INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Encadrants

Gildas AVOINE

Barbara KORDY

Glasir

Application au réseau STAR

Rapport final de projet informatique

Étudiants

Pierre-Marie AIRIAU

Valentin ESMIEU

Maud LERAY

Résumé

Ce rapport revient sur notre projet portant sur l'analyse de la sécurité des systèmes. Dans le cadre de ce projet, nous avons développé Glasir, un logiciel d'analyse d'arbres d'attaque et de défense. Pour ce faire, nous avons également amélioré ADTool, un outil d'édition d'arbres, afin de l'intégrer à notre logiciel.

Ce rapport rend compte de la conduite du projet qui a été mené, au travers d'une série de documents portant entre autres sur les objectifs de Glasir, son implémentation et son manuel.

26 mai 2015

Table des matières

1 Remerciements

Ce projet terminé, nous souhaitons adresser nos remerciements à certaines personnes qui nous ont aidés et soutenus :

Nous remercions en premier lieu nos camarades Corentin Nicole, Hoel Kervadec et Florent Mallard, qui ont partagé avec nous les phases de pré-étude, de spécifications et de planification de ce projet. Ils ont contribué à sa mise en place et sans nul doute à son aboutissement.

Nous souhaitons ensuite remercier particulièrement nos encadrants, Mme Barbara Kordy et M. Gildas Avoine, qui se sont toujours montrés disponibles, impliqués et d'une aide précieuse. Un grand merci à eux pour leurs conseils avisés.

Nous remercions aussi vivement M. Piotr Kordy, le développeur d'ADTool, qui a su se montrer disponible pour répondre à nos interrogations concernant le logiciel.

Nous tenons aussi à remercier les élèves de 5^{e} année du département Informatique, qui nous ont fourni un ensemble d'ADTrees assez conséquents réalisés lors d'un TP. Grâce à eux, nous avons pu tester notre logiciel sur des ADTrees assez représentatifs d'une situation d'analyse de sécurité réelle.

Enfin, nous adressons nos remerciements à M. Jean-Louis Pazat, pour le temps qu'il a consacré à la lecture de nos rapports et au suivi de notre projet.

2 Résumé du projet

2.1 Contexte

À notre époque, assurer la sécurité des systèmes (informatiques ou non) est un enjeu majeur. De nombreuses méthodes de modélisation et d'analyse des systèmes ont donc été développées, dans le but d'identifier les risques et de les quantifier. C'est ainsi que le concept d'ADTrees [1] [7], un formalisme permettant de représenter sous forme d'arbre les étapes d'une attaque précise contre un système, a vu le jour. L'un des logiciels modélisant ces ADTrees s'appelle ADTool [8].

ADTool implémente la plupart des fonctionnalités d'édition d'ADTrees, y compris l'utilisation de paramètres tels que le coût d'une attaque. Mais il a été conçu comme un logiciel de modélisation avant tout, et comporte donc peu d'outils permettant d'analyser les ADTrees.

2.2 Objectifs

L'objectif de ce projet a donc été la réalisation d'un logiciel fournissant à l'utilisateur, un expert en sécurité, des outils d'analyse des ADTrees. Ce logiciel a été nommé Glasir en référence à l'arbre aux feuilles d'or de la mythologie nordique [10], et dont le logo est visible sur la FIGURE 1. Il dispose de trois fonctionnalités principales, qui ont été détaillées dans les rapports précédents :

- l'**Éditeur de fonctions**, qui permet à l'expert de créer de nouveaux paramètres fonctions de ceux déjà présents dans l'arbre ;
- le **Filtre**, qui aide l'utilisateur à ôter de l'ADTree les branches dont les valuations ne respectent pas un certain critère ;
- l'**Optimiseur**, qui sélectionne le(s) meilleur(s) chemin(s) de l'ADTree selon un paramètre donné.

ADTool a quant à lui été amélioré, et est utilisé au sein de Glasir comme viewer d'ADTrees.

2.3 Bilan

Logiciel concerné	Cahier des charges	Implémentation
Glasir	Éditeur de fonctions	✓
	Filtre	✓
	Optimiseur	✓
	Interface graphique	✓
	Gestion des fichiers projets	✓
	Intégration ADTool dans Glasir	✗
	Affichage d'instances multiples d'ADTool	✓
	Bibliothèque de modèles	✓
ADTool	Amélioration de la représentation textuelle	✓
	Affichage multi-paramètres	✗
	Copier/couper/coller	✓
	Annulation de l'action précédente	✓

TABLE 1 – Retour sur le cahier des charges de Glasir.

1. Abréviation d'« Attack-Defense Trees », ou « Arbres d'Attaque et de Défense » en français.

Il est d'ores et déjà possible de dresser un premier bilan du projet, sous la forme d'une rétrospective sur les éléments définies dans le cahier des charges. Ce premier bilan est présenté sous la forme de la TABLE 1.

La 1re colonne de la TABLE 1 rappelle le cahier des charges tel qu'il a été défini dans le rapport de planification [1], c'est-à-dire sous forme de tâches unitaires. La 2e colonne de la TABLE 1 indique si la tâche correspondante a bien été effectuée (symbole ✓) ou non (symbole ✗). Comme la TABLE 1 permet de le voir, la plupart des fonctionnalités promises dans le cahier des charges du projet ont effectivement été livrées. Il est à noter que si les fonctionnalités implémentées sont fonctionnelles et rendent bien le service décrit initialement dans le cahier des charges, leur implémentation a parfois différé de leur conception pour des raisons pratiques. Ces différences seront détaillées et justifiées par la suite dans la SECTION 3.

Quatre rapports ont été rédigés dans le cadre de ce projet : celui de pré-étude [3], celui de spécifications fonctionnelles [2], celui de planification [1] et celui de conception [5]. Ils ont permis de mieux définir les objectifs du logiciel, ainsi que l'organisation de son implémentation. Le but de ce rapport final est de rendre compte de la tenue des objectifs qui ont été fixés au début du projet, et des méthodes mises en place pour y parvenir. Il évoquera aussi les ajustements qui ont été mis en place afin de gérer les imprévus auquels nous avons fait face.

Pour cela, ce rapport est composé d'une collection de documents indépendants couvrant l'ensemble de ces aspects : rectificatifs de conception, compte-rendus de tests, description des livrables, manuel des logiciels, et pour finir une rétrospective sur la conduite du projet.

FIGURE 1 – Logo du logiciel Glasir.

3 Rectificatifs du rapport de spécifications fonctionnelles

Cette section a pour but d'énoncer et de justifier les différences constatées entre les fonctionnalités prévues dans le rapport de spécifications fonctionnelles [2] et leur implémentation lors de la phase de développement. La SOUS-SECTION 3.1 commence d'abord par énoncer les modifications les plus générales concernant l'architecture du logiciel, avant d'entrer plus en profondeur dans le code, dans les sous-sections suivantes.

3.1 Architecture générale

La FIGURE 2 illustre l'architecture générale imaginée lors du rapport de spécifications fonctionnelles [2]. Suite à des contraintes techniques, notamment dûes au fait qu'un programme Java (ADTool dans notre cas) est difficilement contrôlable depuis un programme C#, l'architecture a été revue pour correspondre à celle présentée à la FIGURE 3. C'est pourquoi ADTool ne fait finalement pas partie intégrante de Glasir, mais est lancé en tant que simple « viewer » d'ADTrees. En effet, il est impossible, depuis Glasir, de détecter des modifications faites dans ADTool, comme par exemple la modification d'une valuation de l'une des feuilles de l'ADTree. Par conséquent, la décision fut prise de désactiver toutes les fonctions d'édition dans ADTool lorsque Glasir se lance, donnant ainsi à ADTool un mode spécial baptisé « viewmode ». Dans ce mode, l'utilisateur ne peut que visualiser un ADTree, sans pouvoir le modifier (les menus et les raccourcis clavier ont été supprimés). De plus, la bibliothèque de modèles n'est plus incluse dans Glasir : cette décision est justifiée dans la SOUS-SECTION 4.4.

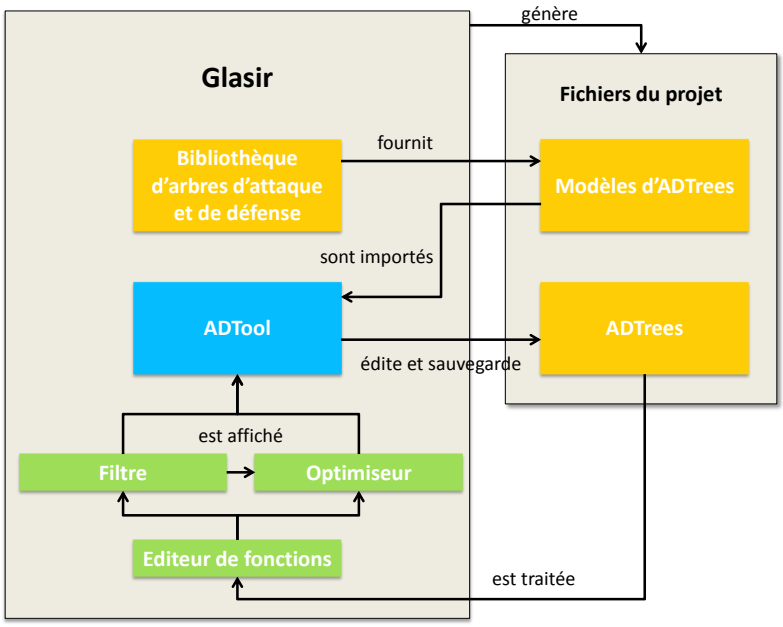

FIGURE 2 – Architecture initialement prévue.

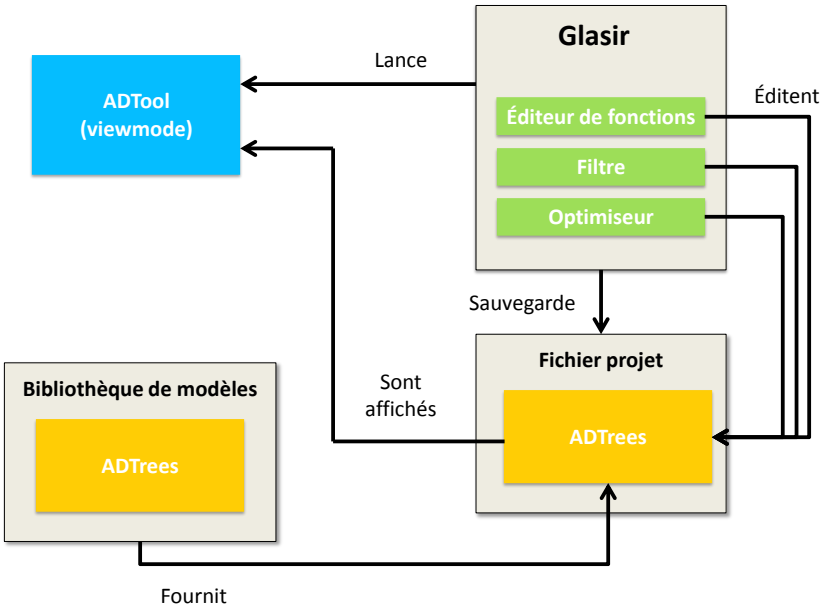

FIGURE 3 – Architecture réelle.

Les modules principaux de Glasir que sont l'Éditeur de fonctions, le Filtre et l'Optimiseur ont eux aussi été un peu revus par rapport à l'implémentation prévue dans le rapport de conception [5]. Ces changements sont principalement liés à des questions de cohérence soulevées après le début du développement. Les SOUS-SECTIONS 3.2, 3.3 et 3.4 détaillent les modifications apportées à ces modules.

3.2 Algorithme de l'Éditeur de fonctions

Bien que nous n'ayons pas jugé nécessaire de détailler l'algorithme de l'Éditeur de fonctions lors de la rédaction du rapport de spécifications fonctionnelles [2], celui-ci s'est révélé plus délicat à implémenter que prévu. Son nouveau fonctionnement est donc détaillé ci-après, par une présentation de l'ALGORITHME 1 qui a été effectivement implémenté.

Algorithme 1 functionEdition(racine, param1, param2, function)

```
if type(racine) == defense then
  for all sub in fils(racine) do
    functionEdition(sub, param1, param2, function)
  end for
  return
end if

if type(racine) == feuille then
  p1 = param1(racine)
  p2 = param2(racine)
  newParam(racine) = function(p1,p2)
  return
else
  for all sub in fils(racine) do
    if (type(sub) == defense then
      p1 = param1(racine)
      p2 = param2(racine)
      newParam(racine) = function(p1,p2)
    end if
    functionEdition(sub, param1, param2, function)
  end for
end if
```

Les notations non explicites de l'ALGORITHME 1 sont présentées ci-dessous :

- `racine` correspond au nœud racine de l'arbre ou sous-arbre manipulé ;
- `param1` est une fonction renvoyant la valeur du 1er paramètre qui nous intéresse pour un nœud donné ;
- `param2` est une fonction renvoyant la valeur du 2^{e} paramètre qui nous intéresse pour un nœud donné ;
- `function` est une fonction calculant un résultat à partir de deux paramètres ;
- `fils` est une fonction renvoyant la liste des fils du nœud passé en paramètre ;
- `mode` précise si il s'agit d'un nœud disjonctif ou conjonctif ;
- `type` précise si il s'agit d'un nœud d'attaque ou de défense, ou bien d'une feuille.

L'algorithme de l'Éditeur de fonctions est donc un algorithme récursif, qui cherche les feuilles de l'ADTree pour ajouter le nouveau paramètre à celles-ci.

3.3 Algorithme du Filtre

L'algorithme original du Filtre, représenté par l'ALGORITHME 2, comportait quelques erreurs telles que la possibilité de conserver des sous-nœuds d'un nœud conjonctif qui ne respectaient pas la condition de filtrage. Cet algorithme a donc été corrigé en conséquence pour devenir l'ALGORITHME 3 qui a été implémenté.

Algorithme 2 filtre(racine, rules)

```
for r in rules do
  if not r(racine) then
    delete(r) // will delete subtrees as well
    return
  end if
end for

for n in fils(racine) do
  filtre(n, rules)
end for
```

Algorithme 3 filtre(racine, param, valLim)

```
val = param(racine)

if pire(val, valLim) then
  delete(racine) // will delete subtrees as well
  return

else
  if mode(racine) == et then
    for all sub in fils(racine) do
      subval = sub.param
      filtre(sub, param, valLim+subval-val)
    end for
  else
    for all sub in fils(racine) do
      filtre(sub, param, valLim)
    end for
  end if
end if
```

Dans l'ALGORITHME 3, `valLim` est la valeur limite (maximale ou minimale selon les cas) acceptée pour le paramètre `param` avec lequel on souhaite filtrer l'ADTree. Et `pire` est une fonction renvoyant *vrai* si son 1er paramètre `val` est moins bon (c'est-à-dire inférieur ou supérieur, selon les cas) que son 2e paramètre `valLim` pour le domaine de valuation considéré. Elle renvoie *faux* dans le cas contraire.

3.4 Algorithme de l'Optimiseur

L'algorithme initial de l'Optimiseur, tel que décrit dans le rapport de spécifications fonctionnelles [2], est rappelé par l'ALGORITHME 4.

Algorithme 4 opti(racine, param)

```
l_fils = fils(racine)
if vide(l_fils) then
  return
end if

if mode(racine) == ou then
  v = param(racine)
  for n in l_fils do
    if not defense(n) and param(n) != v then
      delete(n) // will delete subtrees as well
    end if
  end for
end if

for n in fils(racine) do
  opti(n, param)
end for
```

Après avoir implémenté le Filtre, il est apparu qu'il était possible de l'utiliser pour simplifier l'implémentation de l'Optimiseur. Le nouvel algorithme obtenu (l'ALGORITHME 5) n'impactant pas le résultat, c'est finalement lui qui a été choisi pour coder l'Optimiseur.

Algorithme 5 opti(racine, param)

```
v = param(racine)
filtre(racine, param, v)
```

La présente section a permis de décrire les différences constatées à la fin du projet entre les promesses du rapport de spécifications fonctionnelles [2] et l'implémentation réelle. Un autre rapport avait été rédigé avant le début du développement, celui de conception [5]. Là encore, des écarts sont constatés avec ce qui a été codé, c'est ce dont fait état la SECTION 4.

4 Rectificatifs du rapport de conception

Cette section a pour objectif de comparer les fonctionnalités décrites lors du rapport de conception [5] à celles qui ont été ensuite implémentées.

4.1 Affichage multiple des paramètres

Cette nouveauté n'a finalement pas été mise en place. En effet, l'étude approfondie du code source d'ADTool a montré qu'un tel changement aurait nécessité une refonte quasi-totale de la structure du logiciel. De telles modifications auraient demandé un temps de travail bien trop conséquent comparé au temps disponible pour la réalisation du projet. De toute manière, l'intérêt de ce nouvel affichage avait finalement été remis en cause : le risque de perdre en clarté et en lisibilité lors de la manipulation des ADTrees avait semblé trop important.

4.2 Couper/copier/coller

La FIGURE 4 rappelle la fonctionnalité de couper/copier/coller telle qu'imaginée lors du rapport de conception [5]. Après avoir mieux pris connaissance de la structure interne d'ADTool, il a paru beaucoup plus simple de réaliser cette fonctionnalité comme illustré sur la FIGURE 5.

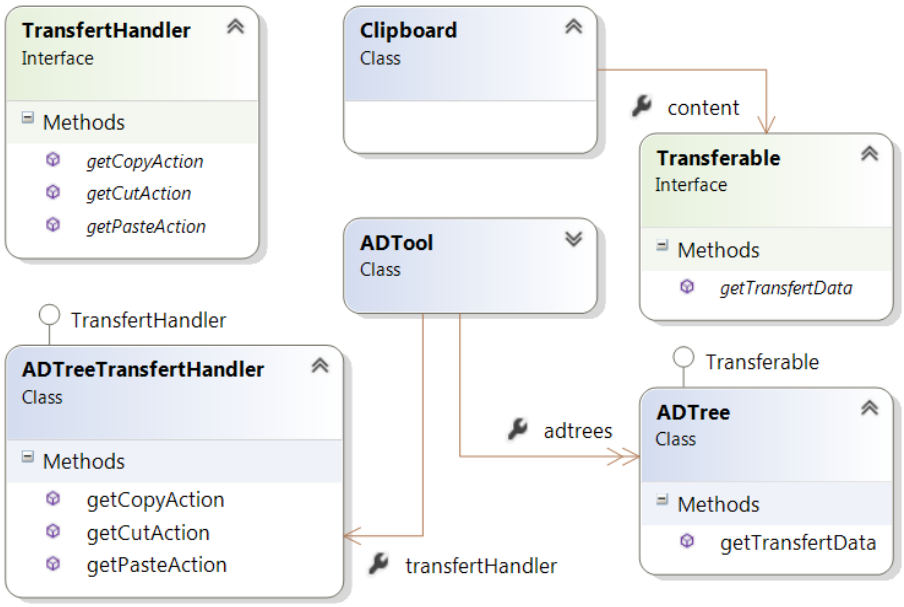

FIGURE 4 – Diagramme de classes prévu pour le couper/copier/coller.

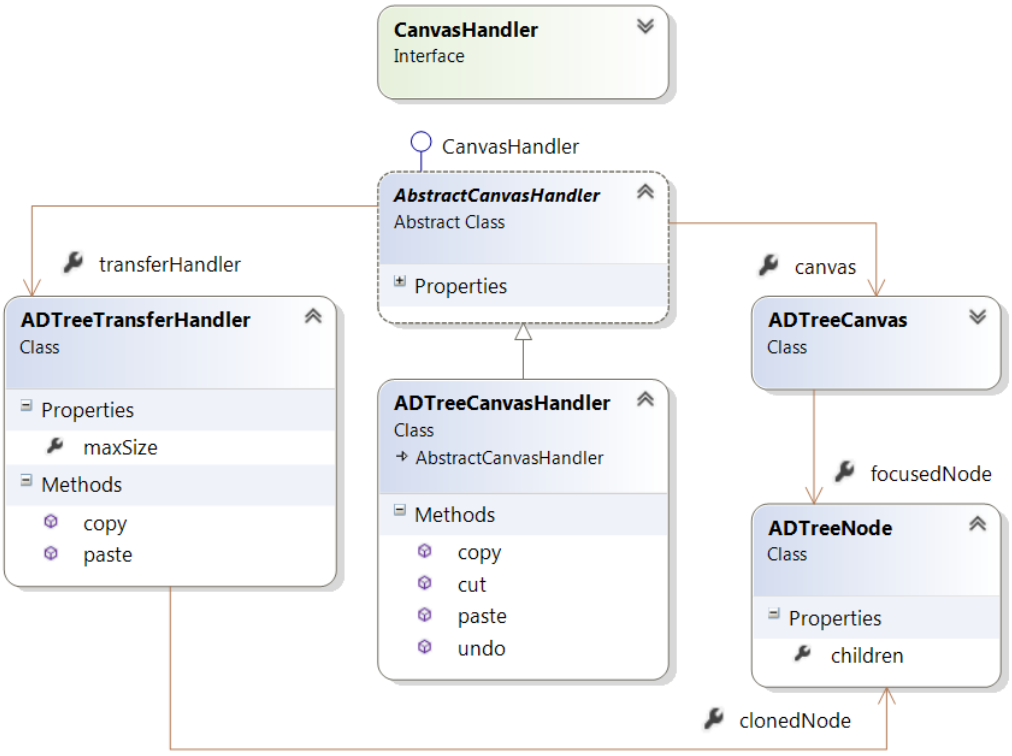

FIGURE 5 – Diagramme de classes réellement implémenté pour le couper/copier/coller.

Dans ADTool, un ADTree est en réalité un *ADTreeNode* possédant des fils, qui sont eux-mêmes des *ADTreeNode*. L'ADTree courant est affiché à l'écran via un *ADTreeCanvas*. Les actions au clavier et à la souris réalisées par l'utilisateur sont captées par la classe *AbstractCanvasHandler*, qui possède le *ADTreeCanvas* sur lequel agit l'utilisateur. Il a donc été ajouté à l'*AbstractCanvasHandler* un attribut de *ADTreeTransferHandler*, une nouvelle classe qui contient les méthodes de copie et de collage d'un *ADTreeNode*. Ainsi, lorsque l'utilisateur lance la commande « couper », par exemple, cette dernière est captée par la classe *CanvasHandler* qui demande alors à sa classe parente (*AbstractCanvasHandler*) de récupérer l'*ADTreeNode* à couper (il s'agit de l'attribut *focusedNode* de *ADTreeCanvas*). L'*ADTreeTransferHandler* se charge ensuite de l'opération de copie en stockant dans *clonedNode* un clone de l'*ADTreeNode* coupé. Dans le cas de l'action « couper », la suppression de l'ADTree coupé est gérée en interne par l'*AbstractCanvasHandler*. Pour le collage, les opérations sont sensiblement identiques, sauf que le clone stocké par l'*ADTreeTransferHandler* est restitué à l'*AbstractCanvasHandler* qui se charge de l'ajouter à son *ADTreeCanvas*.

4.3 Annulation des dernières actions

La FIGURE 6 illustre l'implémentation de l'annulation de la dernière action telle qu'imaginée lors du rapport de conception [5], dans lequel il n'était pas prévu de pouvoir effectuer plusieurs annulations à la suite. De la même façon que pour le couper/copier/coller, la structure d'ADTool a conduit à implémenter l'annulation différemment, comme indiqué sur la FIGURE 7.

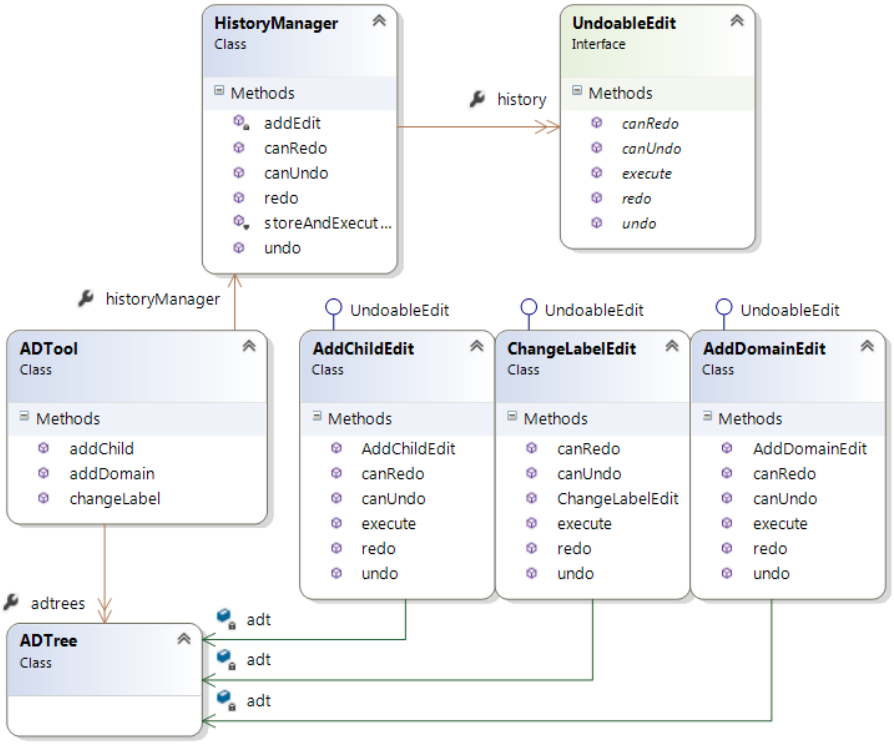

FIGURE 6 – Diagramme de classes prévu pour l'annulation des dernières actions.

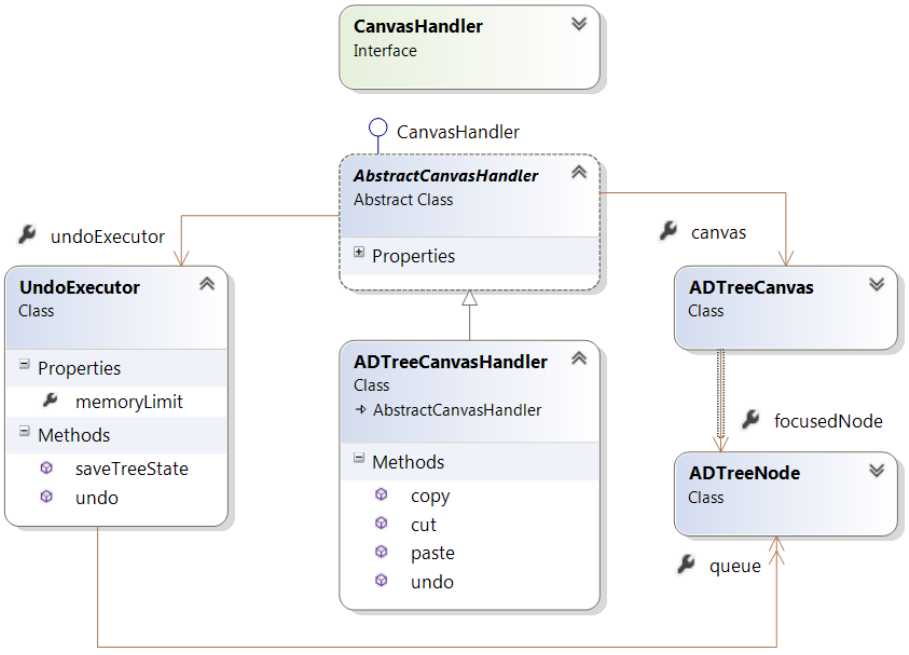

FIGURE 7 – Diagramme de classes réellement implémenté pour l'annulation des dernières actions.

En effet, associer chaque action à une classe se révélait trop fastidieux, étant donné le grand nombre d'actions possibles. L'*AbstractCanvasHandler* se voit donc doté d'un attribut de *UndoExecutor*, une nouvelle classe qui contient les méthodes d'annulation. Lorsque l'utilisateur réalise une action, un clone de l'ADTree courant est stocké dans la pile de l'*UndoExecutor*. Les clones sont dépilés et rechargés dans l'*ADTreeCanvas* lorsque l'utilisateur utilise la fonctionnalité d'annulation. Afin d'éviter une saturation de la mémoire, la pile ne peut contenir qu'un certain nombre de clones de l'ADTree, supprimant si besoin le plus ancien pour pouvoir ajouter le plus récent. Il est à noter que plus l'ADTree est petit, plus le nombre d'actions annulables est grand, car la pile peut contenir plus de clones avant d'arriver à saturation. Dans le pire des cas, si l'ADTree est conséquent (dans notre cas, s'il contient plus de 1000 nœuds), seule la dernière action sera annulable.

4.4 Bibliothèque de modèles

La FIGURE 8 illustre l'implémentation de la bibliothèque de modèles telle qu'imaginée lors du rapport de conception [5].

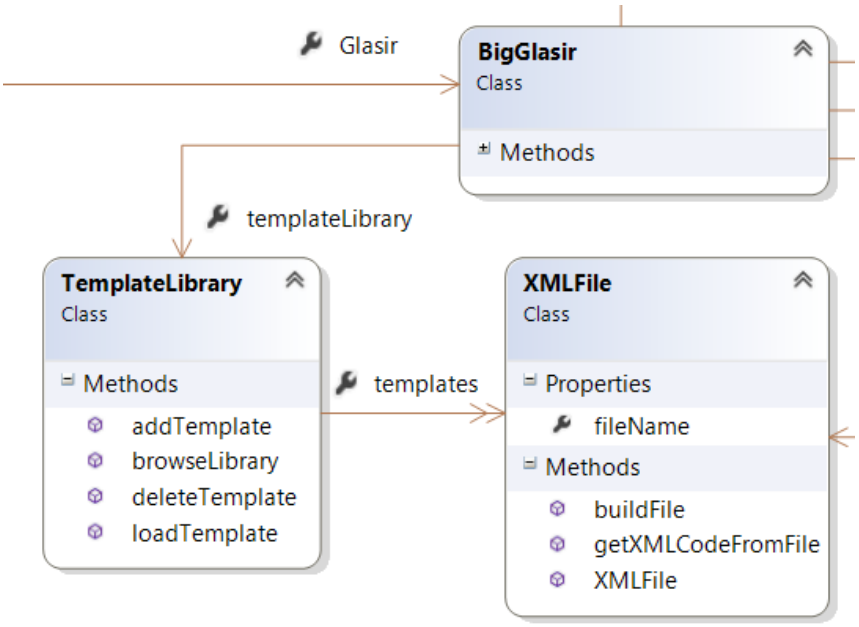

FIGURE 8 – Diagramme de classes prévu pour la bibliothèque de modèles.

Finalement, cette bibliothèque se présente comme un simple dossier (fourni avec Glasir) contenant les modèles d'ADTrees. Ceux-ci portent sur le thème des transports en commun, ce qui correspond à l'étude de cas dans le cadre de ce projet mais ne doit pas restreindre l'utilisation de Glasir. C'est pourquoi il a été décidé de la fournir séparée du logiciel, pour que ce dernier reste adapté à l'analyse en sécurité de n'importe quel système.

5 Méthodologie de test et résultats

Tout au long de l'implémentation de Glasir et des nouvelles fonctionnalités d'ADTool, il a fallu s'assurer du bon fonctionnement du logiciel. Pour cela, nous avons mis en place une méthodologie de test en accord avec les contraintes du projet. La contrainte la plus contraignante était que, Glasir et ADTool étant deux logiciels distincts, nous ne pouvions pas assurer au travers de tests unitaires dans Glasir que les fonctionnalités d'analyse fonctionnaient correctement. En effet, c'était l'affichage correct ou non de l'arbre résultat sous ADTool qui permettait de vérifier l'exactitude de l'analyse. Nous avons donc dû recourir à une méthode différente de celle des tests unitaires.

Le développement de Glasir et l'amélioration d'ADTool ayant été versionnés, nous avions un cadre propice pour le test des fonctionnalités à chaque rendu de version. La méthodologie employée pour tester Glasir a donc été de le faire tester, ainsi que la nouvelle version d'ADTool, par des personnes avec trois profils distincts et complémentaires, que nous allons présenter dans la SOUS-SECTION 5.1. De plus, nous avons pu tester chaque nouveauté sur des ADTrees relativement conséquents, fournis par les élèves de 5INFO à l'issue de l'un de leurs TP. Ces ADTrees étaient intéressants car ils possédaient des nœuds des deux types (attaque et défense), et ils étaient dotés de plusieurs valuations.

5.1 Les profils de testeurs

Les tests de Glasir ont été réalisés de manière plus ou moins continue par trois catégories de personnes avec des profils différents, qui vont être présentés ci-dessous.

Les développeurs Cette catégorie comprend les trois étudiants en charge de l'implémentation du projet : Pierre-Marie AIRIAU, Valentin ESMIEU ainsi que Maud LERAY. Pour chacune des fonctionnalités des logiciels, celui qui avait la charge de l'implémenter la testait au fur et à mesure du développement ; puis une fois l'implémentation terminée, la faisait tester par les deux autres du groupe pour localiser d'éventuels « bugs » restants. La méthode utilisée pour ces tests se décomposait en deux parties : tout d'abord, essayer de pousser les fonctionnalités dans leurs retranchements pour tenter d'aboutir à des résultats incohérents, puis tester le comportement du logiciel lors d'utilisations inappropriées pour localiser les cas d'utilisations non maîtrisés par le code.

L'expert en ADTrees Chacune des versions rendues a ensuite été testée par un expert en ADTrees maîtrisant également la version initiale d'ADTool. Son rôle était de s'assurer que les résultats des fonctionnalités d'analyse étaient cohérents avec le formalisme des ADTrees, dont les règles sont très précisément définies.

L'expert en sécurité Il s'agit enfin d'un expert en sécurité qui connaissait les objectifs du projet, mais qui n'était pas familier avec le formalisme des ADTrees ni avec ADTool. Son rôle était de tester le logiciel avec un oeil « extérieur » afin de soulever des soucis d'ordre ergonomique et/ou pratique qui n'auraient probablement pas été remarqués sinon.

5.2 Rapport de tests

Sont maintenant présentés les différents retours de tests que nous avons eus sur Glasir et sur ADTool après les rendus des différentes versions :

- v0.1 avec l'interface de Glasir, l'Éditeur de fonctions et la refonte de la grammaire d'ADTool ;
- v0.2 avec le Filtre dans Glasir et la fonctionnalité de couper/copier/coller dans ADTool ;
- v1.0 avec l'Optimiseur pour Glasir et l'annulation d'une action pour ADTool.

Sont détaillés ici les retours consécutifs aux rendus des différentes versions de Glasir. Ces retours sont répartis selon la nature des éléments qu'ils contiennent.

5.2.1 Retours d'ordre général sur Glasir

Le principal problème soulevé dès la version 0.1 fut que l'ouverture d'ADTrees depuis Glasir n'était pas toujours possible, à cause d'un manque de précision dans le code lors du lancement d'ADTool (il n'était pas précisé d'ouvrir le fichier .jar avec Java). Ceci était dû au fait que le logiciel avait été testé en amont sur les ordinateurs personnels des développeurs, mais jamais sur une machine vierge où l'association entre .jar et Java n'était pas paramétrée par défaut. Ceci a été corrigé par la suite.

Esthétiquement parlant, la fenêtre principale de Glasir a été critiquée pour son absence de redimensionnement lors de l'étirement de la fenêtre. La taille de la fenêtre a donc été fixée pour empêcher son étirement.

Enfin, Glasir disposant d'une arborescence des ADTrees pour le projet en cours, il a été demandé d'y mettre plus en évidence celui en cours de manipulation. Le nom de l'ADTree courant apparaît donc dorénavant au-dessus des autres, dans une police plus grande, afin d'être mieux visible.

5.2.2 Retours sur les fonctionnalités de Glasir

Au rendu de la version 0.1, l'Éditeur de fonctions comportait de nombreux bugs. Pour commencer, il ne savait pas traiter les paramètres de type booléen, ni la présence de défenses dans un ADTree. Quelques questions ont également été soulevées concernant la gestion des paramètres discrets, tels que la difficulté pour l'attaquant évaluée sous la forme de valeurs L (Low), M (Medium) ou H (High). Enfin, il a été remarqué que plusieurs paramètres créés pouvaient porter le même nom, ce qui posait par la suite des soucis de clarté pour l'analyse des ADTrees. Des décisions ont été prises pour remédier à ces problèmes, qui ont été corrigés depuis.

Un plus grand soin a été ensuite apporté aux tests des fonctionnalités restantes de Glasir, pour fournir aux versions suivantes des implémentations plus abouties du Filtre et de l'Optimiseur.

En conséquence, le rendu de la version 0.2 contenant le Filtre présentait moins de dysfonctionnements. La seule amélioration demandée était d'afficher une fenêtre pop-up du type « Arbre vide après filtrage » plutôt que l'ADTree réduit à « Root » lorsque le filtre appliqué ne permettait de conserver aucun nœud. Cela est en effet plus compréhensible pour l'utilisateur et a donc été implémenté suite à cette remarque.

L'Optimiseur, quant à lui rendu en version finale 1.0, n'a montré aucun bug lors de la phase de test.

5.2.3 Retours sur les nouveautés d'ADTool

Le constat a été fait que la possibilité de modifier les ADTrees via ADTool lors de leur ouverture avec Glasir était problématique, du fait de l'indépendance des deux logiciels une fois lancés. En effet, les changements effectués dans l'un n'étaient pas connus par l'autre et vice-versa. La solution mise en place fut celle d'un « view mode », utilisé pour bloquer toutes les possibilités de modification des ADTrees dans les instances d'ADTool lancé depuis Glasir. Ce nouveau mode fut présent dès la version 0.2.

La seule critique faite à l'égard du couper/copier/coller était que le menu déroulant comportait toujours la possibilité de coller (sans aucun effet) un nœud même si aucun n'avait été préalablement copié ni coupé. Bien que sans conséquences notables, cette petite incohérence fut vite corrigée.

Il a également été demandé après les tests de la version 0.2 d'ajouter en haut de la fenêtre d'ADTool le nom (et même le path complet) de l'ADTree affiché, ce qui a été fait depuis. C'est en effet très pratique lorsque plusieurs instances d'ADTool sont ouvertes simultanément, comme c'est souvent le cas lors de l'utilisation de Glasir.

6 État de Glasir à la conclusion du projet

Cette section présente l'état dans lequel Glasir a été livré à l'issue du projet. On présente ici un récapitulatif des objectifs initiaux et leur état d'achèvement dans la SOUS-SECTION 6.1, avant d'évoquer en SOUS-SECTION 6.2 des améliorations à envisager.

6.1 Tenue des objectifs

Cette sous-section récapitule en premier lieu les accomplissements relatifs à Glasir, avant de mettre en évidence ceux concernant ADTool.

6.1.1 Glasir

L'objectif du projet était de fournir à des experts en sécurité, familiers avec le formalisme des ADTrees, une solution logicielle pour analyser et exploiter ces derniers. L'utilisation de cette solution logicielle, Glasir, devait se faire au moyen d'un ensemble de « modules » constituant les fonctionnalités d'analyse des ADTrees, au sein d'une interface graphique simple et claire. L'interface imaginée pour répondre à ces besoins est celle de la FIGURE 9.

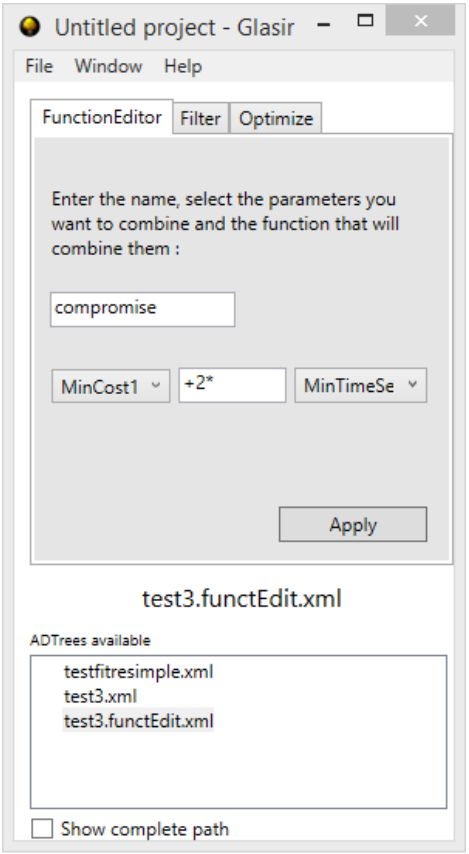

FIGURE 9 – Interface du logiciel Glasir.

L'interface graphique de Glasir a été implémentée avec les éléments qui avaient été promis :
- les trois fonctionnalités principales (Éditeur de fonctions, Filtre et Optimiseur) bien en évidence ;
- la liste des ADTrees contenus dans le projet en cours, sous forme d'arborescence ;
- un menu intuitif pour l'utilisateur.

Le seul écart à noter est qu'ADTool n'est finalement pas complètement intégré à Glasir. Les deux logiciels, bien que liés, restent indépendants dans le sens où bien que Glasir ait besoin d'ADTool pour fonctionner, il ne communique pas avec ce dernier. Les éventuels soucis de cohérence des ADTrees après modification ont été contournés grâce à la mise en place d'un « view mode » pour ADTool, comme expliqué en détails dans la SECTION 3. L'expérience utilisateur n'est pas affectée par cette décision, qui permet en plus de mieux distinguer l'édition des ADTrees de leur analyse. Enfin, il est joint à Glasir un dossier faisant office de bibliothèque de modèles et contenant un certain nombre d'ADTrees pouvant être utilisés pour des tests, mais aussi dans des projets. Ces ADTrees sont basés sur la thématique des transports en commun afin d'illustrer notre étude de cas sur le STAR (Service des Transports en commun de l'Agglomération Rennaise).

L'état des trois fonctionnalités principales de Glasir, qui confèrent au logiciel son intérêt pour l'analyse, est détaillé dans les paragraphes suivants.

L'Éditeur de Fonctions Ce dernier permet comme promis la combinaison de plusieurs paramètres, mais il est seulement possible d'en combiner deux en une seule fois. Cependant, il est ensuite possible de réutiliser les paramètres créés pour en produire de nouveaux, ce qui permet indirectement de combiner plus de deux paramètres. Les fonctions mathématiques utilisables sont comme annoncé les fonctions de base (somme, différence, produit, division), associées à des coefficients éventuels. Seules les fonctions min et max ne sont finalement pas applicables, car la façon d'écrire la fonction ne s'y prêtait pas.

Le Filtre Cette fonctionnalité permet comme prévu de filtrer un ADTree selon un paramètre sélectionné. L'ensemble des valeurs acceptées par le Filtre est à préciser en fournissant une valeur limite (maximale ou minimale selon les cas). Le Filtre conservera alors les chemins de l'arbre étant meilleurs ou au moins aussi efficaces selon le paramètre choisi que la valeur limite. Cela dit, il est impossible de filtrer en une seule fois un ADTree selon plusieurs paramètres, malgré ce qui avait été annoncé dans le rapport de spécifications fonctionnelles [2]. Ceci peut malgré tout être fait en appliquant le Filtre consécutivement sur différents paramètres et sur différentes valeurs limites, jusqu'à obtenir le filtrage définitif souhaité.

L'Optimiseur Comme annoncé, l'Optimiseur permet de conserver simplement le(s) chemin(s) le(s) plus efficace(s) selon le paramètre choisi par l'utilisateur.

Ainsi, malgré quelques changements comparé à ce qui avait été annoncé lors de la phase de conception, Glasir est bien capable de remplir le rôle qui lui avait été fixé : il fournit à l'expert en sécurité des outils d'analyse efficaces et simples d'utilisation.

6.1.2 ADTool

Le cahier des charges prévoyait, en plus de l'implémentation de Glasir, un certain nombre de modifications à apporter à ADTool. Parmi ces dernières, toutes ont été correctement mises en place,

à l'exception de l'affichage simultané de plusieurs paramètres sur les noeuds d'un même ADTree. Cette nouveauté a été abandonnée car elle nécessitait une charge de travail trop importante, comme déjà expliqué dans la SECTION 3.

Le logiciel permet donc toujours de créer, d'afficher et d'éditer des ADTrees de façon simple et intuitive, mais son ergonomie a été améliorée par l'ajout des fonctionnalités suivantes :
- la possibilité de couper/copier/coller des sous-arbres d'un ADTree ;
- l'annulation des actions précédentes.

De plus, la grammaire utilisée pour générer la représentation textuelle des ADTrees dans la fenêtre « ADTerm Edit » a été améliorée. Elle est à présent plus complète et plus lisible, car elle contient aussi les labels des nœuds intermédiaires (et non plus seulement les labels des feuilles).

Toutes les nouveautés évoquées ici apportent une réelle aide à l'expert en sécurité, qu'il veuille créer, éditer ou analyser des ADTrees. Mais le logiciel peut encore être amélioré, c'est pourquoi la SOUS-SECTION 6.2 présente une liste d'idées à creuser dans l'éventualité d'une continuation de l'implémentation.

6.2 Améliorations possibles

Suite à nos recherches sur l'analyse des ADTrees et à l'implémentation de Glasir, nous avons pu identifier des améliorations et de nouvelles fonctionnalités potentiellement utiles pour l'analyse des ADTrees. La liste d'améliorations proposées ci-après n'est évidemment pas exhaustive, elle présente juste quelques pistes qui pourraient être envisagées si le projet avait à être poursuivi.

Interconnectivité ADTool/Glasir Sans parler d'intégrer totalement ADTool dans Glasir, il semble envisageable de trouver un moyen d'informer l'un des modifications effectuées via l'autre, et vice-versa. Il serait plus agréable pour l'utilisateur de pouvoir éditer et analyser les ADTrees à partir d'une même fenêtre logicielle, sans avoir à changer d'outil sans arrêt. Cette interconnectivité permettrait de se passer de la version « view mode » d'ADTool actuellement utilisée dans Glasir.

Suggestion de défenses Un exemple d'un nouveau module d'analyse d'arbres qui nous a paru intéressant consisterait à pouvoir donner à Glasir un arbre représentant un système défendu, mais qui dans la réalité ne l'est pas. L'expert pourrait alors demander à Glasir, en lui ayant au préalable indiqué le coût de l'installation de chacune des défenses, de trouver la disposition idéale des défenses contre une attaque compte tenu d'un critère (par exemple le temps nécessaire à l'attaque, le prix de l'attaque), pour un budget donné.

Fonction de recherche dans ADTool Cette dernière consisterait à pouvoir chercher un mot, ou un expression entière, parmi l'ensemble des labels des nœuds présents dans l'ADTree courant. Cela pourrait par exemple s'effectuer à l'aide d'un raccourci clavier, tel que le classique CTRL+F. Cette fonctionnalité pourrait permettre de vrais gains de temps, surtout dans le cas où l'arbre est de taille conséquente.

7 Rétrospective

Ce rapport a jusqu'alors permis d'établir un récapitulatif des changements effectués comparés aux objectifs initiaux, ainsi qu'un état concret du projet à l'heure actuelle. La présente section présente les acquis issus de cette expérience, ainsi qu'un point de vue critique sur la gestion de projet mise en place.

7.1 Remarques générales

Ce projet de 4e année Informatique fut une expérience très enrichissante. Tout d'abord, il nous a permis de beaucoup progresser en matière de programmation, dans les technologies utilisées : principalement C#, Java et WPF dans notre cas. De plus, nous avons pu au cours de nos recherches nous familiariser tout au long de ce projet avec le formalisme des ADTrees et la problématique de la sécurité des systèmes en général, des domaines que la plupart d'entre nous connaissions peu avant le projet. Ensuite, nous avons été confrontés aux problématiques du travail en équipe : présence d'un coordinateur, répartition des tâches, utilisation des outils de gestion de versions (Git dans notre cas), etc. Enfin, nous avons pu gérer de A à Z un projet d'une assez grande envergure, ce qui a nécessité plusieurs phases de préparation avant de pouvoir démarrer : pré-étude [3], spécifications fonctionnelles [2], planification [1] puis finalement conception [5]. C'est sur l'avant-dernière de ces phases, celle de planification, que nous allons à présent revenir pour présenter un bilan de notre gestion de projet dans la SOUS-SECTION 7.2.

7.2 Réflexions sur la gestion de projet

Cette sous-section n'est qu'un rapide bilan concernant notre gestion de projet, puisqu'un rapport de bilan de planification [4] a été rédigé en parallèle du présent rapport et comporte plus de détails à ce sujet.

Les quelques écarts constatés à la SECTION 3 entre les fonctionnalités promises et celles développées s'expliquent par certains défauts que nous avons identifiés dans notre gestion de projet, et qui sont les suivants :

- une phase de pré-étude insuffisante sur certains points technique ;
- un cahier des charges trop vague ;
- une méthodologie de tests trop approximative.

Ces points à améliorer sont développés dans le rapport bilan de planification [4], c'est pourquoi nous ne nous attardons pas dessus ici. En revanche, ils donnent déjà des idées de pistes à suivre pour améliorer la gestion de nos projets à venir.

Cependant, la planification initialement prévue dans le rapport de planification [1] a globalement été suivie. Les délais fixés ont été respectés, et les fonctionnalités annoncées ont presque toutes été implémentées, et cela de manière fonctionnelle. Un récapitulatif de cette tenue des objectifs a d'ailleurs été présenté dans la SOUS-SECTION 2.3 de ce rapport, et justifié par la suite.

8 Manuel utilisateur

Bienvenue dans le manuel utilisateur de Glasir, un logiciel d'aide aux experts en sécurité basé sur le formalisme des ADTrees [2] [7]. Grâce à Glasir, vous pourrez analyser efficacement vos ADTrees préalablement créés avec ADTool [6], un logiciel open source disponible sur Internet à l'adresse `http://satoss.uni.lu/members/piotr/adtool/`.

Ce manuel commencera par décrire le fonctionnement général du logiciel (création d'un nouveau projet, ouverture d'un projet existant, etc.), avant d'expliquer comment prendre en main les trois fonctionnalités principales que sont l'Éditeur de fonctions, le Filtre et l'Optimiseur.

Après cela, dans la SOUS-SECTION 8.2 vous trouverez des détails concernant l'utilisation d'ADTool, et principalement sur les nouveautés qui lui ont été ajoutées lors de ce projet.

8.1 Glasir

Glasir est un logiciel d'analyse en sécurité open source, développé pour Windows. Il est fourni sous forme d'exécutable (fichier Glasir.exe) et accompagné d'un dossier contenant des exemples d'ADTrees sous format XML que vous pouvez utiliser pour débuter. Pour lancer le logiciel, veuillez double-cliquer sur Glasir.exe. Si vous rencontrez un souci dès cette étape, merci de lire le fichier ReadMe.txt joint au dossier de téléchargement de Glasir.

Agencement général de la fenêtre Lorsque vous démarrez Glasir, la fenêtre présentée sur la FIGURE 10 s'affiche. Voici une description rapide des éléments que vous trouverez dans cette fenêtre :

- la barre de menu située en haut vous permet d'ouvrir des ADTrees, de sauvegarder des projets, ou encore d'obtenir des indications si vous avez besoin d'aide ;
- le bloc central vous permet d'accéder à l'Éditeur de fonctions (*FunctionEditor*), au Filtre (*Filter*) et à l'Optimiseur (*Optimize*) ;
- la ligne de texte en-dessous du bloc central met en évidence l'ADTree sur lequel vous travaillez. Au démarrage, comme aucun ADTree n'est ouvert, le message « No ADTree selected » est affiché ;
- le bloc en bas, surmonté du message « ADTrees available », indique quels sont les ADTrees appartenant au porjet courant. Ce bloc est vide au démarrage car aucun projet n'est ouvert ;

2. Abréviation d'« Attack-Defense Trees », ou « Arbres d'Attaque et de Défense » en français.

– la case cochable *Show complete path*, tout en bas, permet d'afficher le chemin complet des ADTrees listé dans le bloc situé au-dessus.

Nous allons maintenant détailler le fonctionnement de chacune des fonctionnalités de Glasir.

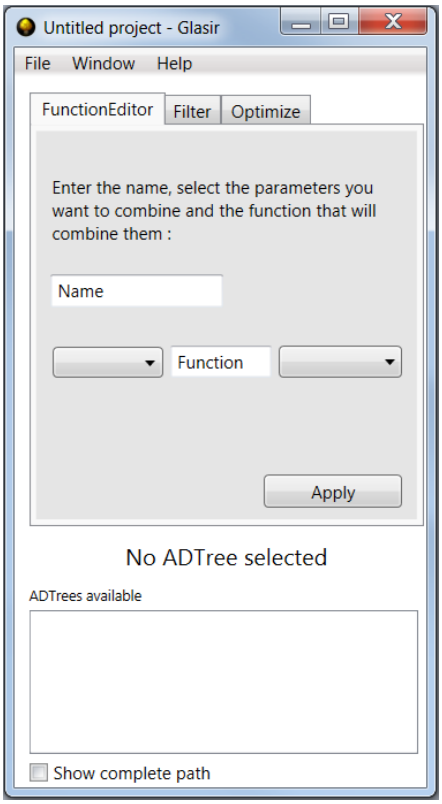

FIGURE 10 – Fenêtre principale s'affichant au démarrage de Glasir.

Ouvrir un ADTree Pour ouvrir un ADTree, cliquez sur le bouton *File* situé tout en haut à gauche, dans la barre de menu. Puis, dans le menu déroulant, cliquez sur *Open ADTree File*, comme indiqué sur la FIGURE 11. Une boîte de dialogue apparaît : parcourez vos dossiers pour sélectionner l'ADTree que vous souhaitez ouvrir. Notez que seuls les ADTrees au format XML générés par ADTool (consulter SECTION 7 du manuel utilisateur d'ADTool [9]) peuvent être analysés par Glasir. Il n'est cependant pas possible d'utiliser des arbres réalisés avec une autre version d'ADTool que celle de Glasir. Une fois l'ADTree sélectionné, cliquez sur le bouton *Open* de la boite de dialogue. ADTool se lance alors pour afficher votre ADTree. Notez qu'au démarrage, cette opération peut prendre quelques secondes. Une fois ADTool lancé, Glasir se présente sous la forme de plusieurs fenêtres, comme indiqué sur la FIGURE 12. Si la fenêtre d'ADTool ne s'affiche pas au bout d'une minute ou qu'un autre programme qu'ADTool se lance, cela signifie que vos fichiers *.jar* ne sont pas associés à Java (ou que vous ne possédez pas Java). Consultez et suivez les étapes du fichier *ReadMe.txt* (en anglais) fourni avec Glasir pour obtenir des informations supplémentaires et corriger le problème.

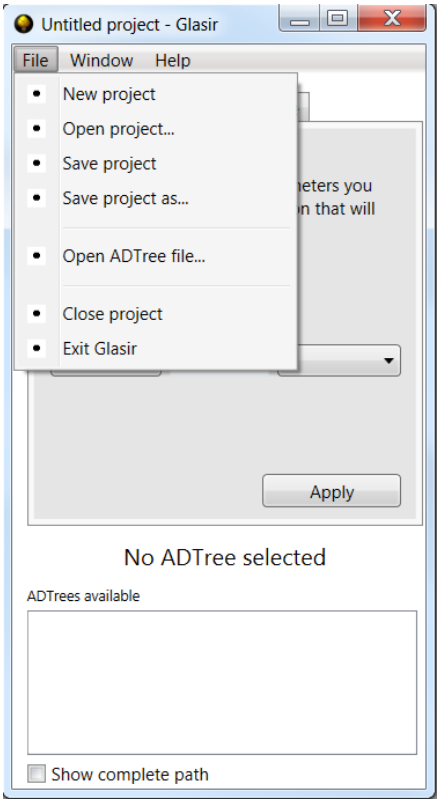

FIGURE 11 – Menu déroulant disponible en cliquant sur le bouton *File* de la barre de menu.

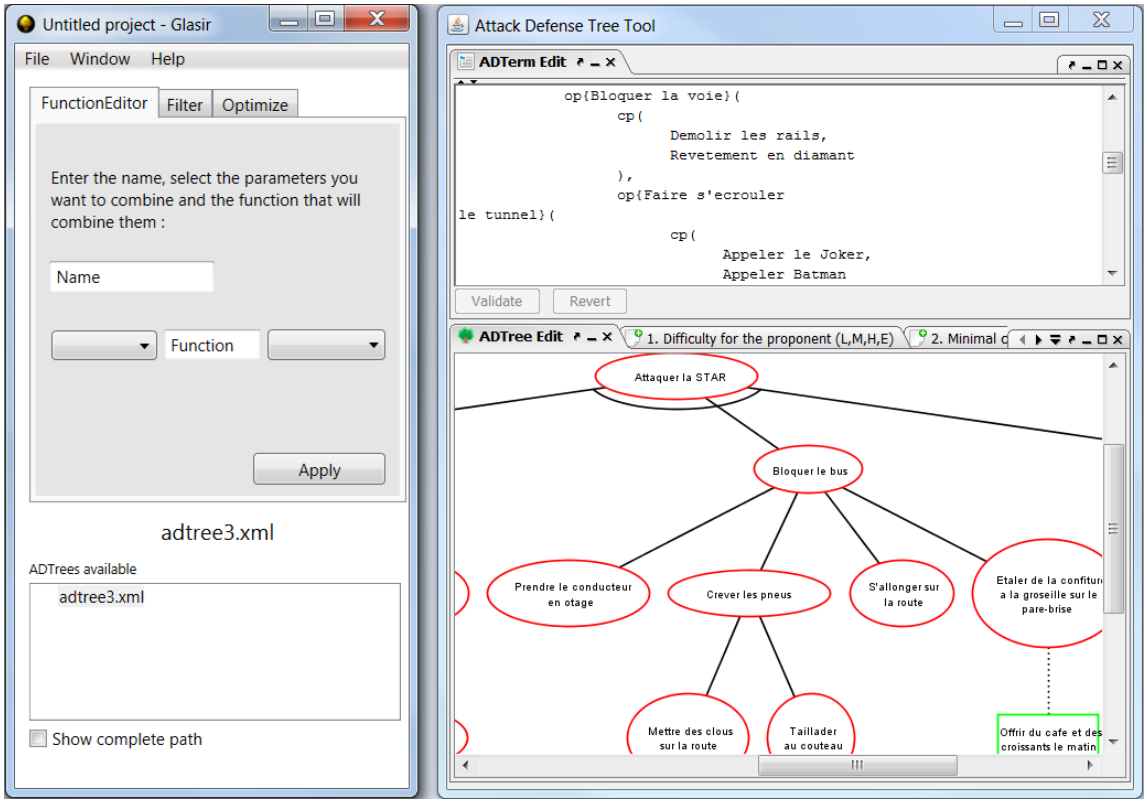

FIGURE 12 – Fenêtres visibles après ouverture d'un ADTree.

Enregistrer un projet Lorsque vous avez ouvert un ou plusieurs ADTrees, vous avez la possibilité de sauvegarder l'état actuel de votre projet, c'est-à-dire la liste des ADTrees ouverts. Pour ce faire, cliquez sur *File*, puis sur *Save project as* pour choisir l'emplacement où enregistrer le fichier du projet. Donnez-lui un nom, puis cliquez sur le bouton *Save* de la boite de dialogue. Une fois ceci fait, le titre de la fenêtre de Glasir portera alors le nom de votre projet. Vous pourrez par la suite cliquer directement sur *Save project* pour enregistrer vos modifications.

Ouvrir un projet Si vous avez déjà un projet enregistré, c'est-à-dire un fichier avec l'extension *.glpf* (*Glasir Project File*), vous pouvez recharger ce fichier : cliquez sur *File*, puis *Open project*. Parcourez vos dossiers pour retrouver votre fichier de projet, sélectionnez-le, puis cliquez sur le bouton *Open* de la boite de dialogue. Si vous aviez un projet ou des ADTrees ouverts, ils seront fermés pour que votre projet enregistré puisse être ré-ouvert. Notez que toute modification non-sauvegardée ne sera pas enregistrée lors de la fermeture.

Aide Si vous avez des problèmes avec le lancement d'ADTool, vous pouvez cliquer sur le bouton *Help* de la barre de menu pour obtenir des informations supplémentaires. Nous vous invitons aussi à suivre les consignes indiquées dans le fichier *ReadMe.txt* (en anglais) fourni avec Glasir.

Changer d'ADTree courant Pour pouvoir utiliser les modules de Glasir, vous devez sélectionner un ADTree courant, sur lequel les opérations des modules seront appliquées. Le bloc en bas de la fenêtre principale dresse la liste des ADTrees de votre projet. Vous avez la possibilité de cocher la case *Show complete path*, en bas, pour voir le chemin complet de chaque ADTree. Cliquez sur un ADTree dans la liste pour qu'il devienne l'ADTree courant. Son nom est alors indiqué à la ligne du dessus, en plus gros. Si aucun ADTree n'est disponible ou sélectionné, le message « No ADTree selected » est affiché à la place. Notez que lorsqu'un nouvel ADTree est ouvert, ce dernier devient automatiquement l'ADTree courant.

Éditeur de fonctions Cette fonctionnalité a pour intérêt d'exprimer un compromis entre deux valuations d'un ADTree en en créant une troisième, fonction des deux premières. Par exemple, si vous avez un ADTree qui a comme paramètres « Minimal cost for the proponent »(en euros) et « Minimal time for the proponent (sequential) »(en heures) et que vous estimez qu'une heure « coûte »par comparaison 20 euros, vous pouvez créer un nouveau paramètre de type « Minimal cost for the proponent »exprimant un compromis entre les deux paramètres qui sera égal à coût+20*temps.

ATTENTION : le nouveau paramètre aura pour type celui du premier paramètre sélectionné (à gauche), par exemple si le premier paramètre est de type « Minimal time for the proponent (sequential) », alors le nouveau paramètre calculé le sera aussi. Si vous sélectionnez un paramètre de type discret, un nombre de cases égal au nombre valeurs possibles s'affichera. Vous pourrez alors pondérer ces valeurs de la plus petite (à gauche), à la plus grande (à droite).

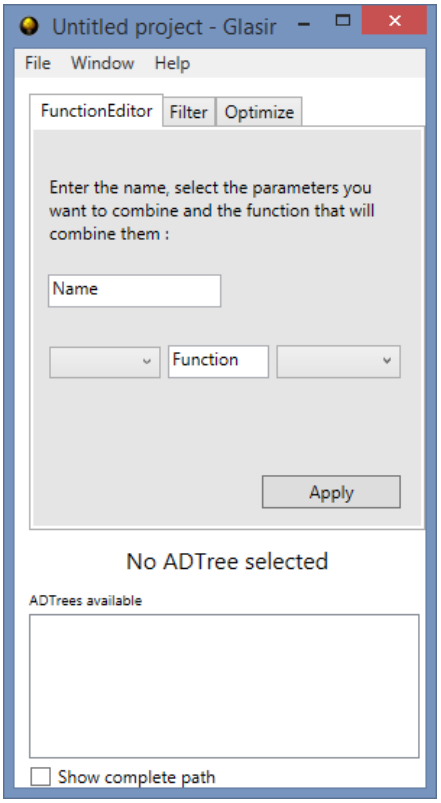

FIGURE 13 – Onglet FunctionEditor.

Pour utiliser l'Éditeur de fonctions, suivez les étapes suivantes :

- Assurez-vous que l'ADTree que vous voulez utiliser est ouvert et désigné comme l'arbre courant ;
- Sélectionnez l'onglet « FunctionEditor » (voir FIGURE 13) ;
- Sélectionnez les deux paramètres de l'ADTree que vous voulez combiner au moyen des deux ComboBox présentes dans l'onglet ;
- Inscrivez dans la textBox « Function » la fonction que vous voulez utiliser pour le calcul du nouveau paramètre. Par exemple, si vous voulez faire « newParam = firstParam + 2*second-Param », inscrivez « +2* » ;
- Entrez le nom que vous voulez donner au nouveau paramètre créé dans la TextBox « Name » ;
- Enfin, cliquez sur le bouton « Apply » pour créer le nouveau paramètre.

S'ouvrira alors un nouvel arbre dans une nouvelle fenêtre d'ADTool, qui sera nommé comme l'arbre initial suivi de « .functEdit » . Le nouvel ADTree sera enregistré par défaut dans le répertoire où se trouve le logiciel Glasir, mais rien ne vous empêche par la suite de le déplacer. Si un problème survient, l'exécution sera stoppée et une boite de message apparaîtra pour vous donner les raisons de l'interruption.

Filtre Cette fonctionnalité sert à élaguer les chemins d'un ADTree qui ne respectent pas une certaine condition. Par exemple, l'ADTree présenté à la FIGURE 14 a comme paramètre « Minimal cost for the proponent »(en euros) : si l'on cherche à identifier les chemins que pourrait emprunter un attaquant potentiel qui ne peut pas dépenser plus de 300 euros, le Filtre permettra alors d'élaguer l'ADTree pour respecter cette condition. L'élagage sera ainsi fait de manière à ne conserver que les feuilles de l'arbre qui apparaissent dans au moins un chemin permettant d'accomplir l'attaque avec 300 euros ou moins. Ce qui conduira à l'arbre présenté à la FIGURE 15

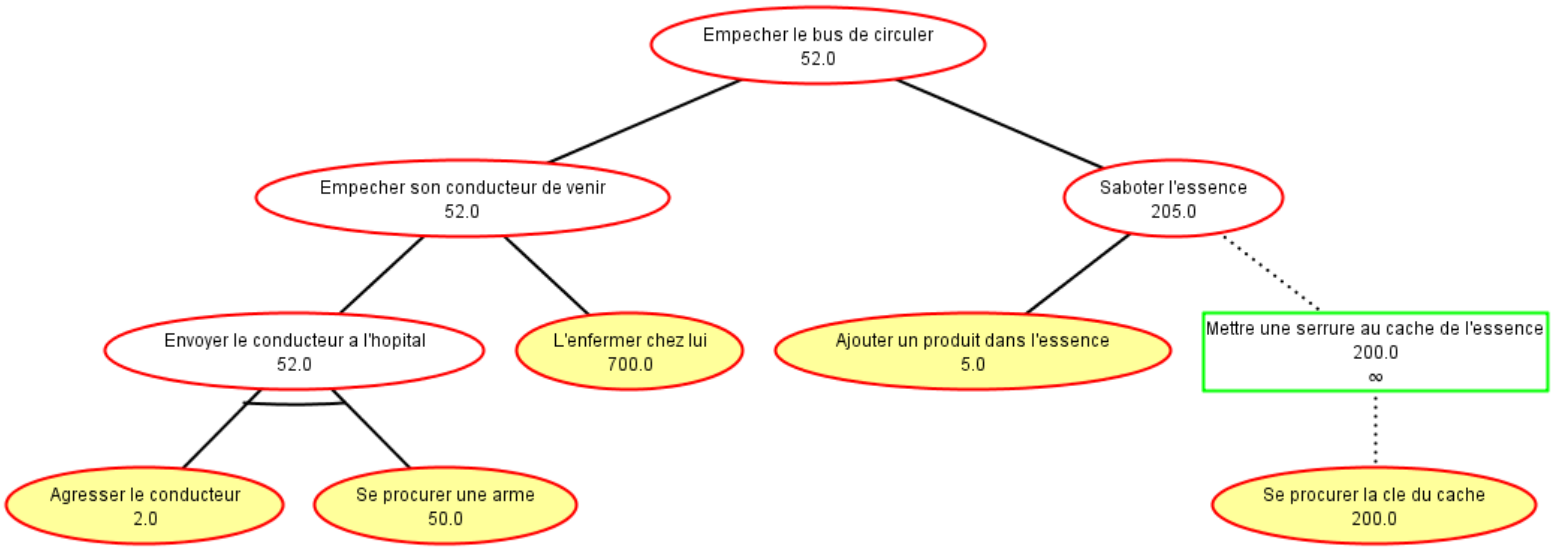

FIGURE 14 – ADTree d'origine.

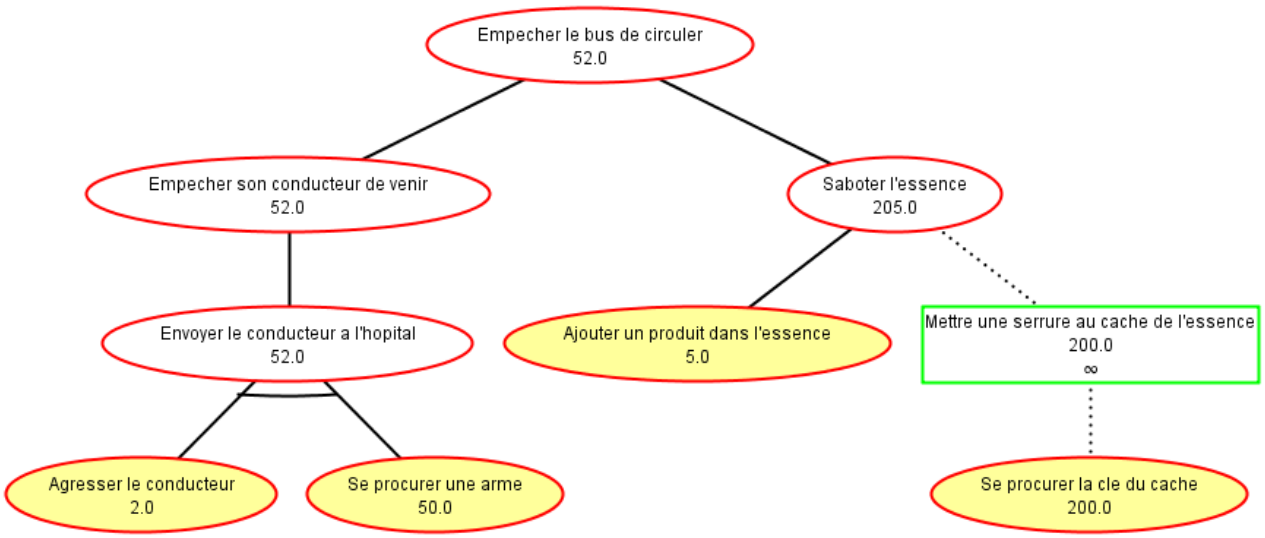

FIGURE 15 – ADTree après filtrage.

L'onglet permettant d'utiliser le filtre dans Glasir est présenté à la FIGURE 16 :

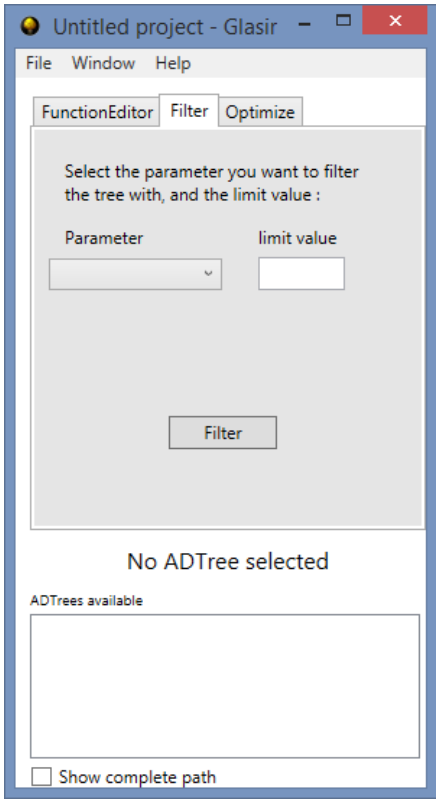

FIGURE 16 – Onglet Filter.

Pour utiliser le Filtre, suivez les étapes suivantes :

- Assurez-vous que l'ADTree que vous voulez filtrer est ouvert et désigné comme l'arbre courant ;
- Sélectionnez l'onglet « Filter » (voir FIGURE 16) ;
- Sélectionnez le paramètre selon lequel vous voulez filtrer l'arbre, grâce à la ComboBox présente dans l'onglet ;
- Indiquer dans la textBox « Limit Value » la valeur limite acceptable (min ou max selon les cas) par le Filtre pour le paramètre sélectionné. Vous pouvez rentrez une valeur textuelle correspondant à l'une des valeurs de l'ensemble du paramètre si l'ensemble est discret ;
- Enfin, cliquez sur le bouton « Filter ».

S'ouvrira alors un nouvel arbre, correspondant à l'arbre élagué, dans une nouvelle fenêtre d'ADTool. Il sera nommé comme l'arbre initial, suivi de « .filter ». Le nouvel arbre sera enregistré par défaut dans le répertoire où se trouve Glasir, mais rien ne vous empêche de le déplacer par la suite. Si un problème survient, l'exécution sera stoppée et une boite de message apparaîtra pour vous donner les raisons de l'interruption.

Optimiseur Cette fonctionnalité sert à élaguer les chemins d'un ADTree de manière a ne conserver que le(s) meilleur(s) chemin(s) selon un paramètre donné. Par exemple avec l'arbre de la FIGURE 14, l'arbre optimisé obtenu selon le paramètre « Minimal cost for the proponent »(en euros) ne conservera que les feuilles de l'arbre qui apparaissent dans au moins un chemin permettant à l'attaquant d'accomplir l'attaque avec 52 euros seulement. Le résultat est l'ADTree de la FIGURE 17

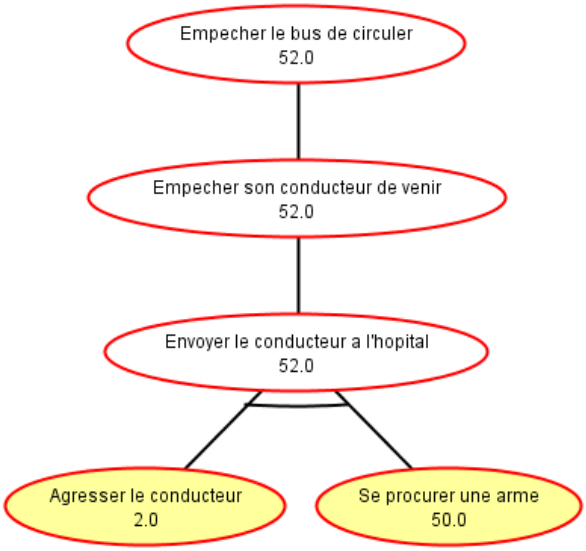

FIGURE 17 – ADTree après optimisation.

L'onglet permettant d'utiliser l'optimiseur dans Glasir est présenté à la FIGURE 18 :

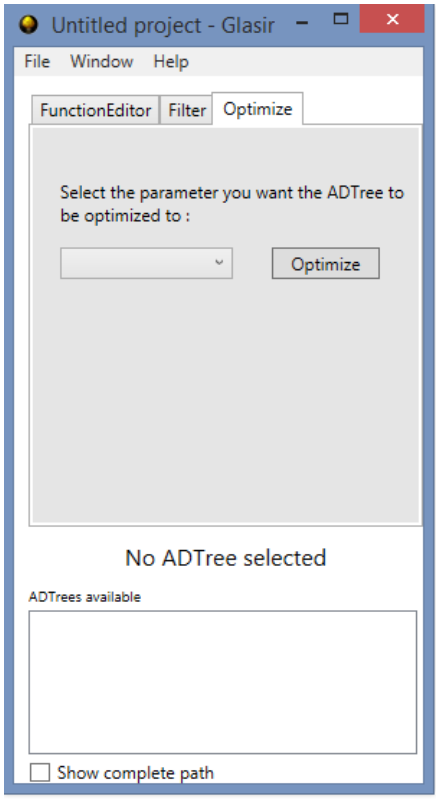

FIGURE 18 – Onglet Optimize.

Pour utiliser l'Optimiseur, suivez les étapes suivantes :

- Assurez-vous que l'ADTree que vous voulez filtrer est ouvert et désigné comme l'arbre courant ;
- Sélectionnez l'onglet « Optimize » (voir FIGURE 18) ;
- Sélectionnez le paramètre selon lequel vous voulez optimiser l'arbre, avec la ComboBox présente dans l'onglet ;
- Enfin, cliquez sur le bouton « Optimize ».

S'ouvrira alors un nouvel arbre correspondant à l'arbre élagué dans une nouvelle fenêtre d'ADTool, qui sera nommé comme l'arbre initial suivi de « .Optimizer ». Le nouvel arbre sera enregistré par défaut dans le répertoire où se trouve Glasir, mais rien ne vous empêche de le déplacer par la suite. Si un problème survient, l'exécution sera stoppée et une boite de message apparaîtra pour vous donner les raisons de l'interruption.

8.2 Nouvelles fonctionnalités d'ADTool

Pour le fonctionnement basique d'ADTool, vous pouvez vous référer au manuel utilisateur officiel [9] disponible sur Internet [3]. Le guide ici présent ne détaillera que l'utilisation des nouvelles fonctionnalités d'ADTool, qui sont le copier/couper/coller ainsi que l'annulation d'une action.

3. Voir à l'adresse suivante : http ://satoss.uni.lu/members/piotr/adtool/manual.pdf

Copier/couper/coller Ces fonctionnalités vous seront utiles si vous désirez couper/copier un sous-arbre de l'ADTree courant afin de le coller ensuite à un autre emplacement dans ce même ADTree. Il est à noter que le sous-arbre coupé/copié sera ajouté en tant que fils du nœud auquel il sera collé. Aussi, la racine du sous-arbre coupé/copié doit être du même type (opponent ou proponent) que son futur nœud parent. Pour couper/copier un sous-arbre puis le coller, suivez les étapes suivantes :

1. Sélectionnez à l'aide d'un clic gauche le nœud racine du sous-arbre que vous désirez couper/copier. Si vous avez déjà sélectionné un nœud dans l'arbre, vous pouvez également vous déplacer jusqu'au nœud souhaité à l'aide des flèches (haut, bas, gauche, droite) du clavier.
2. Effectuez un clic droit sur le nœud sélectionné. Un menu déroulant doit apparaître à côté du nœud sélectionné, comme sur la FIGURE 19.
3. Sélectionnez l'option « Copy Subtree »/« Cut Subtree ») dans le menu déroulant. Vous pouvez également effectuer cette étape à l'aide d'un raccourci clavier, CTRL+C/CTRL+X.
4. Effectuez un clic droit sur le nœud auquel vous voulez coller le sous-arbre coupé/copié, puis sélectionnez dans la liste déroulante « Paste Subtree as Child ». Si cette option n'apparaît pas dans le menu déroulant, c'est que vous n'avez pas préalablement coupé/copié de sous-arbre. Vous pouvez ici encore utiliser un raccourci clavier, CTRL+V.

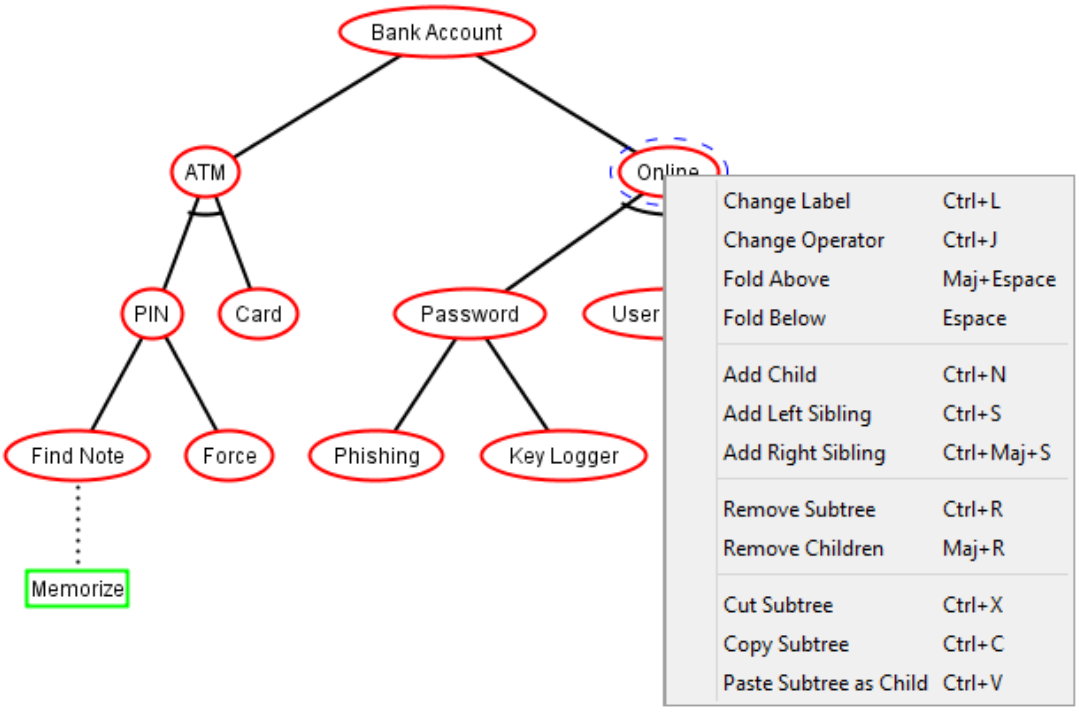

FIGURE 19 – Menu déroulant apparaissant après un clic droit sur un nœud.

Annulation d'une action Il s'agit ici d'annuler une ou plusieurs action(s) effectuée(s) précédemment sur l'ADTree courant. Pour cela, il vous suffit tout simplement d'utiliser le raccourci clavier CTRL+Z autant de fois que nécessaire, jusqu'à retrouver l'état souhaité pour l'ADTree. Vous pouvez également cliquer sur l'icône « Undo last action (CTRL+Z) » en haut à gauche de la fenêtre principale d'ADTool, encadrée en rouge sur la FIGURE 20. Les actions pouvant être annulées sont les changements de labels, les ajouts ou suppressions de nœuds, les changements d'opérateur (conjonction ou disjonction) ainsi que les actions de couper/copier/coller.

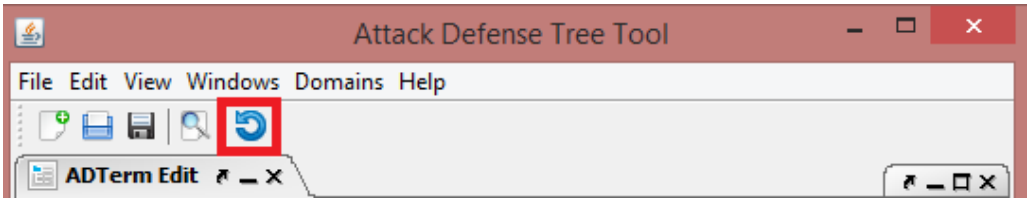

FIGURE 20 – Icône permettant d'annuler l'action précédente.

Références

[1] Pierre-Marie Airiau, Valentin Esmieu, Hoel Kervadec, Maud Leray, Florent Mallard, and Corentin Nicole. Glasir - Rapport de planification, décembre 2014. Rapport de spécification fonctionnelle, projet de quatrième année, INSA de Rennes.

[2] Pierre-Marie Airiau, Valentin Esmieu, Hoel Kervadec, Maud Leray, Florent Mallard, and Corentin Nicole. Glasir - Rapport de spécifications fonctionnelles, novembre 2014. Rapport de spécifications fonctionnelles, projet de quatrième année, INSA de Rennes.

[3] Pierre-Marie Airiau, Valentin Esmieu, Hoel Kervadec, Maud Leray, Florent Mallard, and Corentin Nicole. Rapport de pré-étude, octobre 2014. Rapport de pré-étude, projet de quatrième année, INSA de Rennes.

[4] Pierre-Marie Airiau, Valentin Esmieu, and Maud Leray. Glasir - Rapport de bilan de planification, mai 2015. Rapport de bilan de planification, projet de quatrième année, INSA de Rennes.

[5] Pierre-Marie Airiau, Valentin Esmieu, and Maud Leray. Glasir - Rapport de conception, février 2015. Rapport de conception logicielle, projet de quatrième année, INSA de Rennes.

[6] Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer. ADTool : Security Analysis with Attack–Defense Trees. In Kaustubh R. Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio, editors, *QEST*, volume 8054 of *LNCS*, pages 173–176. Springer, 2013.

[7] Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patrick Schweitzer. Attack–Defense Trees. *Journal of Logic and Computation*, 24(1) :55–87, 2014.

[8] Piotr Kordy and Patrick Schweitzer. The ADTool. `http://satoss.uni.lu/software/adtool`, 2012.

[9] Piotr Kordy and Patrick Schweitzer. The ADTool Manual. `http://satoss.uni.lu/software/adtool/manual.pdf`, 2012.

[10] Andy Orchard. *Dictionary of Norse Myth and Legend*. Cassell, 1997.

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