin de prendre des notes. Ce rôle est communément assuré par les personnes disposant de leur ordinateur à l'INSA afin de rédiger un compte-rendu en direct sur le Google Drive.

Responsable planification Le responsable planification est en charge du suivi et de la mise à jour de la planification du projet. Celui-ci utilisera le logiciel MS Project, comme indiqué dans la Section 6.

Décompte du temps Au sein du groupe, le décompte du temps passé sur le projet se fait de manière autonome. Nous utilisons pour cela un tableur créé sur Google Drive. Le coordinateur et le responsable planification consulteront ce tableur afin d'effectuer au mieux la répartition des tâches restantes.

Nous avons également décidé de nous réunir de manière hebdomadaire le mercredi soir. Au cours de ces réunions, nous nous concertons sur les tâches à réaliser, et les répartissons entre nous. C'est aussi à ce moment que nous débattons sur les principales questions qui seront posées au cours de la prochaine réunion avec nos encadrants.

7.2 Planification

En septembre, nous avons suivi des cours sur les ADTrees (donnés par Barbara Kordy) afin de se familiariser avec le concept, puis nous avons découvert ADTool et son fonctionnement. Nous avons ensuite jugé utile de suivre un cours de cryptographie afin de mieux saisir les protocoles de communication sécurisée. Gildas Avoine nous a donc dispensé un cours de deux heures, ce qui nous

a permis d'appréhender les concepts de protection des cartes Korrigo. Ceux-ci pourraient nous être utiles pour analyser et valuer les risques liés à ce type de carte.

Puis, en octobre, nous avons réalisé l'étude de l'existant sur le sujet de notre projet. Puis nous avons élaboré notre cahier des charges et défini l'architecture de notre logiciel. Enfin, nous nous sommes attelés à la rédaction de ce rapport de pré-étude. Comme nous avons noté le temps passé à travailler sur ce rapport, nous pouvons en déduire approximativement le temps qu'il nous faudra pour livrer les suivants. Pour rendre un rapport de vingt pages, nous avons passé, entre rédaction et correction, environ vingt-cinq heures chacun.

Pour la suite, bien que la planification ne soit pas encore totalement établie, nous pouvons déjà en donner les principaux axes.

En novembre, nous allons définir les spécifications fonctionnelles de notre logiciel afin de préciser davantage les différentes interactions entre les modules présents. Nous nous réunirons pour détailler l'interface graphique, puis nous nous mettrons d'accord sur la répartition des tâches et la manière d'implémenter les améliorations d'ADTool.

Ensuite, jusqu'aux vacances de Noël, nous établirons la planification complète du projet. Elle sera plus détaillée et contiendra un diagramme de Gantt expliquant la répartition des différentes tâches entreprises et leur ordonnancement. À ce moment, nous aurons donc tous les outils nécessaires à la modélisation de notre projet. Au terme de cette échéance, nous serons en mesure de présenter notre planification ainsi que nos spécifications lors des soutenances des 18 et 19 décembre.

De début 2015 à mi-février, nous définirons l'architecture interne de notre logiciel, et nous pourrons en parallèle commencer l'implémentation. Suivra enfin le développement complet de notre logiciel, jusqu'à la fin de l'année scolaire. Enfin, nous présenterons le projet complet durant les soutenances des 28 et 29 mai.

8 Risques

Afin d'améliorer la gestion de notre projet, et de ne pas nous faire surprendre par des évènements imprévus, nous avons décidé de les recenser. Cette liste donne les principaux problèmes potentiels ainsi que les éléments de résolution envisagés.

8.1 Risques humains

Le risque pouvant impacter le plus notre projet est de ne pas respecter le cahier des charges. Le facteur principal est humain : en effet, trois membres du groupe partent étudier à l'étranger dans le cadre de la mobilité internationale. Il n'en restera donc que trois à travailler sur l'implémentation. De ce fait, un investissement moindre dans le projet de l'un des membres restants conduirait à un retard d'autant plus important au second semestre. Planifier une réunion hebdomadaire et compter nos heures devrait permettre d'éviter ce problème. Il se peut également que nous ayons légèrement sur-estimé nos compétences, notre capacité à nous former, et annoncé une tâche trop difficile à exécuter. L'application pourrait nécessiter plus de temps que nous ne l'avions initialement prévu, et donc que nous ne puissions pas rendre un projet respectant le cahier des charges initial.

8.2 Risques techniques

Il existe aussi des risques plus facilement identifiables. Nous pouvons par exemple nous rendre compte que nos choix d'implémentation, annoncés dans la Section 5, ne fonctionnent pas comme nous l'avions prévu. Il est aussi possible que nous rencontrions des difficultés à déployer l'application, la plupart d'entre nous n'en ayant jamais développé. Nous pourrions ainsi connaître des problèmes dans la communication des différents modules. Nous nous serons renseignés sur le sujet et aurons envisagé plusieurs solutions, mais si celle retenue ne donne pas entière satisfaction, nous serons obligé d'en changer. Si ceci intervient à un stade avancé du projet, il y aura un retard dû à l'adaptation de l'implémentation. Ceci peut aboutir à une impossibilité de livrer le logiciel dans l'état dans lequel nous le décrivons ici.

La perte des données est un risque majeur en informatique, et peut potentiellement faire perdre l'intégralité de plusieurs mois de travail. Toutefois, l'utilisation de Git et le fait que nous ayons chacun une copie (à jour) du dépôt sur nos ordinateurs supprime quasiment ce risque.

8.3 Critiques

Dès cette première livraison, nous pouvons identifier des pistes d'amélioration dans notre organisation interne, en particulier dans le cadre de la rédaction de ce rapport. En effet, sitôt le plan du rapport établi, nous avons réparti le travail de rédaction entre nous sans nous concerter de manière précise sur le contenu. L'homogénéité du rapport a souffert de ce cloisonnement. Nous avons donc perdu du temps pour résoudre ce problème. À l'avenir, nous pourrons apprendre de cette erreur due principalement à notre manque d'expérience dans ce domaine. En effet notre groupe de travail est plus important que d'habitude et les rapports plus conséquents.

INSA Rennes
20 Avenue des Buttes de Coësmes
CS 70839
35708 Rennes Cedex 7

Tél. +33 (0) 2 23 23 82 00
Fax +33 (0) 2 23 23 83 96

www.insa-rennes.fr

INSA

UNIVERSITÉ EUROPÉENNE DE BRETAGNE

Commission des Titres d'Ingénieur

RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE