|  |
| --- |
| **univ-lorraineMacintosh HD:Users:festor:Desktop:Logos Lorraine INP:LOGO LINP-  Couleur.gif** |
| Rapport de stage |
| *Etude expérimentale portant sur la collaboration basée sur la confiance* |
|  |
| ***David Rubino*** |
|  |

**Année 2013-2014**

Stage de 2e année réalisé dans l’entreprise *Inria*



Maître de stage : *Claudia Ignat*  
Encadrant universitaire : *Rémi Badonnel*

**Déclaration sur l’honneur de non-plagiat**

**Je soussigné(e),**

**RUBINO David**

**Elève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy**

**N° de carte d’étudiant(e) : 31010226**

**Année universitaire : 2013 - 2014**

**Auteur(e) du document, mémoire, rapport ou code informatique intitulé :**

**Etude expérimentale portant sur la collaboration basée sur la confiance**

Par la présente, je déclare m’être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu’il a été rédigé entièrement par mes soins. J’affirme n’avoir ni contrefait, ni falsifié, ni copié tout ou partie de l’œuvre d’autrui, en particulier texte ou code informatique, dans le but de me l’accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l’Université, et qu’en cas de manquement aux règles en la matière, j’encourrais des poursuites non seulement devant la commission de discipline de l’établissement mais également devant les tribunaux de la République Française.

**Fait à Nancy le 7/ 7/ 2014**

**Signature :**

**Remerciements**

Je tiens tout d’abord à remercier mon maître de stage, Madame Claudia-Lavinia IGNAT, qui m’a encadré tout en long de mon stage, et m’a apporté l’aide et le soutien nécessaires pour me permettre d’avancer et de mener ma mission à bien,

Monsieur Mounir SADOUI, étudiant en Master 2 à l’Université de Lorraine, avec qui j’ai étroitement collaboré pour le bon développement de l’expérience, et qui m’a notamment aidé pour mieux comprendre et programmer avec le logiciel z-Tree,

Madame Valerie SHALIN, professeur de psychologie à Wright State University, qui nous a aidés lors de l’élaboration du protocole de l’expérience,

Madame Olivia FOX, doctorante à Wright State University, qui nous a également aidés pour l’élaboration du protocole de l’expérience,

Monsieur Mehdi AHMED-NACER, thésard à l’Université de Lorraine, qui nous a aidés à élaborer la première version de l’algorithme du calcul de la confiance.

Je remercie tous les membres de l’équipe que j’ai eu l’occasion de côtoyer et qui m’ont apportés leur soutien tout au long de mon stage.

Enfin, je souhaite remercier les membres du jury d’avoir bien accepté de juger ce travail.

**Résumé**

Très souvent sur l’Internet, des utilisateurs sont amenés à collaborer sur un projet commun qui requiert à cette communauté de donner accès aux donnés à des utilisateurs. Pendant cette collaboration, les utilisateurs peuvent construire une relation de confiance qui les encouragera à travailler ensemble plus tard. Ici, on montre l’impact d’un indice de confiance par utilisateur à l’aide d’une expérience qui met en relation une communauté d’utilisateurs qui s’échangeront de l’argent.

Pour implémenter cette expérience, z-Tree a été utilisé pour garantir un environnement qui a déjà été testé pour ce genre d’expérience. Deux jeux ont été implémenté : le premier sera un simple échange entre les utilisateurs sans l’aide de données externes ; le second implémentera l’algorithme de Trust et ainsi aidera les utilisateurs à faire de meilleurs choix quant à l’argent à envoyer à leur partenaire.

Les résultats de cette expérience permettront aux psychologues d’évaluer l’efficacité de l’algorithme de Trust dans cette implémentation, et ainsi les aideront à faire des assertions quant à l’influence de l’indice de confiance lors de la collaboration entre utilisateurs dans une communauté.

**Mots-clés :** *Collaboration, algorithme de Trust, z-Tree*

**Abstract**

Very often on the Internet, users are brought together to collaborate on a common project that requires the community to let access to the data to specific users. During their collaboration, users can build a trust relationship that will encourage them to work together again in the near future. Here, the impact of a trust index per user is shown, using an experiment that brings together a community of users who will exchange money.

To implement this experiment, z-Tree was used to guarantee an environment that has already been proved for these kinds of experiments. Two games are implemented: one will be a simple exchange between users without any data help; the second one will implement the Trust Game algorithm and thus will help users make wiser decisions regarding the money to send to their partner.

Results of this experiment will enable psychologists to evaluate the efficiency of the Trust Game algorithm in this implementation, and then help them to make assumptions regarding the influence of this trust index in the collaboration between users in a community.

**Keywords:** *Collaboration, Trust Game algorithm, z-Tree*

Table des matières

[Introduction 5](#_Toc394583309)

[1 Présentation d’Inria 6](#_Toc394583310)

[1.1 Activité principale 6](#_Toc394583311)

[1.2 Situation géographique et environnement socio-économique 6](#_Toc394583312)

[1.3 Organigramme simplifié 7](#_Toc394583313)

[1.4 Données économiques, humaines et juridiques 8](#_Toc394583314)

[1.5 Stratégie de développement 8](#_Toc394583315)

[1.6 Service d’accueil durant le stage 9](#_Toc394583316)

[2 Mission du stage 9](#_Toc394583317)

[2.1 Présentation du contexte 9](#_Toc394583318)

[2.2 Protocole de l’expérience 11](#_Toc394583319)

[2.2.1 Description générale 11](#_Toc394583320)

[2.2.2 Détermination de l’indice de confiance 13](#_Toc394583321)

[2.2.3 Aspect utilisateur 15](#_Toc394583322)

[2.3 Implémentation 15](#_Toc394583323)

[2.3.1 Implémentation en Java 15](#_Toc394583324)

[2.3.2 Implémentation en z-Tree 18](#_Toc394583325)

[2.3.3 Déploiement réseau 24](#_Toc394583326)

[2.3.4 Solution choisie pour l’expérience 25](#_Toc394583327)

[2.4 Résultats de l’expérience 25](#_Toc394583328)

[Conclusion 26](#_Toc394583329)

[Bibliographie 27](#_Toc394583330)

[Annexes 28](#_Toc394583331)

[1. Détermination de l’algorithme du Trust 28](#_Toc394583332)

[2. Code Java 29](#_Toc394583333)

[3. Code z-Tree 30](#_Toc394583334)

[4. Parameter Table 33](#_Toc394583335)

Introduction

Lors de ma 2e année a TELECOM-Nancy, j’ai eu l’opportunité de réaliser un stage dans le centre de recherche Nancy – Grand Est d’Inria à Villers-lès-Nancy. Durant mon stage, j’ai travaillé avec des membres de l’équipe SCORE sur une étude expérimentale basée sur la collaboration portant sur la confiance. Cette étude, menée en partenariat avec Wright State University (Ohio, USA), avait pour but de déterminer le rôle accordé à la confiance lors de la collaboration de plusieurs utilisateurs sur un même projet.

Aujourd’hui, de nombreux systèmes d’Internet utilisent des ressources partagées entre plusieurs utilisateurs qui peuvent être modifiées librement. Plusieurs utilisateurs ont ainsi la possibilité de collaborer sur un même projet. C’est le cas par exemple des logiciels de gestion de version comme Git ou SVN. Dans ces logiciels, les processus d’authentification sont régis par des modèles de sécurité traditionnels qui peuvent parfois être lourd à implémenter mais aussi à utiliser. L’idée pourrait être de trouver une alternative à ces modèles de sécurité en utilisant un autre modèle basé sur la confiance entre utilisateurs. Pour observer le comportement d’utilisateurs sur un tel modèle, il est nécessaire de conduire une expérience pour recueillir des résultats à analyser.

Certaines expériences ont déjà été conduites sur l’étude de la confiance entre utilisateurs, comme une réalisée par Joyce Berg, professeur à l’Université d’Iowa. Cette dernière a servi de point de référence pour le développement de notre propre expérience tout au long du projet.

Dans un premier temps, je présenterai l’activité d’Inria ainsi que sa situation géographique et ses points stratégiques. Je décrirais ensuite la mission qui m’a été confiée tout au long du stage, tout d’abord en détaillant le protocole de l’expérience que nous avons conduit, puis ensuite en parlant de l’implantation des logiciels qui ont été nécessaires pour réaliser cette expérience.

# Présentation d’Inria

## Activité principale

Inria est un institut public de recherche créé en 1967 à Rocquencourt, près de Paris. Entièrement dédiées aux sciences du numérique, les activités d’Inria s’articulent autour de trois points fondamentaux : la recherche, le transfert de technologies qui permet de mettre en œuvre les résultats de recherche au niveau de l’industrie, et le partage et la transmission des connaissances, notamment grâce à environ un millier d’étudiants qui viennent travailler ou se former parmi des équipes. [1]

Au niveau de la recherche, Inria articule actuellement ses recherches autour de trois principaux thèmes : la sureté et sécurité des systèmes informatique, la cognition, et la simulation, optimisation et contrôle des systèmes complexes. D’autres institutions travaillent conjointement avec Inria dans ces buts de recherche. Parmi eux on trouve le CNRS, l’Université de Lorraine, l’Université de Strasbourg, l’Université de Franche-Comté ou encore le Max Planck Institut fur Informatik.

Le transfert de technologies permet de mettre en œuvre les résultats de recherche au niveau d’industries stratégiques, comme par exemple les télécommunications, la santé, les transports, l’énergie, l’aéronautique, la défense. Inria compte de nombreux partenaires industriels, notamment Microsoft Research et Alcatel-Lucent Bell Labs avec qui ils possèdent un laboratoire commun. Total, EDF R&D ou encore France-Telecom-Orange sont d’autres partenaires du groupe.

## Situation géographique et environnement socio-économique

Le siège d’Inria se trouve à Rocquencourt, près de Versailles. Inria possède des centres de recherche dans toute la France, notamment à Bordeaux, Grenoble, Sophia Antipoplis, ou encore Nancy, le centre où j’ai eu l’occasion d’effectuer mon stage. Le centre de recherche Nancy – Grand Est se trouve sur le campus science de l’Université de Lorraine à Villers-lès-Nancy.

Les centres Inria sont situés à proximité des universités afin de pouvoir collaborer avec les pôles de recherche de ces dernières. De plus, cela permet un meilleur contact avec les étudiants et l’intégration de jeunes talents dans les équipes de recherche pour poursuivre un doctorat ou une carrière dans la recherche une fois diplômés.

Les villes dans lesquelles sont implantées Inria sont aussi des pôles permettant un lien étroit avec l’industrie. Par exemple, le centre de Sophia-Antipolis peut ainsi collaborer avec des entreprises locales et leur permettre d’appliquer leurs résultats de recherche sur un nouveau produit. Cela rejoint donc l’un des objectifs fondamentaux d’Inria, à savoir le transfert de technologies de la recherche à l’industrie.

## Organigramme simplifié

La figure 1 représente l’organigramme du laboratoire Inria au niveau national.



*Fig. 1 : Organigramme simplifie d’Inria au niveau national* [2]

## Données économiques, humaines et juridiques

Au niveau national, Inria emploie 4397 personnes dont 1412 chercheurs et maitre de conférences, et 1247 doctorants. Chaque année, 730 étudiants rejoignent Inria pour des stages de fin d’étude. Sur Nancy, ce sont plus de 500 personnes qui sont accueillies, avec plus de 50 nationalités représentées. Le centre de Nancy – Grand Est regroupe une vingtaine d’équipes de recherche et a permis, depuis 2000, la création de 12 start-up. [1]

En 2012, Inria a reçu 104.5 millions d’euro de subventions de la communauté Européenne. Le budget d’Inria s’estimait à 264.3 millions d’euros en 2012, avec 5402 publications écrites cette année-là, dont 1831 articles.

## Stratégie de développement

Inria cherche à avoir une présence régionale au niveau de la France. En 2012, de nouveaux bâtiments ont été mis à la disposition d’Inria à Bordeaux, Lille, et Saclay. Cette présence régionale permet à de nombreux étudiants mais aussi industries de bénéficier des services des centres à différents niveaux collaboratifs. Certaines structures sont également crées en partenariat avec le gouvernement Français. Selon les mots de Michel Cosnard, directeur d’Inria : « Nous sommes des partenaires dans un nombre de projets à grandes échelles qui donnent un véritable coup de fouet pour notre expertise scientifique mais aussi pour nos activités de transfert de la technologie. » [1] Cette stratégie permet, en France mais aussi dans d’autres pays comme le Chili, d’effectuer un développement en fonction des ressources de la région ou du pays. Inria a donc une stratégie de localisation qui lui permet de s’adapter et diriger ses recherches en fonction du potentiel présent dans la zone géographique du centre.

Une autre spécificité de la stratégie de développement d’Inria est l’importance accordée à l’impact sociétal des recherches encourues. Aujourd’hui, les champs d’application des projets de recherche concernent tous les domaines de l’activité humaine, ce qui reflète bien le développement de l’informatique et des sciences du numérique dans tous les aspects de notre société. Dans ces cas-là, il est important de s’interroger sur l’aspect éthique des applications de recherches en cours. Pour cela, Inria a créé le COERLE (Comité Opérationnel d’Evaluation des Risques Légaux et Ethiques) afin de déterminer les risques potentiels que peuvent générer certains projets.

En tant que contributeur à un monde de technologies du numérique, l’ambition d’Inria est d’avoir « un impact économique et social en plus d’un impact scientifique. » (Antoine PETIT, directeur général adjoint) [1]

## Service d’accueil durant le stage

Durant mon stage, j’ai intégré une équipe du LORIA (Laboratoire Lorrain de Recherche en Informatique et ses Applications), unité de recherche commune à Inria, au CNRS, et à l’Université de Lorraine [3]. J’ai travaillé dans l’équipe SCORE, dirigée par François Charoy. Cette équipe articule ses travaux autour de l’hébergement d’entreprises distribuées sur l’Internet. [4] En particulier, elle possède 3 axes thématiques de recherche :

* Systèmes collaboratifs distribués
* Gestion des processus « business » et service informatique
* Interopérabilité et modélisation d'entreprise

L’équipe comprend une vingtaine de personnes. J’ai personnellement eu l’occasion de travailler sur le problème de la collaboration liée à la confiance, dont s’occupe mon maitre de stage, Claudia Ignat. Ce projet est mené en partenariat avec Wright State University, Ohio, qui a détaché deux personnes dans l’équipe SCORE pour la réalisation du projet. Un étudiant de master et moi-même étions chargés d’implémenter les solutions et outils informatiques nécessaires au bon déroulement du projet.

# Mission du stage

## Présentation du contexte

Le projet de recherche s’intéresse aux problèmes de collaboration entre plusieurs utilisateurs en fonction de la confiance. On cherche à savoir quel sera le comportement des utilisateurs si ceux-ci doivent collaborer sur un même projet, dans un objectif commun. L’intérêt est de montrer par l’expérimentation s’ils sont plus intéressés par un objectif personnel que par l’objectif final. Afin de montrer cela, l’équipe Score a envisagé de réaliser une expérience avec de véritables utilisateurs pour recueillir des résultats et ensuite les analyser.

Ces problèmes de collaboration ont déjà fait le sujet de nombreuses études. La plus célèbre a été conduite par Joyce Berg pour l’Université d’Iowa [5]. Dans cette expérience, plusieurs sujets étaient sélectionnés et disposés dans deux salles différentes (Room A et B). Une troisième salle (Room C) contient des enveloppes. Chaque utilisateur dispose à la base de $10. Aléatoirement, à chaque utilisateur de Room A est associé un utilisateur de Room B. Le but de l’expérience était, pour chaque paire d’utilisateur, de s’échanger de l’argent par l’intermédiaire d’enveloppes contenues dans Room C. L’utilisateur peut soit collaborer et donner ses $10 à son partenaire – en totalité ou en partie –, soit garder l’argent pour lui. Dans le cas où il y a collaboration, la somme reçue par le deuxième joueur est le triple de la somme donnée par le premier. Ce deuxième utilisateur a ensuite soit le choix de garder l’argent pour lui, soit de le renvoyer – en totalité ou en partie – à son partenaire de jeu. Cependant, une condition est imposée : on ne peut renvoyer qu’au maximum le montant que l’autre utilisateur a envoyé. Le schéma ci-dessous détaille une telle simulation pour 2 échanges entre 2 utilisateurs.

Etape 1

**Utilisateur 2**

Somme initiale: $10

**Utilisateur 1**

Somme initiale: $10

Envoi de $7

Etape 2

**Utilisateur 2**

Nouvelle somme: $31 = $10 de début + 3\*$7 envoyés

Envoi de $11

**Utilisateur 1**

Nouvelle somme: $3

Etape 3 : fin du jeu

**Utilisateur 2**

Nouvelle somme: $20

**Utilisateur 1**

Nouvelle somme: $14

On remarque que, dans une telle situation, si le processus de collaboration est respecté, l’utilisateur 1 ne perd pas d’argent et l’utilisateur 2 en gagne. La confiance est donc établie entre les 2 utilisateurs.

D’autres alternatives à cette expérience sont possibles. On peut par exemple associer à chaque utilisateur un indice de confiance qui augmentera ou diminuera en fonction de son aptitude à collaborer. Les autres utilisateurs pourront voir l’indice de confiance de leur partenaire et celui-ci permettra donc de prendre une décision concernant l’argent qu’ils enverront. [6] Une autre option est de montrer aux utilisateurs un histogramme des parties précédentes, en affichant la somme envoyée par chaque utilisateur ainsi que le triple de la somme envoyée par l’initiateur. Ce sont ces différents paramètres que nous avons pris en compte dans la réalisation du protocole de l’expérience.

Les applications possibles de tels systèmes se trouvent dans les systèmes collaboratifs, comme par exemple le Peer-to-Peer. En effet, en estimant la confiance d’un utilisateur, la communauté peut se protéger elle-même des personnes refusant toute collaboration et bénéficiant du service des autres pour son propre intérêt. Ainsi, les utilisateurs pourront utiliser cet indice de confiance comme métrique pour déterminer avec quels utilisateurs ils veulent collaborer ou pas.

## Protocole de l’expérience

### Description générale

L’expérience s’inspirera du modèle de Berg en utilisant quelques variantes. Comme chez Berg, des utilisateurs seront mis en binôme et s’échangeront de l’argent pendant une période. Cependant, tous les joueurs humains joueront avec des joueurs programmés à l’avance, ceci afin de mieux mesurer la réaction des utilisateurs humains lorsqu’ils enverront de l’argent. Toutefois, les utilisateurs ne sauront pas qu’ils jouent avec des utilisateurs programmés.

Un utilisateur peut être soit initiateur, soit récepteur. L’utilisateur ne choisit pas son rôle, celui-ci lui est assigné à l’avance dans le programme. Il est important que chaque joueur humain soit autant de fois initiateur que récepteur afin que son comportement soit évalué plus justement.

Chaque utilisateur commence avec une somme initiale de $10. L’initiateur envoie une somme d’argent au récepteur. Le récepteur reçoit la somme envoyée par l’initiateur multipliée par trois. Il peut ensuite renvoyer une somme à l’initiateur. Cette somme peut être nulle, mais il ne peut pas renvoyer plus que l’argent envoyé par l’initiateur multiplié par trois.

Dans le cas des utilisateurs simulés, la somme envoyée n’est pas laissée au hasard. Lorsque l’utilisateur simulé est récepteur, il y aura 2 cas : la moitié des utilisateurs simulés renverront 10% de la somme reçue, tandis que les autres renverront 70% de la somme reçue. Lorsque l’utilisateur simulé est initiateur, on calcule l’indice de confiance du partenaire et, en fonction de sa valeur, l’utilisateur simulé envoie à son partenaire une somme fixe.

Le schéma d’échange sera répété pendant un certain nombre de périodes. Ce nombre de périodes est déterminé par le nombre d’utilisateurs présents lors de l’expérience. Il est nécessaire qu’il y ait suffisamment de données recueillies pour que l’analyse puisse conduire à des résultats cohérents. Il a été décidé de conduire l’expérience avec 8 utilisateurs, dont 4 seraient simulés. Tous les utilisateurs humains doivent jouer avec des utilisateurs simulés, il est donc interdit que 2 humains jouent entre eux. De plus, ils doivent être récepteurs et initiateurs au moins une fois. Il y a donc 8 associations possibles qui satisfont à ces conditions. Pour une meilleure cohérence des résultats, il est important de répéter ce schéma une fois, ce qui donne finalement un nombre de périodes égal à 16.

Deux jeux sont disponibles. Le premier implémente le protocole décrit ci-dessus. Le deuxième est identique au premier à l’exception que cette fois-ci, l’indice de confiance sera connu des utilisateurs. En effet, auparavant, seul le système le connaissait afin de savoir quelle somme devait être envoyée par les utilisateurs simulés lorsque ceux-ci initiaient la partie. Dans le deuxième cas, les utilisateurs humains y ont aussi accès, ce qui permettra de déterminer l’influence que cet indice a sur la réaction des utilisateurs.

Auparavant, une troisième variante avait été intégrée. Celle-ci consistait à afficher un histogramme qui permettrait ainsi aux utilisateurs de voir ce que leur partenaire a joué lors de la période précédente. Cette mesure pourrait donc modifier le comportement des joueurs. Cependant, il a été décidé de ne pas mettre cet histogramme dans l’expérience finale. D’une part, cela aurait rallongé l’expérience, et d’autre part le but principal de l’expérience étant de valider la métrique de confiance, un jeu avec l’histogramme n’aurait pas été très utile pour réaliser cet objectif-là.

Chaque sujet reçoit une somme d’argent pour sa participation à l’expérience. Dans l’expérience de Berg, les sujets recevaient tous la même somme. Cependant, nous souhaitons tenir compte lors du paiement de la performance des utilisateurs dans le jeu. Pour cela, on applique un taux de 0.01 à l’argent gagné durant le jeu. En fonction de la somme obtenue, on procède au paiement selon l’algorithme suivant.

si argent\_gagné <= 5 :

payer 0

si 5 < argent\_gagné <= 15 :

payer 10

si 15 < argent\_gagné <= 25 :

payer 20

si argent\_gagné > 25 :

payer 30

Le protocole global de l’expérience a été établi et soumis pour approbation à la psychologue responsable du projet à Wright State University. L’expérience initiale ainsi que sa variante constituent les jeux testés par les utilisateurs.

### Détermination de l’indice de confiance

Nous avons proposé plusieurs solutions pour la métrique de confiance. La première à avoir été implémenté suivait l’algorithme suivant (1) :

if (0 <= argent reçu < argent envoyé)

indice de confiance += argent reçu / argent envoyé – 1

if (indice de confiance < 0)

indice de confiance = 0

if (argent envoyé < argent reçu <= 3\*argent envoyé)

indice de confiance += argent reçu / (3\*argent envoyé)

if (indice de confiance > 0)

indice de confiance = 1

Le problème de cette méthode est qu’il est possible d’avoir une même valeur de trust dans des cas différents. Une autre alternative (celle que nous avons retenue) est de modéliser le trust en fonction de l’argent reçu par un modèle linéaire (cf. fig. 2).

*Fig. 2 : Graphique représentant l’évolution du trust en fonction de la contribution*

En déterminant les équations de la droite, nous obtenons nos formules de calcul du trust (voir annexe 1 pour la démonstration). Le nouvel algorithme est le suivant (2) :

if money returned < money received:

if :

Cet algorithme permet de calculer le trust de base. C’est celui qui est utilisé lorsque l’utilisateur joue lors de la première période de jeu. Dans les périodes suivantes, un autre algorithme est nécessaire afin de prendre en compte les trusts précédant lors du nouveau calcul. Cet algorithme est donné par A. Das et M.M. Islam dans l’une de leurs publications [6] (3).

Previous trust correspond au trust de la période précédente.

Current trust est la nouvelle valeur de trust calculé à partir de l’algorithme (2).

Le coefficient c est compris entre 0 et 1 et indique si l’on donne plus d’importance à l’ancienne valeur du trust ou à la nouvelle. Dans notre cas, c = 0.5.

x est appelle déviation et calcule la variation entre l’ancienne valeur du trust et la nouvelle.

Le coefficient est appelle poids : il correspond à la modulation qu’on accorde au trust. Lors de notre première implémentation, nous avons choisi un coefficient de 0.5.

A l’aide de tous ces paramètres, nous pouvons obtenir une bonne modulation du trust qui prend en compte l’historique de confiance du joueur. Cela permet de refléter une vision plus juste de la confiance du joueur au niveau global du jeu et pas seulement sur une période.

### Aspect utilisateur

Du côté de l’utilisateur, il est nécessaire d’avoir des instructions précises à donner. Les utilisateurs sont juste prévenus qu’ils participent à une expérience mais n’en savent pas plus. Pour mieux les aider à appréhender l’expérience, nous leur donnons des instructions qu’ils lisent avant de commencer l’expérience. S’ils ont la moindre question, ils lèvent la main et l’un des membres de l’équipe vient leur répondre.

En plus de recueillir des résultats, nous donnons un questionnaire à remplir à la toute fin de l’expérience. Ce questionnaire est implémenté à l’aide du logiciel z-Tree (voir 2.3.2). Cela permet d’avoir des retours sur l’expérimentation et permettra de prendre en compte certains paramètres à l’avenir.

## Implémentation

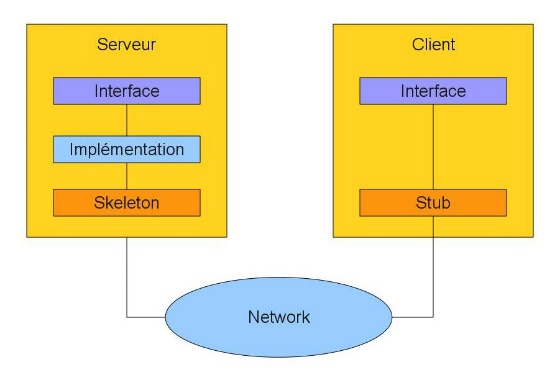
Pour réaliser l’expérience, deux implémentations étaient possibles : implémenter une interface en Java ou utiliser z-Tree, un logiciel utilisé dans la recherche fournissant une interface et permettant de programmer des expériences. Les deux solutions ont été développées pour voir laquelle était la mieux adaptée pour les utilisateurs lors de l’expérience et voir les limites que chaque solution représente. Dans un premier temps, je détaillerais la structure du logiciel Java, puis je parlerais de l’implémentation avec z-Tree.

### Implémentation en Java

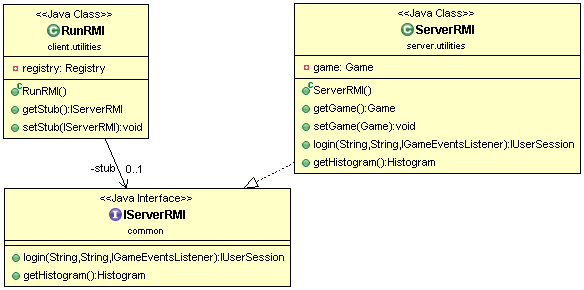
Le logiciel est décomposé en 3 parties principales : une partie serveur, une partie client et une partie commune – utilisée notamment pour tout ce qui concerne la sauvegarde des données.

Pour réaliser la connexion client-serveur, on implémente l’interface RMI fournie par Java et qui permet d’interagir entre un serveur et un client (cf. fig. 3). Ainsi, sur une seule machine, il est possible de simuler les comportements des clients et du serveur. De plus, lors d’un test sur un réseau local, il n’y a qu’à donner un fichier .jar exécutable aux utilisateurs et lancer le serveur.

L’interface IServerRMI permet de gérer l’identification à l’aide d’une méthode login et de récupérer l’histogramme correspondant à une partie. ServerRMI implémente cette interface du côté serveur et RunRMI du côté client. Cette dernière classe permet au client de se connecter au réseau de clients existant déjà en passant par le Stub (cf. fig. 4).



*Fig. 3 : Fonctionnement global du RMI* [7]



*Fig. 4 : Diagramme de classes pour l’implémentation du RMI*

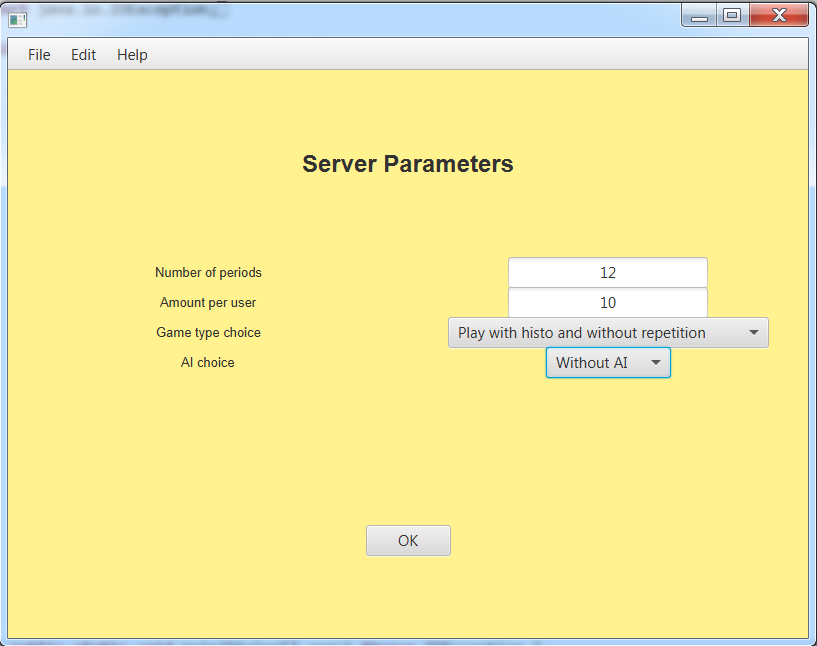
Le serveur doit définir les classes nécessaires pour la connexion et l’identification future des clients, ainsi que les structures correspondant aux différents types de jeu. Un utilisateur est défini par son adresse email qui servira de login ainsi qu’un mot de passe et une somme initiale d’argent avec laquelle il commence la première période. Chaque utilisateur qui arrive sur le jeu est à tour de rôle affecté à un partenaire. S’il arrive qu’il y ait un nombre impair d’utilisateur, on applique l’algorithme FIFO : le premier joueur arrivé laissera sa place au dernier arrivé dès qu’il choisit d’arrêter le jeu.

La classe Game définit les paramètres correspondant à un jeu. En plus des fonctionnalités incluses dans le cahier des charges standard de l’expérience, d’autres ont été rajoutées pour d’autres protocoles expérimentaux. Ainsi, il est possible de jouer avec une intelligence artificielle, ou bien encore de modifier le nombre de répétitions au sein d’une période, c’est-à-dire le nombre d’échanges que 2 joueurs peuvent effectuer au sein de la même période (dans l’expérience normale, il n’y a qu’une seule répétition par période).

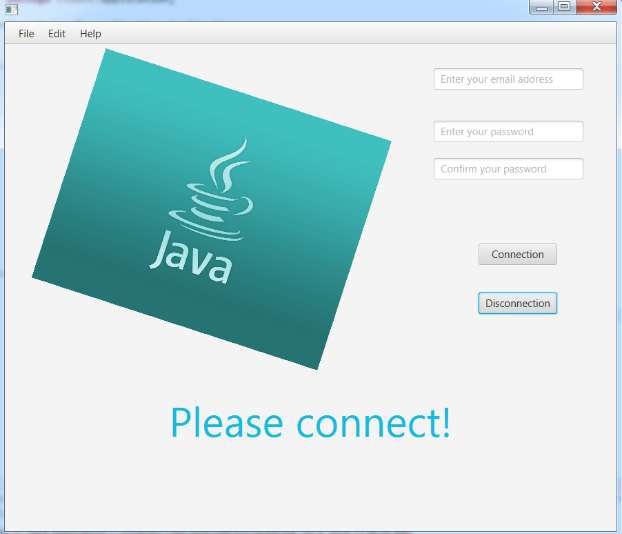
Au niveau du client, les données du jeu sont passées au serveur qui les sauvegarde dans un objet DataSample, contenant le montant joué par un client lors d’une période, le montant reçu par son partenaire, le temps qu’il met à jouer, le numéro de la période, et son partenaire. Cette classe, ainsi que la classe TreeSet de Java, sont utilisées pour construire l’histogramme.

Celui-ci ne sera pas toujours visible selon le type de jeu choisi. Dès qu’un jeu est terminé et si l’option du jeu le permet, un bouton apparait sur l’interface client principale permettant d’afficher l’histogramme des parties précédemment jouées.

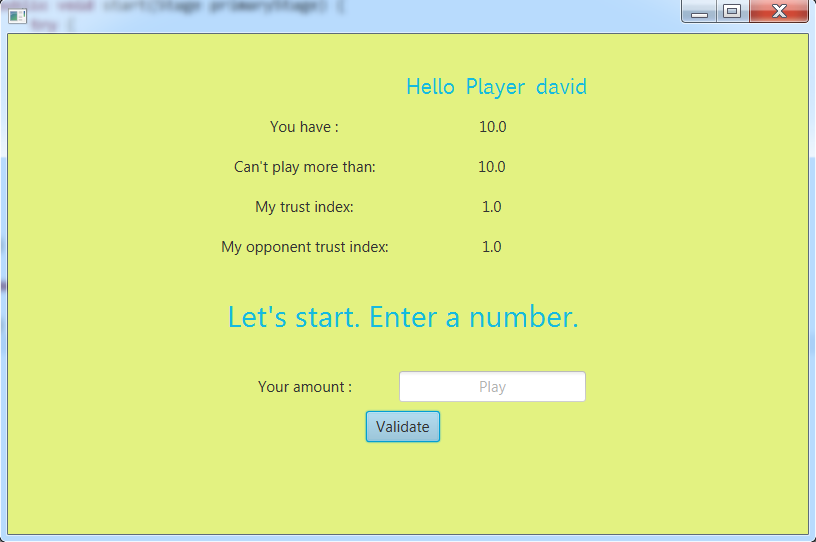
La librairie JavaFX a été utilisée pour créer les interfaces graphiques. Ces différentes interfaces graphiques sont représentées sur les figures 5 à 8.



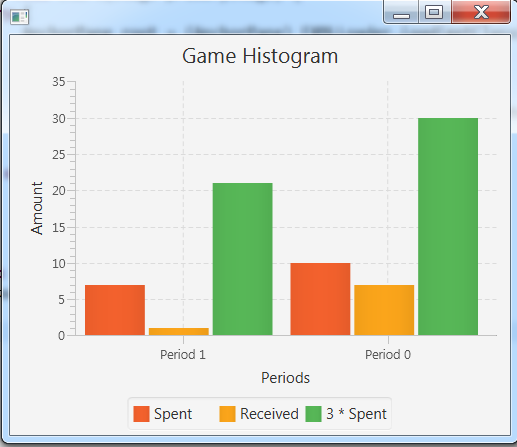
*Fig. 5 : Interface du serveur*



*Fig. 6 : Interface principale du client*



*Fig. 7 : Interface de jeu du client*

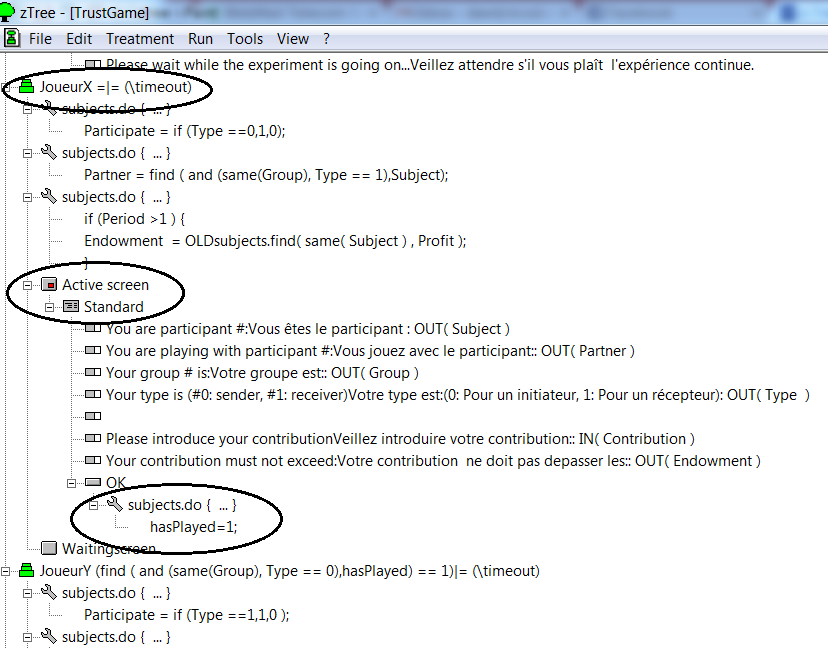


*Fig. 8 : Interface de l’histogramme*

L’objet DataSample sert également à la sauvegarde des données. Celles-ci sont enregistrées dans un fichier .csv permettant ensuite aux psychologues d’effectuer l’analyse sur le comportement des joueurs en fonction des différents paramètres de jeu (voir annexe 2 pour le code Java permettant cette sauvegarde).

### Implémentation en z-Tree

La deuxième implémentation utilise z-Tree [8], un logiciel développé par Urs Fischbacher de l’Université de Zurich pour réaliser des expériences dans le domaine économique. Il s’agit d’un logiciel fournissant une interface permettant de composer une expérience en plusieurs phases, appelés stages. Chaque stage se matérialise par une interface graphique sur laquelle on peut intégrer différents composants comme des boutons, du texte, des graphiques… (cf. fig. 9). Il est possible d’associer plusieurs programmes spécifiques à chaque stage. On peut également associer un programme global à l’ensemble du fichier développé. Le langage de programmation utilisé est spécifique à z-Tree.



Programme

Interface graphique

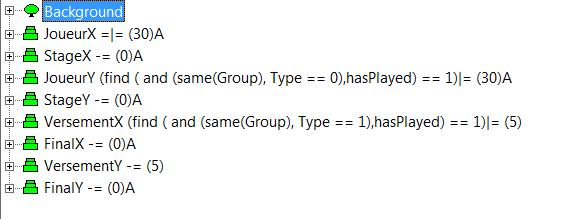
Stage

*Fig. 9 : Interface d’un fichier z-Tree*

L’une des difficultés majeures de z-Tree par rapport à Java est qu’il a fallu comprendre comment le logiciel fonctionnait et trouver les commandes nécessaires pour implémenter les différents jeux, le tout avec une documentation très restreinte. En effet, la communauté d’utilisateurs utilisant z-Tree est très faible dû à son utilisation spécifique à la recherche. Il a donc été beaucoup plus long de programmer avec z-Tree qu’avec Java.

Le principal problème était la sauvegarde des variables propres à chaque utilisateur afin de les réutiliser tout au long du jeu. Pour ce faire, z-Tree possède des outils intégrés pour la sauvegarde des variables appelées tables. Les plus utilisées sont les tables global, subject et session. Il est aussi possible de créer ses propres tables. Chaque table peut conserver les données soit pendant un stage, une session (correspondant à un jeu c’est-à-dire à un fichier de configuration z-Tree tel que celui-ci-dessus) ou bien un traitement (c’est-à-dire une expérience complète qui peut comporter plusieurs fichiers lancés à la suite). Dans ce cas-là, les données de la table sont sauvegardées jusqu’à ce qu’on ferme z-Tree. Lorsqu’on souhaite récupérer des données d’une table pour un traitement spécifique, il faut souvent réaliser des jointures de tables. Des exemples de code z-Tree sont donnés en annexe 3.

Pour implémenter les différents jeux, il a été nécessaire de créer différents fichiers qu’on lancerait à la suite lors de l’expérience. Au niveau des stages, on sait que le jeu se déroule en binôme et qu’il est nécessaire d’avoir 2 phases pour l’utilisateur : une première durant laquelle il inscrira le montant qu’il veut jouer, et une deuxième qui lui indiquera, après que son partenaire ait joué, son nouveau solde ainsi que la somme jouée par son partenaire. Suivant le cas, on affichera aussi l’indice de confiance. Il est donc nécessaire d’avoir 2 stages par utilisateur, soit 4 stages au total car les programmes associés à chacun des 2 joueurs sont différents. Cependant, lorsqu’il a été nécessaire de simuler les utilisateurs, on s’est rendu compte qu’il fallait rajouter des stages afin d’exécuter un programme pour que l’utilisateur simulé passe au stage suivant. Vu que les utilisateurs simulés joueront à la fois comme initiateur et récepteur, il a fallu donc rajouter 4 stages supplémentaires pour y exécuter du code propre à chaque utilisateur simulé (voir annexe 3). Ces 8 stages sont visibles sur la figure 10 et sont détaillés ci-dessous.



*Fig. 10 : Fichier comprenant les 8 stages de l’expérience*

JoueurX définit les actions de l’utilisateur qui commence le jeu. On lui associe le type 0 et un groupe. Le type et le groupe sont associés dès le début du jeu dans la table Parameter Table pour chaque joueur. Cette table est donnée en annexe 4. Lors de ce stage, l’utilisateur entre le montant qu’il souhaite envoyer à son partenaire. Dans le cas d’un utilisateur simulé, on exécute un programme qui calcule la somme à renvoyer en fonction de l’indice de confiance du partenaire, comme expliqué en 2.2.1.

Il est nécessaire d’avoir un fonctionnement asynchrone entre les différents stages pour les utilisateurs. En effet, par défaut lors d’une expérience, z-Tree va attendre que tous les utilisateurs ait finit un stage avant de lancer le suivant. Dans notre cas, cela voudrait dire qu’une fois que le joueur X d’un groupe a joué, son partenaire ne peut pas commencer tant que tous les joueurs X des autres groupes n’ont pas joué. Ce phénomène pourrait être très ennuyeux pour les utilisateurs et donc il est nécessaire d’adopter un comportement asynchrone. Pour cela, z-Tree laisse la possibilité d’ajouter une condition pour démarrer le stage. Il suffit de déclarer une variable hasPlayed propre à chaque utilisateur qu’on met à 1 lorsque le joueur a terminé le stage. Il est ainsi possible de démarrer le stage suivant en donnant comme condition que hasPlayed doit être égal à 1. Le code permettant de réaliser ce comportement est donné en annexe 3.

StageX est exécuté uniquement par les utilisateurs simulés lorsque ceux-ci sont initiateurs. L’intérêt de ce stage est d’exécuter un programme permettant de mettre la variable hasPlayed à 1 pour les utilisateurs simulés. En effet, dans le cas d’un utilisateur réel, cette variable est mise à 1 lorsque l’utilisateur appuie sur le bouton OK. Cependant, cela n’est pas possible dans le cas d’un utilisateur simulé. Il a donc été nécessaire d’implémenter un stage supplémentaire permettant de réaliser cette action.

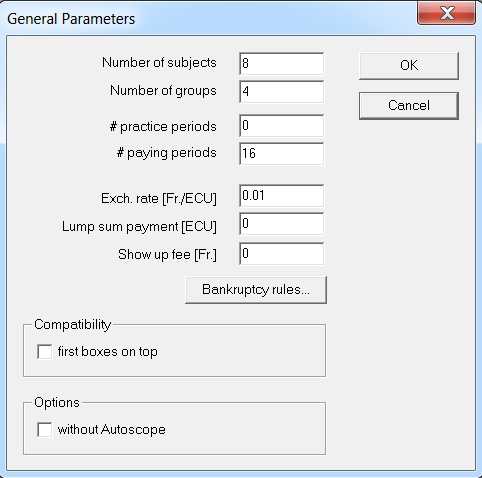
JoueurY définit les actions du partenaire de X, donc la deuxième personne à jouer. On lui associe le type 1. Selon le protocole de l’expérience, il reçoit le triple de la somme envoyée par X et ne peut pas renvoyer plus que ce montant-là. Pour les utilisateurs simulés, et selon le protocole de l’expérience, il y a renvoi de 10% pour les utilisateurs 1 et 3 et de 70% pour les utilisateurs 2 et 4. Lorsque le joueur est humain, celui-ci rentre la somme à envoyer à son partenaire et, en appuyant sur le bouton OK, la variable hasPlayed de ce joueur est mise à 1. Dans le cas d’utilisateurs simulés, la mise à jour de cette variable se fait dans un stage supplémentaire.

StageY est exécuté uniquement par les utilisateurs simulés et suit le même principe que StageX. Il permet de lancer le programme mettant la variable hasPlayed à 1, cela n’étant pas possible dans le stage précédant.

VersementX et VersementY sont lancés simultanément une fois que JoueurY est terminé. Il s’agit de phases récapitulatives qui permettent aux utilisateurs de connaitre leur nouveau solde. Chaque phase est quittée au bout de 5 secondes à l’aide d’un timer.

FinalX et FinalY sont lancés une fois que toutes les périodes sont terminées. Ils permettent d’afficher le solde total que l’utilisateur possède à la fin du jeu. Bien que chaque utilisateur commence le jeu suivant avec $10, l’argent gagné est conservé en mémoire afin de procéder au paiement à la fin de l’expérience.

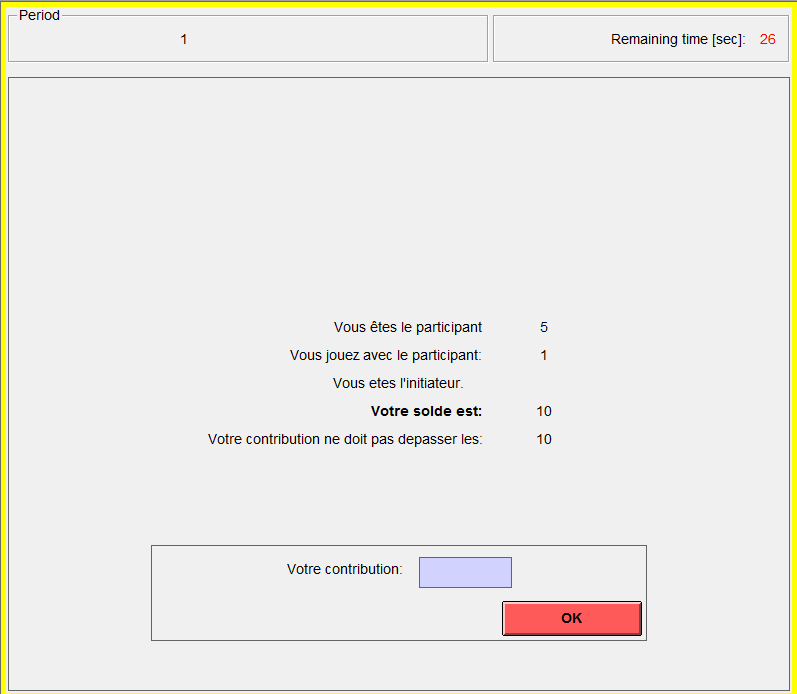
Une fois les programmes écrits, on peut les lancer dans z-Tree à condition qu’il y ait assez de clients connectés. Le nombre de clients, de groupes et de périodes est défini dans les paramètres globaux de chaque programme (cf. fig. 11).



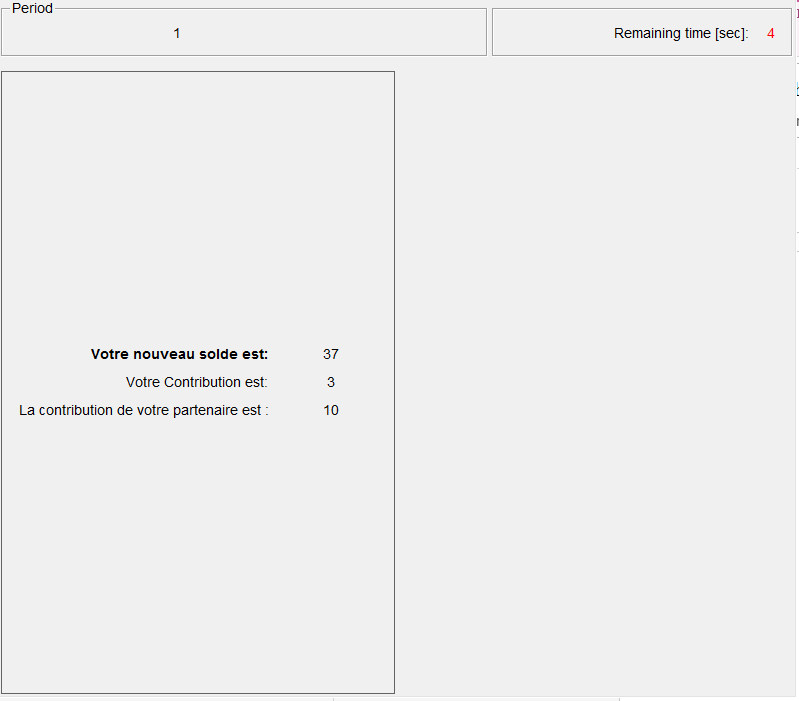
*Fig. 11 : Paramètres globaux d’un programme*

Pour lancer un client, on utilise le logiciel z-Leaf, qui n’est autre que la partie client de z-Tree. Chaque utilisateur exécute z-Leaf. Une fois que tous les utilisateurs ont ouvert z-Leaf, on lance le programme dans z-Tree sur le serveur et les utilisateurs peuvent effectuer l’expérience.

Au niveau graphique, les interfaces présentées par z-Tree sont très simples, et clairement moins évoluées que celles de Java. Cependant, vu qu’il ne s’agit que d’une expérience et que le logiciel ne sera pas publié plus tard, l’aspect design est donc moins important par rapport à l’aspect fonctionnel. La figure 12 présente l’interface qui s’affiche dès que le programme est lancé. Elle permet à l’utilisateur de rentrer le montant qu’il souhaite envoyer à son partenaire. La figure 13 s’affiche lorsque les 2 partenaires ont joués. C’est un récapitulatif de ce qui s’est passé dans la période. On y indique la somme d’argent envoyée au partenaire, celle reçue du partenaire, et le solde actuel pour l’utilisateur.

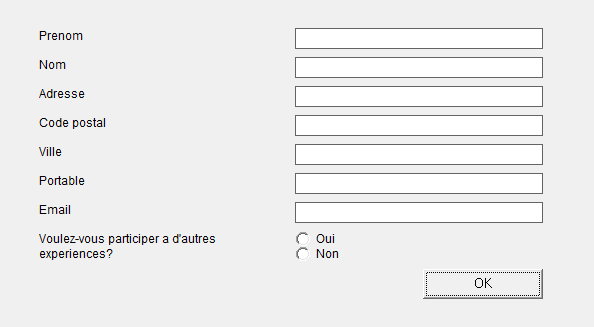


*Fig. 12 : première interface client*

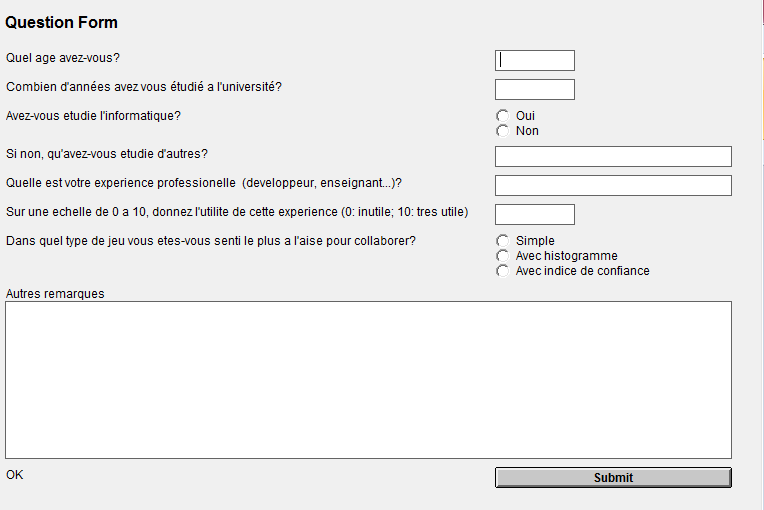


*Fig. 13 : deuxième interface client*

A la fin de l’expérience, il est possible, à l’aide de z-Tree, de lancer un questionnaire pour les utilisateurs. En effet, le logiciel possède des outils intégrés qui facilitent la création d’un tel questionnaire. Ainsi, un formulaire existe déjà pour recueillir les coordonnées des participants (cf. fig. 14). En plus de cela, une deuxième partie, spécifique à l’expérience, pose des questions sur les antécédents de l’utilisateur (expérience professionnelle) et sur le bon déroulement de l’expérience (cf. fig. 15). Ces données vont permettre ensuite aux psychologues de comprendre pourquoi les utilisateurs ont agi d’une certaine façon lors du jeu.



*Fig. 14 : Formulaire pour les coordonnées*



*Fig. 15 : Questionnaire concernant l’expérience*

### Déploiement réseau

Le déploiement réseau a posé pas mal de difficultés. En effet, il n’était pas possible de connaitre quel était le fonctionnement interne de z-Tree, et les documentations à disposition étaient très minces quant à la quantité d’information sur un tel déploiement. Il fallait tout d’abord créer un serveur qui lancerait z-Tree. Il a donc fallu demander au responsable réseau de générer une adresse IP publique.

L’adresse du server : **ztree-server.loria.fr**

L’adresse IP du server : **152.81.2.28**

Une fois le serveur crée, il fallait paramétrer un VPN pour que des clients distants puissent accéder au serveur depuis z-Tree. Les premières pistes étaient de passer par OpenVPN [9]. Il a fallu apprendre à se servir de ce logiciel et télécharger des fichiers de configuration pour que plusieurs ordinateurs soient sur le même VPN. Ces fichiers de configuration sont fournis sur le site officiel. Le problème venait d’une installation assez complexe, et donc dans le cas où les utilisateurs se trouveraient n’importe où sur l’Internet, il aurait fallu qu’ils configurent tout seul le VPN. Cela aurait pu les décourager de poursuivre l’expérience.

Une deuxième approche était de procéder à l’expérience en local, c’est-à-dire la réaliser dans les locaux d’Inria. Il était bien sûr toujours possible d’utiliser OpenVPN et de le configurer sur les ordinateurs utilisés pour l’expérience. Cependant, Inria possède un VPN personnalisé, Cisco AnyConnect VPN, auquel on peut accéder par l’intermédiaire de ses identifiants Inria. On a donc opté pour choisir ce VPN plutôt qu’OpenVPN. Le tutoriel étant sur l’intranet d’Inria, il était facile de le configurer sur chaque ordinateur utilisé pour l’expérience.

Pour lancer un client, il ne s’agit pas simplement de lancer z-Leaf car il faut passer en paramètre l’adresse IP du serveur. Il était donc nécessaire d’écrire un script pour lancer l’application. Un exemple était disponible sur le site de The University of Texas at Dallas. [10]

START zleaf.exe /language en /name David /server 152.81.2.28 /size 800x700 /fontsize 14

Ainsi, on fournit à chaque utilisateur un dossier dans lequel se trouvent l’exécutable z-Leaf et un fichier .bat. Il n’a donc plus qu’à lancer ce script pour commencer l’expérience.

### Solution choisie pour l’expérience

Après réflexion de quelle solution à utiliser pour réaliser l’expérience, il a été décidé d’utiliser l’implémentation en z-Tree. Plusieurs raisons ont amené cette décision. Tout d’abord, au niveau de l’expérience, tout ce qui est implémenté dans l’application Java l’est aussi avec z-Tree. Ensuite, une fois l’expérience réalisée, l’équipe de recherche publiera un article sur les résultats de l’expérience. Z-Tree étant un outil standard dans ce genre d’expériences et ayant déjà été utilisé dans le cadre de la recherche, il aurait fallu justifier pourquoi avoir implémenté une solution en Java alors que z-Tree fournit tous les outils nécessaires pour conduire l’expérience. Le fait que le logiciel Java soit plus maintenable, puisse évoluer et soit plus « user-friendly » ne sont pas des arguments suffisant dans le domaine de la recherche. Pour la crédibilité au niveau de la communauté scientifique, il est donc essentiel d’utiliser un outil qui a déjà fait ses preuves. Par exemple, plusieurs expériences sur le trust ont déjà été réalisé à l’aide de z-Tree, notamment une récemment par Andreas T. Lechner. [11]

## Résultats de l’expérience

Pour analyser les résultats de l’expérience, il est nécessaire de sauvegarder les données des utilisateurs dans des fichiers. Du coté de z-Tree, le logiciel génère automatiquement un fichier Excel contenant toutes les tables du programme au cours du jeu. La table subject est celle qui permet d’obtenir les informations pertinentes pour l’analyse. Les informations utilisées pour l’analyse sont le numéro de période, le numéro de groupe, le numéro du sujet, le type du sujet, le numéro du partenaire, la somme dont l’utilisateur dispose au début de chaque période, la contribution de l’utilisateur, la contribution du partenaire, le profit lors de la période, et le profit total du joueur (qui n’est autre que la somme des profits de chaque période). Les réponses des utilisateurs au questionnaire sont également sauvegardées dans des fichiers Excel.

Du coté de Java, on procède également à la sauvegarde des données sous la forme de fichiers .csv, comme spécifié en 2.3.1. La structure du logiciel étant différente par rapport à z-Tree, la méthode de sauvegarde n’est pas tout à fait la même. Ainsi, dans un premier fichier settings\_server.csv, on sauvegarde les paramètres du serveur (date de lancement du jeu, nombre de périodes, montant initial par utilisateur, type de jeu et choix de l’IA). Dans un second fichier settings\_client.csv, on sauvegarde les données du client récupérées à partir de l’objet DataSample, à savoir la date du jeu, le numéro de période, l’utilisateur commençant la partie, son adversaire, le montant envoyé, le montant reçu, le temps de jeu pendant la période, le nouveau solde de l’utilisateur, le nouveau solde de son adversaire et, le cas échéant, les indices de confiance de l’utilisateur courant et de son adversaire.

Ces fichiers de sauvegarde constituent la base quant à l’analyse des résultats conduite par les psychologues. Ils permettront de valider la métrique de confiance et serviront de preuve empirique lors de la publication des travaux de recherche.

Conclusion

Lors de ce stage à Inria, l’objectif était d’implémenter un logiciel qui permettrait de conduire une expérience sur des utilisateurs afin de mesurer leur aptitude à collaborer sur un même projet. La solution proposée permet ainsi de conduire l’expérience en tenant compte de différents paramètres, qui permettront, lors de l’analyse par les psychologues, de mieux cerner les motivations et aptitudes d’une personne à collaborer. A partir des résultats obtenus, il est donc possible de publier une étude sur l’influence d’un indice de confiance dans une communauté d’utilisateurs. Il reste de nombreuses possibilités à la suite de ce projet. On pourrait par exemple imaginer un algorithme permettant de prédire le comportement de l’utilisateur avant que celui-ci ne joue, ou bien encore tenter d’améliorer la convergence de l’algorithme de confiance.

Personnellement, ce stage au sein d’Inria fut très enrichissant puisqu’il m’a permis de découvrir le monde de la recherche. Ayant déjà eu l’occasion de travailler en entreprise, j’ai pu me rendre compte de la différence entre le pôle de la recherche et le pôle de l’industrie. J’ai été très souvent amené à revenir sur mon travail et à implémenter de nouveau des solutions légèrement différentes des précédentes. J’ai aussi eu l’occasion de rechercher des éléments de solution dans des articles de recherche internationaux, ce qui a développé ma curiosité et mon esprit de synthèse. Je peux maintenant mieux cibler dans quel type de structure je souhaiterais effectuer mon stage en 3e année et travailler plus tard.

# Bibliographie

|  |  |
| --- | --- |
| [1] | Inria, «Annual Report 2012,» 2012. |
| [2] | Inria, «Organigramme général d'Inria,» [En ligne]. Available: http://www.inria.fr/institut/organisation/organigramme-general. [Accès le 7 Juillet 2014]. |
| [3] | LORIA, «Le LORIA,» [En ligne]. Available: http://www.loria.fr/le-loria-1. [Accès le 31 Juillet 2014]. |
| [4] | O. Brenner, «SCORE,» 19 Janvier 2012. [En ligne]. Available: http://www.loria.fr/la-recherche/equipes/score. [Accès le 31 Juillet 2014]. |
| [5] | J. BERG, «Trust, Reciprocity, and Social History,» chez *Games and Economic Behavior 10, 122-142 (1995)*, Iowa City, 1994. |
| [6] | A. Das et M. M. Islam, «SecuredTrust: A Dynamic Trust Computation Model for Secured Communication in Multiagent Systems,» chez *IEEE Transactions On Dependable And Secure Computing*, 2012. |
| [7] | A. Defrance, «Le fonctionnement de RMI (Remote Method Invocation): Developpez.com,» 15 Avril 2010. [En ligne]. Available: http://alain-defrance.developpez.com/articles/Java/J2SE/micro-rmi/. [Accès le 9 Juillet 2014]. |
| [8] | U. Fischbacher, «z-Tree: Zurich toolbox for ready-made economic experiments,» chez *Economic Science Association*, Zurich, 2006. |
| [9] | OpenVPN Technologies, Inc., «OpenVPN,» [En ligne]. Available: https://openvpn.net/. [Accès le 11 Juillet 2014]. |
| [10] | E. E. Haruvy, «The University of Texas at Dallas,» [En ligne]. Available: www.utdallas.edu/~eeh017200/tree.ppt. [Accès le 10 Juillet 2014]. |
| [11] | A. T. Lechner, «Religion and Trust in Strangers: Evidence from a Trust Game with Third Party Punishment,» Passau, 2013. |

Annexes

1. Détermination de l’algorithme du Trust

On se fixe 3 points :

* (0,0) : si je renvoie 0, mon indice de confiance sera 0
* (1,0.5) : si je renvoie une fois ce mon partenaire m’a envoyé, mon indice de confiance sera 0.5 (je mérite la moitié de la confiance car j’ai renvoyé l’argent que mon partenaire a investi).
* (3,1) : si je renvoie 3 fois le montant que mon partenaire m’a envoyé, mon indice de confiance sera au maximum (je prouve que je suis digne de confiance en donnant tout ce que je peux donner)

A partir de ces points, on peut déduire les équations des 2 droites.

Sur [0, endowment]

Soit x1, x2 deux points tels que x1 [0,1] et x2 [0,1]

D’après les propriétés des droites linéaires, on a :

Sur [endowment, 3\*endowment]

Soit x1, x2 deux points tels que x1, x2 [endowment, 3\*endowment]

D’après les propriétés des droites affines, on a :

# Code Java

Le code Java servant à la sauvegarde des données est présenté ici. Ce code-ci est implémenté dans une classe spécifique FileSaver, que l’on appelle à partir des objets DataSample. Comme les méthodes sont très proches au niveau du client et du serveur (seules les données sauvegardées changent), j’ai décidé de n’inclure ici que les méthodes correspondant à la sauvegarde des données du serveur.

Code pour générer le fichier csv du serveur

**public** **void** generateCsvServer() {

**try** {

*settingsServer*.createNewFile();

FileWriter writer = **new** FileWriter(*settingsServer*.getName(), **true**);

writer.append("Date");

writer.append(',');

writer.append("Number of periods");

writer.append(',');

writer.append("Amount per user");

writer.append(',');

writer.append("Game type choice");

writer.append(',');

writer.append("AI choice");

writer.append('\n');

writer.flush();

writer.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

Code pour écrire dans le fichier csv du serveur

**public** **void** writeCsvServer() {

**try** {

FileWriter writer = **new** FileWriter(*settingsServer*.getName(), **true**);

DateFormat dateFormat = **new** SimpleDateFormat("MM/dd/yyyy hh:mm a");

Date date = **new** Date();

writer.append(dateFormat.format(date));

writer.append(',');

writer.append(**this**.game.getAllowedStrokes() + "");

writer.append(',');

writer.append(**this**.game.getAmountPerUser() + "");

writer.append(',');

writer.append(**this**.game.getGameChoice() + "");

writer.append(',');

writer.append(**this**.game.getTypeAI().getModelAI());

writer.append('\n');

writer.flush();

writer.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

1. Code z-Tree

J’ai choisi d’inclure ici le code z-Tree le plus pertinent. Le code est commun aux 2 programmes réalisés. La particularité du deuxième programme est qu’il permet d’afficher l’indice de confiance, mais cet indice de confiance est déjà calculé dans le premier programme afin de déterminer la somme que les utilisateurs simulés enverront à leur partenaire.

Code pour l’assignation du type de joueur (0 : émetteur ; 1 : récepteur)

num\_subjects = subjects.maximum (Subject);

i=1;

repeat {

i=i+1;

subjects.do {

rand = random();

}

subjects.do {

rank = subjects.count (rand<=:rand);

}

} while (subjects.sum (Subject) != subjects.sum (rank) & i<10);

subjects.do {

Type = if (rank <= \num\_subjects/2,0,1);

}

Code du joueur X

Participate = if (Type ==0,1,0);

Partner = find (and (same(Group), Type == 1),Subject);

if (Period >1 ) {

Endowment = OLDsubjects.find (same( Subject ) , Profit );

}

Code du joueur Y

Start if find (and (same(Group), Type == 0),hasPlayed) == 1

Participate = if (Type ==1,1,0 );

Partner = find (and (same(Group), Type == 0), Subject);

PartnerDecision = find (and (same(Group), Type == 0), ContributionX);

if (Period >1 )

Endowment = OLDsubjects.find (same( Subject ) , Profit );

Code du versement de X

Start if find (and (same(Group), Type == 1),hasPlayed) == 1

Participate = if (Type == 0,1,0);

PartnerDecision = find (and (Group == :Group, Type == 1), ContributionY);

Partner = find (and (same(Group), Type == 1), Subject);

Profit = Endowment - ContributionX+ PartnerDecision;

Decision = find (and (Group == :Group, Type == 0), ContributionX);

Code du versement de Y

Participate = if (Type == 1,1,0);

PartnerDecision = find (and (Group == :Group, Type == 0), ContributionX);

MyContribution = find (and (Group == :Group, Type == 1), ContributionY);

Profit = Endowment - MyContribution+ 3\*PartnerDecision;

Code de l’algorithme du Trust pour le joueur X

if (Period ==1 ) {

TrustN=1;

} else {

TrustN = OLDsubjects.find (same( Subject ) , TrustN);

}

Code de l’algorithme du Trust pour le joueur Y

if (MyContribution< PartnerDecision) {

TrustN= 0.5\*MyContribution/PartnerDecision;

} else {

TrustN= 0.25\*(MyContribution/PartnerDecision)+0.25;

}

if (Period>1) {

TrustA = OLDsubjects.find (same( Subject ) , TrustN);

TrustN= (0.5\*TrustN)+(0.5\*TrustA);

}

Code pour un utilisateur simulé initiateur

if (TrustN < 0.25) {

ContributionX = 1;

}

if (TrustN < 0.5 & TrustN >= 0.25) {

ContributionX = 2;

}

if (TrustN < 0.75 & TrustN >= 0.5) {

ContributionX = 3;

}

if (TrustN < 1 & TrustN >= 0.75) {

ContributionX = 4;

}

if (TrustN == 1) {

ContributionX = 5;

}

Code pour un utilisateur simulé récepteur

if (Subject==1 | Subject==3 ) & Type == 1

ContributionY = rounddown(0.1\*PartnerDecision\*3,1);

if (Subject ==2 | Subject ==4) & Type == 1

ContributionY = rounddown(0.7\*PartnerDecision\*3,1);

1. Parameter Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Subject  Period | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
| P1 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G1  T0 | G2  T0 | G3  T0 | G4  T0 |
| P2 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G2  T0 | G1  T0 | G4  T0 | G3  T0 |
| P3 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G1  T1 | G4  T1 | G3  T1 | G2  T1 |
| P4 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G3  T0 | G2  T0 | G1  T0 | G4  T0 |
| P5 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G2  T1 | G1  T1 | G4  T1 | G3  T1 |
| P6 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G3  T1 | G2  T1 | G1  T1 | G4  T1 |
| P7 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G4  T1 | G3  T1 | G2  T1 | G1  T1 |
| P8 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G4  T0 | G3  T0 | G2  T0 | G1  T0 |
| P9 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G1  T0 | G2  T0 | G3  T0 | G4  T0 |
| P10 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G2  T0 | G1  T0 | G4  T0 | G3  T0 |
| P11 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G1  T1 | G4  T1 | G3  T1 | G2  T1 |
| P12 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G3  T0 | G2  T0 | G1  T0 | G4  T0 |
| P13 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G2  T1 | G1  T1 | G4  T1 | G3  T1 |
| P14 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G3  T1 | G2  T1 | G1  T1 | G4  T1 |
| P15 | G1  T0 | G2  T0 | G3  T0 | G4  T0 | G4  T1 | G3  T1 | G2  T1 | G1  T1 |
| P16 | G1  T1 | G2  T1 | G3  T1 | G4  T1 | G4  T0 | G3  T0 | G2  T0 | G1  T0 |

**Explications du tableau :**

G1 = Groupe 1 (idem pour G2, G3, et G4)

T0 si initiateur

T1 si récepteur

Les utilisateurs simules sont S1, S2, S3, S4

Les utilisateurs réels sont S5, S6, S7, S8