UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MOTIVATIONS DES PROGRAMMEURS ET SIGNALISATION À L'INDUSTRIE

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN ÉCONOMIQUE

PAR

DOMINIQUE DUCHESNEAU

FÉVRIER 2011

REMERCIEMENTS

Un grand merci au département des sciences économiques de l'UQAM, principalement à mon directeur de recherche Claude-Denys Fluet qui m'a initié aux modèles d'équilibres bayesiens dans leur vraie nature. Merci également à Stéphane Pallage, Max Blouin et Nicolas Marceau pour m'avoir permis l'étude de la théorie des jeux en général. Un merci tout spécial à Fabian Bastin et Mathieu Lemoine, du Département d'informatique et de recherche opérationnelle de l'Université de Montréal, pour leurs apports sur les plans informatique et juridique des licences libres. Enfin, un merci tout particulier à ma conjointe pour son soutien indéfectible.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Ce mémoire présente une nouvelle approche à l'étude économique des logiciels libres. Les logiciels libres, contrairement aux logiciels commerciaux, sont gratuits et peuvent être étudiés, modifiés et redistribués. Les programmeurs élaborent et distribuent ces logiciels gratuitement. La littérature actuelle suggère deux raisons principales à cela : le plaisir et un incitatif de carrière. Pour protéger leurs œuvres, les programmeurs font appel aux licences libres qui, comme les licences commerciales, définissent les usages et droits accordés aux utilisateurs. Deux licences sont majoritairement employés : la GPL et la BSD. Chacune, de par ses clauses, permet une visibilité différente des programmeurs de la communauté Open Source. Ainsi, elles sont ici employées dans un modèle de choix optimal des programmeurs afin de déterminer si les différentes motivations des programmeurs devraient être prises en compte lors du choix de la licence. Les principaux résultats suggèrent que (1) tenir compte des motivations des programmeurs est une source d'utilité importante pour la population et (2) les licences semblent actuellement utilisées à l'inverse de ce qu'elles devraient être.

INTRODUCTION

> *It is not from the benevolence of the butcher, the brewer, or the baker, that we expect our dinner, but from their regard to their own self-interest. We address ourselves, not to their humanity but to their self-love, and never talk to them of our own necessities but of their advantages.*
>
> **Adam Smith**

Près de 240 ans après sa publication originale, cette citation, tirée de *(...) the Wealth of Nations*, est toujours vraie. La théorie microéconomique contemporaine nous enseigne que l'*homo œconomicus* est rationnel, donc qu'il agit dans son propre intérêt. Plus est meilleur que moins, et à moins que ce ne soit profitable pour soi-même, l'altruisme n'est pas économique. Ou plutôt, si le gain personnel qu'un individu tire de l'altruisme n'est pas supérieur au coût engendré par l'effort d'aider les autres, l'individu néanmoins altruiste n'est pas rationnel. Ceci laisse donc entendre que les motivations poussant les individus à entreprendre une action peuvent être de nature différente.

Par exemple, prenons un adolescent, vers la fin du secondaire, devant choisir une carrière. Un éventail de choix s'offre à lui, qui peuvent être ramenés à deux grandes catégories : devra-t-il choisir une carrière lui assurant un bon revenu ou plutôt une carrière qui lui permettra de retirer du « plaisir » ? Comment concilier les deux options ? Dans une autre situation, un parent doit-il travailler de longues heures afin d'être à l'aise financièrement ou travailler moins afin de passer plus de temps avec sa famille malgré un revenu moindre ? Si par contre ce parent est une infirmière, étant donné les conditions de travail actuelles dans les hôpitaux, quel bien-être doit passer en premier, le sien ou celui de ses patients ? Dans un autre ordre d'idées, prenons un travailleur d'usine confronté à un vote de grève, ou un patron d'usine devant une décision de lock-out. Comment chacun décidera ? Est-ce que l'arrêt de travail, avec ses conséquences

financières, surviendra (avec peut-être une amélioration à la clé), ou est-ce que chacun décidera que les conditions actuelles sont suffisantes, et qu'ils ne peuvent se permettre un arrêt de travail ? Ou encore, est-ce qu'un patron doit engager le candidat le plus diplômé ou le plus passionné ?

Un simple hobby, qui peut s'avérer dispendieux, s'explique par la fierté qu'en retire l'individu qui le pratique ou le plaisir qu'il en retire. Les jeux d'argent, c'est bien connu, proposent des gros lots énormes, mais avec une probabilité de gagner si faible qu'en réalité, il faut s'attendre à perdre de l'argent plutôt qu'à en gagner. Il y a là de la « désutilité » sociale, mais une utilité privée. Or, la science économique explique que ces deux phénomènes sont rationnels néanmoins ; que ce soit du point de vue de l'industrie qui offre ce service ou du consommateur qui juge une option meilleure que les autres, chacun y trouve son profit.

Mais qu'est-ce qui sous-tend la rationalité ? En d'autres mots, quelles sont ces motivations qui poussent un agent à adopter un certain comportement, tout égoïste soit-il ? La psychologie regroupe les incitatifs en groupes. On parlera par exemple de renforcement positif ou négatif, selon que l'on récompense le succès ou que l'on punisse l'échec. Il y a en réalité des dizaines de catégories de motivations. Ce mémoire traitera des motivations derrière un travail s'apparentant au bénévolat : les motivations intrinsèques et les motivation extrinsèques d'un programmeur au sein de la communauté Open Source.

Les logiciels *libres* (ou *Open Source*) se distinguent des logiciels commerciaux à deux égards. Premièrement, les logiciels libres sont généralement gratuits, ou si peu coûteux que seulement les coûts de production et de distribution sont couverts. Ensuite, ils ne sont généralement pas développés dans des entreprises, mais plutôt par une communauté hétéroclite de programmeurs dispersés aux quatre coins de la planète, travaillant ensemble de manière virtuelle, par Internet. Ces programmeurs ne sont pas rémunérés pour leur travail, contrairement aux développeurs travaillant pour des entreprises commerciales. Pourtant, les logiciels que la communauté Open Source produit sont souvent de meilleure qualité que les logiciels commerciaux, bien que moins répandus.

La communauté Open Source a recours à des licences, tout comme les entreprises commerciales, afin de protéger les droits des différents auteurs des logiciels. Ces licences octroient des privilèges aux utilisateurs de logiciels, contrairement aux licences commerciales, qui restreignent les usages possibles de ces logiciels. Ces licences peuvent être regroupées en deux « familles ». On parlera des licences au copyleft fort ou au copyleft faible. Le choix d'une licence est effectué souvent pour une raison pratique : une licence populaire, bien connue, a plus de chances d'être choisie. Mais devrait-elle l'être ? De plus, les deux organismes principaux qui s'occupent de ces licences sont divisés essentiellement par leurs positions « idéologiques ». L'un prône la liberté à tout prix, l'autre vise plutôt la conciliation des logiciels commerciaux et libres. Est-ce que ces deux positions ne pourraient-elles pas être utilisés pour orienter un programmeur joignant la communauté Open Source vers un projet mieux adapté à son talent propre ?

Donc, cette « industrie » des logiciels libres comporte des millions d'individus qui travaillent volontairement sans être rémunéré. Qui plus est, ils donnent leur production gratuitement et assument parfois eux-mêmes les coûts de distribution de leur produit. Aussi, ils incluent également une permission de le démonter afin de l'étudier, de le modifier et de le redistribuer à la simple condition que les auteurs originaux soient mentionnés dans le produit modifié.

Il est permis de se demander ce qui pousse ces programmeurs à s'organiser en communauté afin de défendre leur mode de production. Plusieurs de ces programmeurs affirment agir ainsi pour permettre à chacun l'accès à des logiciels de qualité. D'autres diront plutôt qu'ils aiment programmer et qu'ils voient là une bonne occasion de pratiquer un hobby. Certains programmeurs de la communauté Open Source sont devenus célèbres et ont même réussi à amasser une certaine fortune grâce aux logiciels libres, le premier d'entre eux étant sans doute Linus Torvald, créateur de Linux. Quelles que soient leurs motivations, ces programmeurs font appel à l'une des quelques dizaines de licences libres qui existent afin de protéger leurs droits et leurs programmes. La licence est choisie en fonction de divers critères, tels la philosophie sous-jacente à la

licence, la propagation virale de la licence, la permission de réutiliser des bibliothèques de programmes, etc.

Ce mémoire présente dans le premier chapitre un bref historique de la communauté Open Source et une revue de la littérature économique sur ce sujet. Les concepts utilisés et mentionnés jusqu'à présent y seront aussi définis. Dans le second chapitre, le modèle qui y est développé tentera de reproduire le comportement de la communauté Open Source actuelle, pour laquelle tous les projets sont sur un pied d'égalité. Pour ce faire, il n'y aura qu'une seule licence, la GPL, qui, par sa composition, permet un fort rayonnement des programmeurs ayant participé à un projet. Les programmeurs sont hétérogènes, différenciés par leur talent propre. Au chapitre suivant, une seconde licence sera incorporée au modèle, la licence BSD. Celle-ci est telle que tout code produit peut être utilisé par un logiciel commercial, à la condition que la licence et les crédits soit inclus dans ce logiciel ou dans la documentation du logiciel commercial. Cette licence est donc adaptée pour un programmeur qui ne recherche ni la gloire ni la célébrité. Le quatrième chapitre présentera quelques situations plausibles afin de déterminer lequel des deux paradigmes est préférable. Il est attendu, à ce stade, qu'un paradigme où les motivations personnelles de chacun sont respectées est bénéfique pour le bien-être de la société.

CHAPITRE I

DES LOGICIELS LIBRES À L'ÉCONOMIQUE

L'informatique telle qu'on la perçoit maintenant a commencé à se développer dans les années soixante. C'est à cette époque que les premiers ordinateurs se sont répandus et que l'informatique est devenue une science à proprement parler. Chacune de ces machines était indépendante des autres, c'est-à-dire que les programmes étaient spécifiques à un seul modèle d'ordinateur, et les modèles étaient généralement spécifiques à un seul laboratoire. En fait, les ordinateurs étaient livrés avec leurs systèmes d'exploitation et leurs logiciels propres. Durant une vingtaine d'années, le développement s'est fait essentiellement dans de grands laboratoires institutionnels tels que ceux d'universités américaines et de grandes entreprises, notamment les laboratoires du MIT (CSAIL), de Berkeley (département du génie électrique), de Bell (alors Bell-AT&T), d'IBM et de Xerox. Comme la science informatique était encore très jeune, de nombreux problèmes se présentaient et les chercheurs des différents laboratoires n'hésitaient pas à se consulter pour les résoudre. Parce que les différentes machines n'étaient généralement pas compatibles entre elles, la résolution des problèmes d'un laboratoire n'avait que peu d'influence sur la recherche des autres laboratoires, et la consultation interlaboratoire a fait progresser le domaine à pas de géants.

Dans les années soixante-dix, il a été jugé anti-concurrentiel de fournir gratuitement les logiciels avec les ordinateurs. Pour régulariser la situation, divers langages de programmation dits « portables » (c'est-à-dire qui pouvaient servir sur plusieurs types de machines) furent développés afin d'élaborer des systèmes d'exploitation

pouvant être utilisés sur tout type de machines. Ces derniers étaient toujours livrés avec leurs sources puisque sinon les programmeurs ne savaient pas à quoi exactement ils avaient affaire. Avec la propagation des micro-ordinateurs dans les années quatre-vingt est apparue la commercialisation des logiciels. La possibilité de gains pécuniaires a entraîné un comportement jugé répréhensible chez un nombre sans cesse grandissant de programmeurs : ceux-ci récupéraient des sections de code (lorsque ce n'était pas des logiciels entiers), les modifiaient minimalement et se les appropriaient (à l'aide d'une licence ou de droits exclusifs sur le code source du logiciel) dans le but de les commercialiser. Ce faisant, des programmes initialement disponibles pour tous, sans contraintes, devenaient commerciaux sans que les auteurs originaux n'en retirent le moindre bénéfice ni aient le moindre mot à dire à ce sujet. En 1985, Richard M.

Tableau 1.1 : Libertés garanties par la Free Software Foundation

Liberté d'usage :	Chacun peut utiliser tout logiciel comme bon lui semble, qu'il soit un particulier, une entreprise ou une institution.
Liberté d'étude :	Chacun doit pouvoir étudier le fonctionnement d'un logiciel.
Liberté de redistribution :	Chacun peut redistribuer tout logiciel à qui le désire.
Liberté de modification :	Chacun peut modifier et améliorer un logiciel et rendre ses modifications publiques dans le but d'aider la communauté.

Source : The Free Software Foundation

Stallman a décidé de s'opposer à ces pratiques en fondant la Free Software Foundation (FSF), qui a pour mandat de promouvoir le libre accès aux logiciels. Il faut ici interpréter *free* comme « libre », et non comme « gratuit ». Pour ce faire, il a créé le concept de « logiciel libre » (*Free Software*), auquel est attachée une « licence libre ». Intitulée *General Public Licence* (GPL), cette licence définit un logiciel libre comme répondant à quatre libertés essentielles (voir le tableau 1.1), par opposition aux licences commerciales qui interdisent à un utilisateur de logiciel d'employer celui-ci à toute autre fin que celle prescrite par l'éditeur du logiciel. De plus, une licence commerciale interdit notamment l'accès au code source, afin de préserver les « secrets industriels » de l'entreprise. On

ne parle pas de brevets, quoique le sujet soit débattu de nos jours, particulièrement en Europe.

La FSF a créé le système d'exploitation GNU's not Unix-Linux, qui est gratuit et distribué selon les termes de la GPL, afin de mettre son idéologie en pratique. De nos jours, il s'agit d'un concurrent important aux principaux systèmes d'exploitation commerciaux. Selon la W3C school, 88,6 % des micro-ordinateurs utilisent l'une ou l'autre version de Microsoft Windows®, 6,5 % utilisent Mac OS® et 4,5 % utilisent GNU-LINUX®. La GPL est cependant soumise à de nombreuses critiques. Le ton moralisateur de la licence y est pour quelque chose, le but premier de la FSF étant d'empêcher la « privatisation » des logiciels. Le principal point de désaccord sur la GPL est son aspect « contaminant », ou « caractère viral » : un projet qui souhaite réutiliser une partie de code assujeti à la GPL doit également l'être, et ainsi de suite. C'est pour freiner cet aspect qu'un groupe de programmeurs a fondé, en 1998, l'Open Source Initiative (OSI). Le but de ce groupe était de redéfinir le logiciel dit libre en abandonnant les positions idéologiques de la FSF ; l'OSI vise plutôt la promotion des logiciels libres en tant que produits de meilleure qualité. Dans ce but, ils ont créé la définition d'un « logiciel Open Source », qui doit satisfaire à 10 critères (voir le tableau 1.2).

Tableau 1.2 : Critères composant la définition de « logiciel Open Source »

Redistribution gratuite :	Le logiciel doit pouvoir être redistribué à volonté.
Code Source :	Le code source d'un logiciel doit être gratuit et inclus dans la distribution ou facilement disponible.
Projet dérivés :	Le logiciel doit pouvoir être modifié et la version modifiée peut être redistribuée.
Intégrité de l'original :	Le code source original doit demeurer intact, les modifications étant effectuées sous forme de rustines (*patches*).

Absence de discrimination individuelle :	Personne ni aucun groupe ne peut se voir refuser l'utilisation et/ou la modification du logiciel.
Absence de restriction d'usage :	Aucun champ d'application ne peut être exclu, et la commercialisation doit être permise.
Distribution de la licence :	Les droits accordés par la licence rattachée à un logiciel doivent s'appliquer à tous, sans que personne n'ait à exécuter les termes d'une autre licence.
Non-spécificité :	Une licence ne doit pas être spécifique à un seul produit.
Non-restriction logicielle :	Une licence doit permettre la récupération du code source dans un cadre autre que l'Open Source.
Neutralité envers la technologie :	La licence ne doit pas requérir de matériel spécifique pour être exécutée.

Source : The Open Source Initiative

La principale différence entre les deux organisations est essentiellement la philosophie de leurs fondateurs : pratiquement tous les logiciels qui satisfont aux critères de l'OSI pour être Open Source garantissent par le fait même les libertés du logiciel libre de la FSF. Autrement dit, la compétition entre la FSF et l'OSI n'est pas sur le plan des logiciels en tant que tels, puisque la définition de logiciel Open Source est assez large pour englober les libertés garanties par la GPL, et vice-versa.

La FSF, bien qu'elle n'interdise pas le commerce des logiciels, tente de réduire leurs prix au minimum. Le prix de revente devrait se limiter aux coûts de distribution, puisque la reproduction d'un logiciel a un coût nul. L'OSI tente plutôt d'établir un cadre pragmatique avec un esprit d'entreprise, afin d'inciter des entreprises telles que Netscape à « libérer » le code de leurs logiciels, comme ce fut le cas avec le navigateur Mozilla, qui est devenu Firefox. Bien que de telles actions soient aujourd'hui courantes, il s'agissait d'une première sans précédent à l'époque. Cependant, l'élaboration de

la définition de logiciel Open Source n'a pas été sans remous, et tous n'étaient pas d'accord. Par exemple, la licence libre utilisée par Netscape est un hybride entre une licence commerciale et la GPL. Il existe en fait de nos jours des dizaines de licences libres correspondant aux différentes caractéristiques désirées par les gestionnaires de projets. Ces licences sont souvent sources de conflits dans la communauté Open Source puisque de nombreux programmeurs ne participeront que si le projet utilise une licence en particulier.

1.1 Qu'est-ce qu'un logiciel ?

Plusieurs termes ont été mentionnés jusqu'à présent; afin d'éviter toute confusion, une explication de ce qui est entendu par *logiciel* est nécéssaire. Un logiciel passe par deux étapes : les programmeurs écrivent le code source, puis le compilent afin de produire le code objet (RAYMOND, 2001a). Les fichiers résultants sont ce que la machine peut interpréter. C'est pourquoi on dira que la compilation « traduit » du langage de haut niveau (compréhensible par le programmeur) vers du langage de bas niveau (directement interprétable par la machine). Il existe des centaines de langages de programmation différents et, en règle générale, un programmeur est compétent dans un très petit nombre de langages.

C'est le code source d'un logiciel qu'un programmeur protège à l'aide de droits d'auteur (LERNER et TIROLE, 2002). Dans le cas du logiciel commercial, en plus d'être protégé ainsi, le code source n'est pas distribué. Les logiciels libres sont, par définition, accompagnés du code source, de sorte que tous puissent avoir accès à ce qui a été écrit par les programmeurs. En fait, les libertés garanties par la FSF, comme les critères de la définition d'Open Source, requièrent l'accès au code source d'un logiciel. De plus, un *fichier journal* (*log file*) accompagne en général toute distribution d'un logiciel libre. Ce fichier contient normalement deux choses : la liste des modifications effectuées depuis la version précédente et la liste des contributions individuelles. Cette liste est un

élément important de la distribution d'un logiciel libre, puisqu'elle permet de protéger les programmeurs individuellement en plus du code source.

Les logiciels libres sont souvent modulaires : ils sont découpés en une grande quantité de *modules*, qui doivent être assemblés pour faire des différentes parties un logiciel à part entière, mais qui peuvent aussi être utilisés par d'autres logiciels. Il y en a plusieurs types ; sans entrer dans les détails techniques, on peut les regrouper en trois catégories :

Bibliothèques : Ce sont les modules de base. Ils contiennent des routines et sous-routines qui peuvent être réutilisées dans plusieurs logiciels. Les bibliothèques ont pour but de ne pas avoir à réinventer la roue dans chaque logiciel qui est développé. Par exemple, les commandes requises pour faire afficher une fenêtre, ou encore pour lancer l'impression d'un document, font partie des bibliothèques.

Modules logiciels : Ces modules sont les éléments qui permettent les fonctions d'un logiciel en particulier. Par exemple, dans le cas d'un logiciel de calcul, les algorithmes de résolution d'équations font partie des modules logiciels. C'est dans ces modules que les programmeurs peuvent se démarquer en résolvant des problèmes ardus.

Modules facultatifs : Comme leur nom l'indique, ces modules sont généralement des ajouts supplémentaires au logiciel. Ils ne sont pas obligatoires, et sont généralement destinés à un usage précis. On peut par exemple penser aux différentes boîtes à outils du logiciel MATLAB®.

La modularité n'est pas, à proprement parler, une caractéristique propre aux logiciels libres. Cependant, les logiciels libres sont réputés pour leur grande modularité et l'efficacité qui en découle. Il faut cependant noter que la modularité a aussi ses limites : trop de modules risque de causer des failles de sécurité lors de l'exécution du logiciel. Une trop grande quantité de modules peut faire en sorte que les communications entre les diverses parties du logiciel deviennent confuses, créant ainsi des portes d'entrées pour les pirates informatiques (FITZGERALD, 2005).

Le mode de production propre à la communauté Open Source entraîne un gain d'efficacité en réduisant les coûts de production [1] d'un logiciel de deux façons. Premièrement, le temps et l'effort requis pour l'amélioration et la correction de bogues est considérablement réduit (LERNER et TIROLE, 2002), le traitement étant « parallélisable » (un même problème ou tâche est traitée par plusieurs personnes ou processeurs) (RAYMOND, 2001a). Par opposition, il arrive souvent que les programmeurs travaillant pour une entreprise commerciale ne connaissent pas le fonctionnement précis d'un logiciel, étant donné qu'ils ne travaillent que sur la section de code qui leur est assignée, sans nécessairement avoir accès aux sections assignées à leurs collègues. Évidemment, cela n'est pas efficace puisque, souvent, des fonctions similaires se retrouvent à plusieurs endroits dans le logiciel (RAYMOND, 2001a).

Deuxièmement, l'accès au code source permet des externalités dites « d'anciens étudiants » (*alumni effect*) : les logiciels libres étant gratuits et modulaires, ils sont plus susceptibles d'être étudiés en classe. Selon Lerner et Tirole (2002), les programmeurs arrivent alors sur le marché du travail en étant déjà familiers avec des logiciels libres, leurs structures et les méthodes utilisées pour résoudre les problèmes rencontrés. Les auteurs définissent ces avantages comme des gains non monétaires des programmeurs participants, mais comme ces avantages profitent plus au projet qu'à n'importe quel programmeur pris individuellement, il est jugé plus pertinent de compter ces deux caractéristiques comme diminuant les coûts de production plutôt que comme gain du programmeur.

1.2 Qu'est-ce qui compose une licence ?

Tel qu'il a été mentionné précédemment, les licences des logiciels servent à protéger les auteurs de codes source par des droits d'auteur. Dans leur définition la plus stricte, selon l'Office de la langue française du Québec, elles sont des documents juridiques

1. Les coûts de production des logiciels libres sont plutôt en termes de temps, de complexité, de main-d'oeuvre qualifiée, etc.

régissant la concession du droit d'utilisation d'un logiciel. Plusieurs lois et plusieurs principes sont derrière les licences libres, et ces dernières comportent quelques sections obligatoires.

Premièrement, la licence libre doit spécifier quel niveau de *copyleft* sera utilisé. Au milieu des années soixante-dix, certains programmeurs ont décidé d'assurer la gratuité de leurs logiciels tout en se moquant gentiment des droits d'auteur. C'est ainsi que le projet Tiny Basic de la People's Computer Company débutait son code source par :

```
TINY BASIC FOR INTEL 8080
VERSION 1.0
BY LI-CHEN WANG
10 JUNE, 1976
@COPYLEFT
ALL WRONGS RESERVED
```

Le jeu de mots est intraduisible, mais l'esprit est clair : les droits d'auteurs deviennent des « gauches d'auteurs », c'est-à-dire que le programmeur abandonne une partie de ses droits tels qu'ils sont définis par la *Loi sur le droit d'auteur* et accepte que son code source soit modifié et redistribué par quiconque le désire. Les obligations décrites ci-dessus font que le code source d'un logiciel sous copyleft ne tombe pas dans le domaine public.

Un copyleft très restrictif fera que la licence se propagera de manière virale pour toute récupération de code, même les bibliothèques logicielles, tandis qu'un copyleft faible permettra de récupérer des sections du code, voire de les inclure dans un logiciel commercial.

Le tableau 1.3 illustre bien certaines propriétés du copyleft. La licence GPL, étant donné sa propagation de type virale, fera en sorte qu'il sera beaucoup plus facile de remarquer les programmeur ayant participé au projet ; si une compagnie de logiciel veut récupérer un projet sous licence GPL, alors elle devra obtenir l'accord du chef du projet, utiliser à son tour la GPL et mettre les contributions individuelles en valeur. À l'opposé, la licence BSD encourage la récupération, et sur son site Web il est même suggéré d'enfouir la

Tableau 1.3 : Exemples de licences libres et de sévérité des licences libres

CRITÈRES	BSD (permissives)	LGPL (restrictive)	GPL (très restrictive)
Conforme à l'OSI	✓	✓	✓
Code source des modifications requis		✓	✓
Intégration dans un logiciel commercial	✓	✳	
Possibilité de changer de licence	✓		

✳ : la LGPL permet la récupération des bibliothèques logicielles seulement.

licence, la liste des contribution individuelles, etc. à la fin du contrat de licence du logiciel commercial qui a récupéré un projet libre assujetti à la licence

La licence devra de plus définir les droit d'auteurs : qui est responsable pour chacun des modules, et comment ces droits d'auteurs peuvent être transférés. Troisièmement, la définition de *logiciel Open Source* doit être incluse. Enfin, une section invariante doit être définie (une portion du code qui ne peut être modifiée, sous aucun prétexte). Le code source initial d'un logiciel libre (la première version) doit être préservé. Les modifications ultérieures doivent se faire sous forme de rustines[2]. Une licence commerciale, pour sa part, stipule en général que le client (l'utilisateur du logiciel) ne peut pas copier, redistribuer ni modifier le logiciel ; il ne peut que l'utiliser de la manière prescrite par le fabricant.

Il y a aussi quelques obligations rattachées à ces licences, tant pour les programmeurs que pour les utilisateurs. Le code source originel et le code des modifications doivent être inclus dans une distribution du logiciel. Les fichiers journaux, qui ont été décrits ci-dessus, doivent aussi être inclus dans une distribution. Ces fichiers ont d'ailleurs une double fonction. Non seulement ils servent à identifier qui a fait quoi, mais aussi ils permettent d'envoyer un signal à l'industrie (LERNER et TIROLE, 2002). Ce signal est perçu par des employeurs potentiels (la participation à un projet Open Source peut

2. De l'anglais *patch*, sorte de module facultatif qui modifie des sections du code source d'un logiciel lors de la compilation.

donc déboucher sur un emploi) et des investisseurs en capital de risque (qui peuvent vouloir démarrer une compagnie dont le produit semble prometteur).

La licence a aussi deux autres conséquences. Un effet de foule (*exhibit fads*) fait qu'une licence connue des programmeurs risque d'être préférée à une nouvelle licence, même si cette dernière procure de plus grands bénéfices à tous les participants. Elle doit aussi servir à prévenir la *mutinerie* au sein d'un projet (*forking of a project*). Cette situation se produit lorsqu'un groupe de programmeurs participant à un projet décident de diriger le projet dans un autre direction que celle déterminée par le chef du projet.

1.3 Qu'est-ce qu'un projet Open Source ?

Un projet Open Source peut être défini comme un ensemble de différents modules soumis à une licence libre et auquel est joint un fichier journal. Par extension, on peut inclure l'ensemble des participants au projet. Dans le but d'analyser le marché des logiciels libres, tout module sera considéré comme un projet à part entière. Puisqu'un programmeur peut se joindre à n'importe quel projet et ne travailler que sur les parties qui l'intéressent (pour peu qu'il soit suffisamment compétent), il tombe sous le sens de simplifier en considérant tout travail de programmation comme un projet Open Source. Cette définition n'affectera pas la généralité de la présente analyse, celle-ci ne portant pas sur les aspects techniques des logiciels libres.

Un projet Open Source voit le jour lorsqu'un programmeur bute sur un problème et tente de le résoudre en créant un logiciel. Il doit écrire suffisamment de code pour démontrer sa crédibilité, et laisser une marge de manœuvre suffisamment importante pour que d'autres programmeurs aient des défis eux aussi (RAYMOND, 2001a). Le chef (*leader*) du projet doit aussi choisir la licence libre à laquelle le projet sera assujetti. Toujours selon Raymond (2001), les programmeurs ne travaillent pas sur des projets Open Source par grandeur d'âme. Programmeur et chef de projet Open Source lui-même, l'auteur voit deux raisons majeures qui motivent les programmeurs :

> The "utility function" Linux hackers are maximizing is not classically economic, but is the intangible of their own ego satisfaction and reputation among other hackers.

Lerner et Tirole (2002) distinguaient les motivations idéologiques des programmeurs du plaisir que ceux-ci retirent de participer à un projet Open Source : certains programmeurs participent à un projet Open Source uniquement dans le but d'utiliser un certain type de licence. Ceux-ci voudront, par exemple, s'opposer au monopole que constitue Microsoft®. Dans le cadre de ce mémoire, étant donnée les particularités de la fonction d'utilité des programmeurs proposées par Raymond (2001), cet aspect sera inclut dans les motivations psychologiques des programmeurs.

En plus de cet aspect « psychologique » des gains que peut retirer un programmeur, on peut compter des bénéfices matériels. Il a été mentionné précédemment qu'un signal est envoyé à l'industrie par l'intermédiaire les fichiers journaux. Ce signal, qui est en réalité un incitatif lié à la carrière, est considéré comme un bénéfice de nature psychologique par Tirole et Lerner (2002). Comme l'emploi ou l'investissement escompté est en réalité une somme monétaire, ce bénéfice sera ici considéré comme faisant partie des bénéfices de nature monétaire. Le chef d'un projet peut aussi compter sur des produits dérivés. Tout en laissant le logiciel libre et gratuit, suite à un succès relativement important, il peut devenir intéressant de fournir du soutien technique, des logiciels complémentaires ou encore des livres en fondant des entreprises à cet effet.

Il n'y a cependant pas que des avantages à participer à un projet Open Source. Le programmeur doit compter un coût d'opportunité assez important pour sa participation. Celui-ci est constitué de divers éléments. Il y a premièrement le temps. Le programmeur qui travaille sur un projet Open Source n'utilise pas son temps pour un emploi rémunéré. Le temps se traduit donc par la perte de salaire encourue par le fait de ne pas travailler dans une compagnie commerciale. Sans compter que s'il travaille en parallèle pour une compagnie commerciale, le temps passé sur le projet Open Source est du temps non consacré à son « vrai » travail (LERNER et TIROLE, 2002). Il y a aussi

une portion irrécupérable qui croît inversement par rapport au talent du programmeur, soit le coût d'apprentissage du code source.

Ainsi, le programmeur participera à un projet Open Source à la suite d'un simple calcul utilitariste : si la différence entre ses gains anticipés et son coût d'opportunité est positive, alors il a intérêt à participer à un projet Open Source (TIROLE et LERNER, 2005). Tirole et Lerner (2005) distinguaient la fonction d'utilité du chef de projet de celle du programmeur participant. Cette distinction ne semble pas pertinente puisque, peu importe sa situation dans le projet, un programmeur est un programmeur. C'est d'ailleurs ce qui ressort de Raymond (2001). Il sera ici considéré qu'il n'y a qu'une seule fonction d'utilité, valide pour la population entière de programmeurs.

La licence doit être choisie judicieusement. Il a déjà été mentionné que le débat entre la FSF et l'OSI se situe plus sur un plan moral que technique. Il en va de même pour les licences libres. Les tenants des licences BSD arguent que les libertés imposées par la GPL ne valent pas celles que l'on peut choisir en adoptant une licence de type BSD, tandis que la FSF et ses tenants disent que les licences de type BSD sont une porte ouverte au « vol » des logiciels par les compagnies commerciales (RAYMOND, 2001a). C'est ce qui est appelé *détournement* (*hijacking*) d'un logiciel, et la licence doit notamment servir à empêcher que cela se produise. Dans les prochains chapitres, deux licences sont retenues : la BSD (le plus faible copyleft) et la GPL (le plus fort copyleft).

1.4 De l'informatique à l'économique

Afin de mieux rendre compte des études de nature économique qui ont été réalisé sur les logiciels libres et les incitatifs auxquels peuvent faire face les programmeurs, voici quelques textes qui ont été publiés ces dernières années et qui, entre eux, représentent bien l'état du sujet.

Les premiers textes ont en fait été publiés par des informaticiens. Évidemment, ils ne présentent pas de modélisation formelle, mais il exposent néanmoins les intuitions

fondamentales nécéssaires à une éventuelle modélisation telle que celle réalisée dans ce mémoire. Dans *The Cathedral and the Bazaar* (RAYMOND, 2001a) Eric Raymond a publié ses essais sur les logiciels Open Source. Il commence par relater l'évolution de l'informatique et ce qu'est, aujourd'hui, un *hacker*, soit un programmeur enthousiaste et non un pirate. Le second texte, qui a donné son titre au livre, compare les deux modes de programmation (Open Source et institutionnel ou commercial). Le mode de production institutionnel est comparable à une cathédrale, où rien n'est laissé au hasard et la moindre ligne est étudiée longuement avant d'être adoptée. La communauté Open Source, elle, est comparable au bazar, où l'échange, la discussion et le fouillis créationnel sont la base de tout ce qui s'y réalise. C'est aussi dans ce texte que sont énoncées les « lois » de la programmation selon Raymond, un ensemble de constats et d'opinions largement partagées sur le dévelopement des logiciels. Un dernier texte de ce livre est digne de mention dans le présent contexte, qui s'intitule « Homesteading the Noosphere » (RAYMOND, 2001b). Dans ce texte, qui discute de la propriété intellectuelle, des droits et « gauches » d'auteurs, en s'appuyant sur des textes de John Locke et sur le théorème de Coase afin de démontrer que la coopération et l'échange produisent de meilleurs logiciels. Les autre textes de ce livre sont plus de nature anthropologique et tentent de convaincre le lecteur que le mode de production Open Source représente la solution de l'avenir pour les entreprises soucieuses de la qualité de leur logiciels.

Dans la même veine, un recueil de textes écrits principalement par des programmeurs tente de faire le tour du sujet. Dans *Perspectives on Free and Open Source Software*, vingt-quatre auteurs discutent des motivations derrière la programmation Open Source, l'évaluation des logiciels et du paradigme Open Source, les processus et les outils des programmeurs, l'économie des logiciels Open Source et les modèles d'affaire, la propriété intellectuelle et le droit, la communauté et la société. Comme le recueil précédent, ce recueil ne présente aucun modèle formel, mais dresse plutôt un portrait des logiciels libres à ce jour. Les articles présentés sont de différentes natures : socio-logique, économique, anthropologique, ingénierie logicielle et juridique. La multitude

des sujets couverts et le grand nombre d'articles empêchent de présenter un résumé détaillé de chacun, mais trois d'entre eux méritent une mention particulière. Dans « Has Open Source a Future » (FITZGERALD, 2005), Brian Fitzgerald discute des problèmes inhérents à la programmation Open Source. Outre ceux de nature technique, un problème retient l'attention : le manque de talent fort. Selon l'auteur, un des problèmes auxquels doit faire face la communauté Open Source est le grand nombre de programmeurs enthousiastes mais de talent moyen ou faible, qui ne sont pas en mesure de « survivre » ; la loi de la jungle aura tôt fait de les reléguer au rang d'illustres inconnus plutôt qu'à celui de développeur de talent.

Dans « Why Hackers do what they do : Understanding Motivation and Effort in Free/Open Source Software Projects » (LAKHANI et WOLF, 2005), Karim Lakhai et Robert Wolf présentent les résultats d'une enquête effectuée auprès de 684 développeurs participant à 287 projets. Cette enquête recherche les vraies motivations des programmeurs participant à un projet Open Source. Deux catégories principales de motivations semblent prévaloir : les motivations intrinsèques (particper pour la satisfaction de la participation) et les motivations extrinsèques (participer dans le but de recevoir quelque chose d'un tiers). Tandis que l'altruisme, le sens de la communauté et le plaisir personnel sont des sources de motivation intrinsèque, la résolution d'un problème et la recherche de rémunération pécuniaire sont des motivations extrinsèques.

Le troisième texte permet vraiment de faire le pont entre l'informatique et l'économique. Dans « Some simple Economics of Open Source », un article en fait paru quelques années auparavant (LERNER et TIROLE, 2002), Jean Tirole et Josh Lerner ont analysé, un phénomène relativement peu connu et très peu étudié en économique, mais qui tend à prendre de l'expension. La programmation Open Source défie les lois économiques, selon eux, puisque des milliers de programmeurs développent des logiciels sans rémunération pécuniaire, et les logiciels produits sont de meilleure qualité que ceux développés par les grandes compagnies. De plus, ils sont gratuits. Cet article se veut une exploration préliminaire du sujet. Presque tous les thèmes mentionnés précédemment sont abordés dans cet article. Les principes d'économie du travail sont

utilisés afin d'identifier les incitatifs des programmeurs, qui veulent envoyer un signal à l'industrie afin d'obtenir un emploi. Sans élaborer de modèle, les auteurs présentent leurs conclusions sur les motivations des programmeurs, effectuent quatre études « empiriques » de projets Open Source qui ont connu un énorme succès en termes de popularité, étudient les défis auxquels la communauté Open Source sera confronté et concluent en effectuant un parallèle avec la vision économique des innovations.

Ces mêmes auteurs ont publié, deux ans plus tard, un second article sur ce sujet, « The Scope of Open Source Licensing » (TIROLE et LERNER, 2005). Cette fois, comme le titre l'indique, les auteurs se sont attardés sur l'aspect juridique des logiciels libres. Encore une fois, aucun modèle n'est développé, mais des éléments plus formels sur la fonction d'utilité des programmeurs, leurs préférences et leur décision commencent à faire leur apparition. Les différentes licences libres sont analysées afin de les modéliser. Les auteurs ont procédé à une étude empirique sur plus de 40 000 projets Open Source, sous forme d'enquête, afin de déterminer les motivations de programmeurs et les éléments analysés par eux et la communauté Open Source lors du choix d'une licence libre. Leur principale conclusion est que les projets destinés aux systèmes d'exploitation commerciaux tendent à utiliser des licences plus restrictives, mais les projets utilisant une licence moins restrictive attirent plus de programmeurs. Ce qui est conforme aux prédictions théoriques des éléments de modélisation présentés.

Plusieurs articles ont par la suite été publiés par rapport à la place du copyleft en science écnonmique. Par exemple, dans « Copyleft – the economics of Linux and other open source software » (Mustonen, 2003), Mikko Mustonen compare l'apparition du copyleft aux rémunérations non pécuniaires que reçoivent les chercheurs qui publient des articles scientifiques. Il construit un modèle dans lequel les choix que font les programmeurs (programmation Open Source ou travail pour une compagnie) sont le principal déterminant de la qualité des logiciels disponibles pour un utilisateur (dans le commerce ou gratuitement). Ensuite, il intègre un monopoleur, qui représente le fournisseur chez qui un utilisateur peut acheter un logiciel. Ce monopoleur doit tenir compte de l'impact des logiciels libres. Il démontre que lorsque les coûts d'apprentis-

sage d'un logiciel sont faibles, le monopoleur tolère la présence de la communauté Open Source. Ainsi, cet article traite plutôt du marché des logiciels, en tentant d'expliquer la présence des logiciels libres et commerciaux, et l'impossibilité pour l'un ou l'autre d'éliminer son rival. Du point de vue des programmeurs, les logiciels assujettis à un copyleft plutôt qu'à un droit d'auteur sont une manière d'obtenir un revenu complémentaire. Il conclut en disant que la qualité des logiciels commerciaux est tributaire des logiciels libres, mais que le copyleft perd sa raison d'être si les logiciels commerciaux disparaissent.

Dans « Le logiciel libre : une nouvelle approche de la propirété intellectuelle » (ZIMMERMAN et JULIEN, 2002), Jean-Benoît Zimmerman et Nicolas Julien abordent le problème de la propriété intellectuelle que posent les logiciels libres. Cet article est de nature juridique, mais présente une description économique du copyleft et des impacts possibles sur les entreprises qui produisent des logiciels, le statut quasi-public des logiciels assujettis à une licence libre et le mode de production de la communauté Open Source. Les auteurs concluent en résumant les principaux problèmes inhérents au copyleft et à ce mode de production, tout en insistant sur le fait que la qualité des logiciels produits et l'efficacité de ce mode de gestion de propriété intellectuelle sont indéniables.

Justin Pappas Johnson, dans « Economics of Open Source Software » (Johnson, 2001), traite les logiciels libres comme des biens publics. Un programmeur choisit de participer ou non à un projet Open Source, ce qui lui coûte le temps consacré à ce projet plutôt qu'à une autre activité. Le code produit peut ainsi être utilisé par quiconque, mettant ainsi le talent du programmeur au service de la communauté. Par, contre un phénomène de parasitisme[3] peut apparaître étant donné l'absence de profit ; cette situation, selon lui, ferait diminuer l'innovation au sein des programmeurs. Le modèle qu'il développe dans cet article est un équilibre bayesien, dans lequel un équilibre est défini comme un ensemble de stratégies (développer un logiciel libre ou non,

3. *free riding*

étant donné les coûts personnels) et de croyances (est-ce qu'un autre programmeur pourrait développer ce logiciel sans que le premier n'ait à se forcer) qui sont cohérents les uns par rapport aux autres. Les paiements que reçoivent les programmeurs ne sont que la récupération des coûts personnels que le développement a demandé. Il conclut qu'en général, les niveaux d'efforts requis et la distribution de l'effort parmi les programmeurs sont inefficaces (trop dilués ou simplement trop faibles). Johnson évalue aussi les déterminants de la taille du bassin de programmeurs de logiciels libres, et détermine que le nombre de programmeurs est important ; un grand nombre implique de meilleures probabilités de développement de logiciels, mais la probabilité de parasitisme augmente en conséquence. D'autres thèmes sont aussi évalués, tels que la structure modulaire des logiciels libres par rapport à la structure monolithique des logiciels commerciaux. Un mode de production de type Open Source, en permettant à chacun de développer ce qu'il désire, augmente la taille du capital humain mais invite au parasitisme. Une production commerciale n'exploite pas à fond les talents individuels, mais ne fait pas face au parasitisme et profite du bien-être agrégé que procure sa production à ses consommateurs. Enfin, la caractéristique qui distingue le plus cet article des autres mentionnés précédemment est que Johnson semble le seul à considérer les logiciels libres comme un bien entièrement public, ce que nient les autres économistes et les programmeurs ayant publié sur le sujet.

Plusieurs textes sur le sujet ont trait plus particulièrement aux motivations des programmeurs. Nombre d'entre eux sont cependant des études empiriques, dont les principaux résultats sont la confirmation de ce que les essais et articles mentionnés précedemment rapportent. Sinon, la manière dont les motivations sont abordées diffère largement de l'approche retenue dans ce mémoire. Par exemple, Yossi Spiegel étudie, dans « The Incentive to Participate in Open Source Project : A Signalling Approach » (spi,), un seul incitatif, soit la signalisation à l'industrie. Il modélise la participation à un projet Open Source comme un jeu de signaux, et démontre que divers facteurs (la visibilité de la performance, la portée du signal de performance) ont un impact plus ou moins important sur le signal envoyé, selon l'équilibre envisagé. Un équilibre (bayesien

parfait), dans ce modèle, se compose d'un ensemble de stratégies (la participation des programmeurs selon le talent) et de croyances (la perception des programmeurs par les firmes qui peuvent les embaucher) qui sont cohérentes les unes par rapport aux autres.

Enfin, un dernier article se doit d'être mentionné. Celui-ci ne traite pas des logiciels libres, mais plutôt des motivations qui peuvent influencer un *participant* à une *activité* quelconque. Dans « Intrinsic and Extrinsic motivations » (BENABOU et TIROLE, 2003), Roland Bénabou et Jean Tirole tentent de réconcilier le point de vue des économistes, qui affirment que les incitatifs mènent à de meilleurs résultats, et celui des psychologues, qui affirment plutôt que les récompenses et les punitions obtenues suite à une action vont plutôt diminuer les raisons personnelles qui poussent les gens à entreprendre une activité. Dans cet article, les auteurs développent un cadre d'application général, dans lequel deux joueurs (un agent et le principal) participent à une activité. L'agent détermine son niveau d'effort, qui, lui, détermine le paiement de l'agent et du principal. Le principal, lui, possède une information supplémentaire (par exemple, la difficulté de la tâche) qui a un impact sur le paiement de l'agent. Dans ce cadre, le principal choisit aussi un élément de motivation externe à l'agent. La résolution de ce modèle s'effectue en deux étapes : le principal prend connaissance de l'information supplémentaire sur l'agent et choisit l'incitatif externe. Ensuite, l'agent observe cet incitatif et le paiement afin de déterminer son niveau d'effort. Ce modèle général est par la suite adapté à diverses situations, dans lesquelles les motivations *intrinsèques* de l'agent (ses raisons personnelles pour participer au projet, ses croyances envers lui-même) et les motivations *extrinsèques* (les incitatifs externes déterminés par le principal) sont analysées l'une par rapport à l'autre dans un modèle d'équilibre bayesien. Un équilibre, dans ce modèle, consiste en un ensemble de stratégies (le niveau d'effort fourni par l'agent) et de croyances (la véracité de l'information supplémentaire détenue par le principal) sont rationnelles les unes par rapport aux autres. Les auteurs concluent en affirmant que les différentes visions sur ce sujet ne sont pas incompatibles ; les incitatifs externes aux participants peuvent être une source de motivation, mais peuvent facilement se retourner contre celui qui fournit ces incitatifs.

Bien entendu, les textes présentés ici sont loin d'être une revue de littérature exhaustive, mais représentent bien les différentes approches dans la recherche sur ce sujet. Le modèle présenté dans les prochains chapitres est basé sur le cadre général présenté ici, mais adapté à la production des logiciels libres selon les éléments tirés des autres textes mentionnés.

CHAPITRE II

MODÉLISATION DE BASE

Le premier modèle développé tente de reproduire la situation actuelle de la communauté Open Source. Le modèle ne comportera qu'une seule licence puisque, actuellement, plus de 50 % des logiciels libres sont assujetti à la licence GPL. La raison pour laquelle il s'agit de la GPL est son caractère contaminant : la visibilité des programmeurs se trouve mise de l'avant, et le signal qui est envoyé à l'industrie est, *a priori*, plus fort. En fait, il faut comprendre ici une chose : les programmeurs, peu importe leurs motivations personelles, visent essentiellement la même chose : une amélioration, ou plutôt une meilleure perception de leur réputation par la communauté.

Donc, comme il a été mentionné, ce premier modèle ne tient pas compte des motivations des programmeurs. Il est ici considéré qu'il n'y a qu'une seule raison pour qu'un programmeur participe à un projet Open Source, soit ses considérations quant à sa réputation. Aussi, le talent des programmeurs est inobservable, ce qui est une première source d'incertitude. La communauté Open Source ne peut qu'estimer le talent d'un programmeur selon ses contributions.

Il sera donc question de la motivation « réputationnelle » des programmeurs et non de leurs motivations personelles. Il sera dorénavant considéré que chacun participe à la communauté Open Source dans le but d'être perçu comme étant doté d'un grand talent. Ce faisant, le programmeur peut envoyer un signal plus intérressant tant à la

communauté qu'à l'industrie ; le gain qu'un programmeur retire de sa participation à un projet libre sera donc une meilleure réputation au sein de ses pairs.

Un programmeur qui participe à un projet devra passer par deux étapes afin de voir sa réputation améliorée. Il doit d'abord réussir son projet : une *réussite* signifie qu'un projet a été entrepris et que le logiciel en résultant fonctionne. Ensuite, étant donné un logiciel réussi, il faut que la communauté le voie. La licence peut y jouer un rôle, tout comme la nature et la difficulté du projet, l'intérêt de la communauté, etc. Ainsi, après une réussite, un programmeur doit se démarquer des autres projets. Concrètement, il sera question dans le modèle d'une probabilité de réussite et d'une probabilité de visibilité d'un projet. C'est seulement après une réussite visible que la réputation d'un programmeur est augmentée. Le tableau 2.1 résume cette nomenclature.

2.1 Déroulement du jeu

Les programmeurs, ou les joueurs dans le modèle, sont tous ceux qui ont déjà fait le choix de participer à un projet Open Source. Ils sont hétérogènes, chacun étant d'un type $\theta \in [0,1]$ différent, qui représente son talent naturel, associé à une densité $f(\theta)$. Ce θ est distribué uniformément ($f(\theta) = 1$), donc le talent est réparti également dans la population. Cette hypothèse sera retenue à des fins de simplicité [1].

Le talent est la compétence *ex ante* du programmeur i telle qu'elle est perçue par la communauté. Par exemple, un programmeur de talent aura une meilleur réputation, puisqu'il est probable qu'il a déjà produit du code digne d'impressionner la communauté Open Source, alors qu'un programmeur débutant, qu'il soit talentueux ou non, n'aura qu'une réputation faible. Si le jeu était répété, son talent élevé lui permettrait d'obtenir plus facilement un premier succès, de se faire remarquer et donc d'acquérir une bonne réputation rapidement.

---

1. Il serait relativement simple de remplacer la distribution uniforme par une distribution normale.

Tableau 2.1 : Lexique des types de succès

Réussite : Une réussite survient lorsqu'un projet devient un logiciel à proprement parler, qu'il produit des résultats utilisables.

Visibilité : Suivant une réussite, un projet ou un programmeur doit se distinguer au sein de la communauté, afin d'acquérir une certaine popularité ou renommée.

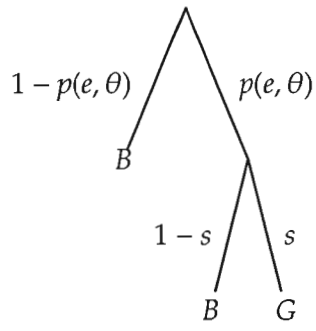
Succès : Un projet ou un programmeur obtient un succès lorsqu'il y a réussite et distinction. C'est un succès qui permet aux programmeurs un gain en réputation.

Chaque programmeur possède une richesse initiale [2], de laquelle il doit déduire un coût d'opportunité (le coût de l'effort à fournir pour connaître l'état d'un projet). Les bénéfices tirés de la participation à un projet Open source ne sont pas monétaires, mais ont plutôt trait à la réputation. C'est pourquoi il sera désormais question de gains en termes de réputation.

Il y a donc deux événements possibles. L'événement G signifie qu'il y a eu un succès. L'événement B est son complément, c'est-à-dire que le participant demeure dans l'anonymat parce qu'il n'a pas réussi son projet ou qu'une réussite est demeurée « invisible ». La probabilité de visibilité (s) est exogène et déterminée par la communauté Open Source. Autrement dit, la réussite se produit avec probabilité $p(e,\theta)$ et est remarquée avec probabilité s. Cette probabilité de réussite est croissante et concave par rapport à l'effort, et est croissante par rapport au talent des programmeurs $\left(p'_e(e,\theta) > 0,\ p''_e(e,\theta) > 0,\ p'_\theta > 0\right)$. De plus, la productivité marginale du talent est croissante peu importe le niveau d'efforts ($p''_{e,\theta} > 0$). Donc pour tout effort fourni, un talent plus élevé aura un effet positif sur la probabilité de réussite. Le déroulement du jeu est illustré dans la figure 2.1

Au noeud initial de l'arbre de la figure 2.1a, le programmeur a entrepris ou participe à un projet Open Source. C'est la première des deux étapes mentionnées précédemment : le projet peut réussir ou échouer. En cas d'échec, il y a réalisation de l'événement B,

2. Il est ici question de la richesse au sens large, c'est-à-dire des avoirs pécuniaires et « psychologiques ».



(a) Arbre de jeu

événement	probabilité
$\theta_S = \begin{cases} \overline{\theta}_G \\ \overline{\theta}_B \end{cases}$	$p[\overline{\theta}_G] = s \cdot p(e, \theta)$ $p[\overline{\theta}_B] = 1 - s \cdot p(e, \theta)$

(b) Espérances de gain

Figure 2.1 : Probabilités des événements

qui procure un gain réputationnel « $\overline{\theta}_B$ » au programmeur. En cas de réussite, qui se produit avec probabilité $p(e, \theta)$ tel qu'il est défini ci-dessus, la deuxième étape survient. Le projet doit devenir visible aux yeux de la communauté Open Source. La probabilité de visibilité étant s, et le gain en cas de succès étant dénoté $\overline{\theta}_G$, l'espérance de gain en cas de succès est donnée par $s \cdot p(e, \theta)$. L'événement B étant le complément de G, le gain anticipé en cas d'échec ou de non-visibilité est donc $1 - (s \cdot p(e, \theta))$. La figure 2.1b résume pour sa part les gains anticipés selon l'événement réalisé. Les événements G et B sont mutuellements exclusifs (il n'est pas possible d'obtenir à la fois un succès et un échec) ; le gain qu'obtient un programmeur θ_S ne pourra prendre qu'une seule valeur parmi les deux possibles, respectivement $\overline{\theta}_G$ et $\overline{\theta}_B$. Ainsi, il est possible pour un programmeur, *a priori*, d'anticiper le gain auquel il peut s'attendre de sa participation à un projet Open Source. La variable aléatoire $\widetilde{\theta}_s$ représente cette anticipation, et elle peut prendre l'une de deux valeurs possibles, soit $\overline{\theta}_G$ ou $\overline{\theta}_B$. L'espérance du gain anticipé est donc l'espérance de la variable aléatoire $\widetilde{\theta}_s$:

$$E\left[\widetilde{\theta}_s\right] = p\left[\overline{\theta}_G\right]\overline{\theta}_G + p\left[\overline{\theta}_B\right]\overline{\theta}_B \quad \text{pour } \widetilde{\theta}_S \in \{\overline{\theta}_G, \overline{\theta}_B\} \tag{2.1}$$

2.2 Fonction d'utilité des programmeurs

Les éléments déterminés précedemment permettent de définir une fonction d'utilité pour les programmeurs de la communauté Open Source :

$$U = \mathscr{V} - c(e) + \beta\theta_S \tag{2.2}$$

Le premier terme est la richesse initiale du programmeur, telle qu'elle a été décrite précedemment. Le second terme est le coût de participation lié à l'effort. Le paramètre β représente l'intérêt de chacun des programmeurs face à la programmation Open Source ; ce n'est pas un facteur d'actualisation, puisqu'il n'y a qu'une seule période. En fait, il s'agit de la volonté d'un programmeur de voir sa réputation augmenter. Enfin, la variable d'intérêt θ_S est le gain qu'un programmeur retire de sa participation à un projet Open Source, donc la perception que les autres ont quant au talent d'un programmeur. Il est à noter que les programmeurs sont hétérogènes par leur talent, mais homogènes par ailleurs.

Les programmeurs ne connaissent initialement pas les gains qu'ils obtiendront *ex post*. La figure 2.1 montre deux niveaux d'incertitudes : la réussite du projet est incertaine, comme la visibilité suivant une réussite. De plus, les gains retirés d'une participation sont tout aussi incertains : le talent du programmeur, qui n'est pas connu, est un élément déterminant de (1) la probabilité de succès et (2), des gains selon l'événement qui se réalise. Il faut donc définir la fonction de gain anticipé, qui est basée sur les probabilités que le projet entrepris produise des résulstats intéressants.

Soit

$$p(e, \theta) = \theta e \; \forall \; e \in [0, 1] \tag{2.3}$$

la probabilité de réussite liée à l'effort, la première étape du modèle. Plus l'effort est grand, plus le succès est probable. Le type du programmeur, lui, est inobservable ; cependant, on peut constater que pour une même chance de succès, l'effort que devra fournir un programmeur de talent pourra être plus petit que celui d'un programmeur de talent moindre, ce qui correspond aux restrictions posées sur $p(e, \theta)$ posées ci-dessus. En cas de réussite, un programmeur peut alors passer à la deuxième étape, la visibilité. S'il est remarqué, le programmeur obtient un succès ; la probabilité de succès (l'événement G) est donc définie comme

$$P[G] = s \cdot p(e, \theta) = s\theta e \tag{2.4}$$

La figure 2.1 illustre les probabilités liées à chacun des deux évènements. Avec ces probabilités, et les réalisations possibles du gain $\left(\theta_S \in \left\{\overline{\theta}_G\ ,\ \overline{\theta}_B\right\}\right)$, la fonction d'utilité anticipée peut être construite de la façon suivante :

$$E(U) = \mathscr{V} - c(e) + \beta \cdot E[\widetilde{\theta_S}] \tag{2.5}$$

Si le coût en termes d'efforts est donné par

$$c(e) = \frac{1}{2}e^2 \tag{2.6}$$

alors la substitution des différents termes de la fonction d'utilité par leur définition permet de définir la fonction d'utilité anticipée ainsi :

$$
\begin{aligned}
E[U] &= \mathscr{Y} - \frac{1}{2}e^2 + \beta\left[s \cdot p(e,\theta)\overline{\theta}_G + 1 - \left(s \cdot p\,(e,\theta)\,\overline{\theta}_B\right)\right] \\
&= \mathscr{Y} - \frac{1}{2}e^2 + \beta\left[s\theta e\overline{\theta}_G + \left(1 - s\theta e\overline{\theta}_B\right)\overline{\theta}_B\right] \\
\overline{U} &= \mathscr{Y} - \frac{1}{2}e^2 + \beta\overline{\theta}_B + \beta s\theta e\left(\overline{\theta}_G - \overline{\theta}_B\right)
\end{aligned}
\tag{2.7}
$$

L'utilité anticipée d'un programmeur, au moment où il s'embarque dans un projet Open Source, est donc une comparaison du coût en termes d'efforts et du gain qu'il s'attend à obtenir, en considérant que son talent (sa réputation) détermine ses chances de succès. Ainsi, la probabilité de succès ou d'échec est utilisée pour pondérer les événements possibles ; la probabilité de succès contribue positivement à l'utilité anticipée tandis que les probabilités d'échec y contribuent négativement. Les gains selon l'événement ($\overline{\theta}_G$, $\overline{\theta}_B$) sont donc exogènes du point de vue des programmeurs, mais ils sont déterminés de manière endogène, afin de déterminer un équilibre bayesien. Ces gains sont définis comme l'espérance conditionelle du talent, sachant l'événement réalisé :

$$
\begin{aligned}
\overline{\theta}_G &= E\,[\theta \mid G] \\
\overline{\theta}_B &= E\,[\theta \mid B]
\end{aligned}
\tag{2.8}
$$

2.3 Effort optimal

L'effort optimal représente la stratégie de meilleure réponse d'un programmeur selon son environnement. Autrement dit, quel effort fournira-t-il afin de maximiser ses probabilités de succès étant donné son type, les paiements anticipés en cas de gain ou d'échec et l'intérêt qu'il porte à la programmation Open Source. La fonction de meilleure réponse d'un programmeur vise à maximiser son utilité anticipée, et elle est donnée par :

$$\frac{\partial \overline{U}}{\partial e} = -e + \beta\left[\theta s\overline{\theta}_G - \theta s\overline{\theta}_B\right] = 0 \tag{2.9}$$

$$\hat{e}(\theta, \overline{\theta}_G, \overline{\theta}_B) = \beta s\theta\left(\overline{\theta}_G - \overline{\theta}_B\right) \tag{2.10}$$

L'équation 2.10, l'ensemble de stratégies des programmeurs, peut être écrite comme

$$\hat{e}(\theta, \overline{\theta}_G, \overline{\theta}_B) \equiv \hat{e}\left(\theta, \left(\overline{\theta}_G - \overline{\theta}_B\right)\right) \equiv \hat{e}(\theta, \Delta) \tag{2.11}$$

On peut constater que l'effort optimal sera modulé par le type de chacun, mais qu'il variera en fonction de la probabilité de se distinguer (s) et de l'importance du gain en termes de réputation. Il ne s'agit pas seulement de la valeur du gain, mais aussi de l'importance relative du gain en cas de succès par rapport au gain en cas d'échec. Ainsi, plus l'intérêt des programmeurs, leur talent ou l'écart entre le gain en cas de succès et le gain en cas d'échec est important, plus l'effort fourni sera grand. Une autre manière d'interpréter l'effort optimal est que, pour un effort donné, un grand talent implique que moins de gain ou d'intérêt sont requis pour obtenir un succès. Ce choix du niveau d'efforts fournis se trouve donc à être la stratégie du programmeur participant à un projet Open Source, qui sera déterminée lors de la résolution du modèle.

L'équation 2.11 permet de déterminer la probabilité de réussite liée à l'effort optimal :

$$p(\hat{e}(\theta, \Delta)) = \theta \cdot \hat{e}(\theta, \Delta) \tag{2.12}$$

$$= \beta s\theta^2\left(\overline{\theta}_G - \overline{\theta}_G\right) \tag{2.13}$$

2.4 Détermination de l'équilibre

La résolution du modèle permettra d'obtenir, d'une part, un effort optimal étant donné un talent initial pour chaque programmeur, qui sera en fonction des gains attendus. Ces paiements sont, pour leur part, déterminés en fonction de l'effort fourni et du talent du programmeur. Autrement dit, un équilibre bayesien parfait sera trouvé. Les gains en termes de réputation, suivant l'effort optimal, sont l'espérance de chacun des gains sachant l'événement réalisé. Ils sont accordés par les pairs, en cas de succès ou de non-succès ; ils permettent, entre autres choses, de déterminer un équilibre bayesien défini comme :

$$Eq = \left\{e^*(\theta), \overline{\theta}_G, \overline{\theta}_B\right\} \tag{2.14}$$

tel que :

- les croyances $\overline{\theta}_G$ et $\overline{\theta}_B$ sont rationnelles étant donné les stratégies et elle sont obtenues à l'aide de la règle de Bayes ; elles sont l'espérance de gain en termes de réputation d'un programmeur qui participe à un projet Open Source ;
- l'ensemble des stratégies $e^*(\theta, \Delta) = s\theta\left(\overline{\theta}_G - \overline{\theta}_B\right)$, c'est à dire le choix du niveau d'efforts fournis suivant le type du programmeur (son θ) et les croyances, soit les gains en termes de réputation attendus.

Ainsi, on peut définir les membres de l'équilibre de la manière suivante :

$$e^*(\theta) = \hat{e}\left(\theta, \overline{\theta}_G^*, \overline{\theta}_B^*\right) \tag{2.15}$$

$$\overline{\theta}_G^* = \int_0^1 \theta \cdot f(\theta|B)d\theta \tag{2.16}$$

$$\overline{\theta}_B^* = \int_0^1 \theta \cdot f(\theta|B)d\theta \tag{2.17}$$

Ces équations pour les gains en cas de succès et en cas d'échec sont déterminées en trois étapes. Dans un premier temps, la probabilité moyenne de succès dans la population sera déterminée par l'effort optimal en fonction du gain potentiel.

$$p(G) = \int_0^1 s \cdot p(e^*(\theta), \theta) f(\theta) d\theta \tag{2.18}$$

$$= \int_0^1 \beta \theta^2 s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right) \tag{2.19}$$

$$p(B) = \int_0^1 \left[1 - s \cdot p(e^*(\theta), \theta)\right] f(\theta) d\theta \tag{2.20}$$

$$= \int_0^1 1 - \beta \theta^2 s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right) \tag{2.21}$$

Ensuite, on peut trouver les fonctions de probabilité conditionnelles de gain à l'aide de la règle de Bayes (le ratio de la probabilité de succès pour un seul programmeur sur l'ensemble de la population).

$$f(\theta|G) = \frac{s \cdot p(e^*(\theta), \theta)}{p(G)} \tag{2.22}$$

$$= \frac{\beta \theta^2 s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}{p(G)} \tag{2.23}$$

$$f(\theta|B) = \frac{s \cdot \left[1 - p(e^*(\theta), \theta)\right]}{p(B)} \tag{2.24}$$

$$= \frac{1 - \beta \theta^2 s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}{p(B)} \tag{2.25}$$

Enfin, les probabilités conditionnelles permettent de déterminer la fonction de gain en termes de réputation.

$$\overline{\theta}_G = \int_0^1 \theta \cdot f(\theta|G) d\theta \tag{2.26}$$

$$= \int_0^1 \frac{\beta \theta^3 s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}{\int_0^1 \beta \theta^2 s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)} \tag{2.27}$$

$$\overline{\theta}_B = \int_0^1 \theta \cdot f(\theta|B)d\theta \tag{2.28}$$

$$= \int_0^1 \frac{1 - \beta\theta^3 s^2\left(\overline{\theta}_G - \overline{\theta}_B\right)}{\int_0^1 1 - \beta\theta^2 s^2\left(\overline{\theta}_G - \overline{\theta}_B\right)} \tag{2.29}$$

On peut constater que la probabilité moyenne d'échec dans la population est la probabilité de ne pas avoir de succès. Les calculs sont détaillés à l'Annexe A

Il est déjà clair que les croyances en cas d'échec ou de non-réussite (θ_B) seront influencées par le gain en cas de succès, par l'intérêt des programmeurs (β) ainsi que par la probabilité de distinction (s). Or, cette équation est quadratique. Il faut donc la résoudre en fonction des paramètres nommés, après avoir vérifié qu'une solution existe bel et bien dans les intervalles pertinents.

La résolution des équations 2.27 et 2.29 (qui est détaillée dans l'appendice A) donne les résultats suivants pour les gains en cas de succès ou d'échec :

$$\overline{\theta}_G^* = \frac{3}{4} \tag{2.30}$$

$$\overline{\theta}_B = \frac{\frac{1}{2} - \frac{1}{4}\beta s^2\left(\overline{\theta}_G - \overline{\theta}_B\right)}{1 + \frac{1}{3}\beta q^2\left(\overline{\theta}_G - \overline{\theta}_B\right)} \tag{2.31}$$

Ceux-ci sont quelque peu surprenants. Dans un premier temps, une valeur fixe pour $\overline{\theta}_G$ implique que lorsqu'il n'y a qu'une seule licence, la communauté Open Source considère qu'un programmeur atteignant un succès mérite une réputation de 0.75 (rappel : $\theta \in [0, 1]$). Le plus surprenant de ce constat est que les « conditions » auxquelles le programmeur fait face n'ont absolument aucune incidence sur le gain en cas de succès. Donc, même un programmeur de talent médiocre, ou qui travaille à un projet pour lequel l'intérêt de la communauté Open Source est nul ($s = 0$) se verra attribué un talent de 0.75 s'il obtient un succès.

Quant à $\overline{\theta}_B$, il est clair que ce n'est pas encore la valeur d'équilibre, l'équation étant quadratique justement en $\overline{\theta}_B$. Certaines informations peuvent cependant être inférées : si l'un des paramètres exogènes au programmeur est nul ($s = 0, \beta = 0$), alors $\overline{\theta}_B = \frac{1}{2}$, soit la moyenne de la population. Donc, en absence d'intérêt de part ou d'autre, rien ne peut être conclu sur le type des programmeurs ; on leur attribue la moyenne. De plus, la moyenne de la population étant $\overline{\theta} = 0.5$, et le gain en cas de succès étant 0.75, il faut donc que $\overline{\theta}_B$ soit inférieur à $\overline{\theta}_G$.

Afin d'obtenir la valeur d'équilibre pour le gain en cas d'échec, plusieurs étapes sont nécessaires :

1. démontrer formellement la borne supérieure de $\overline{\theta}_B$;
2. identifier les bornes des paramètres présents afin que l'équilibre soit cohérent ;
3. rechercher l'existence de solutions et les démontrer formellement ;
4. identifier la bonne racine de l'équation quadratique.

2.4.1 Preuve de l'existence de solution pour $\overline{\theta}_B$

Afin que l'équilibre soit cohérent, il est attendu que le gain en cas d'échec soit plus petit que le gain en cas de succès $\left(\overline{\theta}_B < \overline{\theta}_G\right)$. Le lemme 3 garanti non-seulement que cette condition est satisfaite, mais en plus que le gain en cas d'échec est inférieur à la moyenne. Autrement dit, un programmeur qui échoue son projet ou qui ne devient pas visible dans la communauté sera réputé avoir un talent inférieur à la moyenne.

Lemme 1 : $\overline{\theta}_B$ ***est inférieur à la moyenne de la population***

Puisque

$$\overline{\theta} = \frac{1}{2} = p(\overline{\theta}_G)\overline{\theta}_G + p(\overline{\theta}_B)\overline{\theta}_B$$

et que

$$\overline{\theta}_G = \frac{3}{4} > \overline{\theta}$$

il s'ensuit que

$$\overline{\theta}_B < \overline{\theta}$$

Puisque l'équation déterminée pour $\overline{\theta}_B$ n'est pas linéaire, il faut vérifier s'il existe au moins une solution satisfaisant aux hypothèses posées sur le gain en cas d'échec.

Dénotons tout d'abord le membre de droite de 2.31 par

$$\varphi\left(\overline{\theta}_B\right) = \frac{\frac{1}{2} - \frac{1}{4}\beta s^2\left(\frac{3}{4} - \overline{\theta}_B\right)}{1 - \frac{1}{3}\beta s^2\left(\overline{\theta}_G - \overline{\theta}_B\right)} \tag{2.32}$$

Dans la figure 2.2, les axes représentent $\varphi(\overline{\theta}_B)$ en fonction de $\overline{\theta}_B$ et on retrouve, en rouge, la bissectrice $\left(\varphi\left(\overline{\theta}_B\right) = \overline{\theta}_B\right)$, afin de vérifier graphiquement si une solution est possible dans les intervalles pertinents. Il apparait que des solutions existent pour certaines valeurs des paramètres d'intérêt puisqu'il y a des croisements entre certaines courbes et la bissectrice. Les bornes de $\overline{\theta}_B$ proviennent des hypothèses posées sur ce gain et du lemme 3 : le gain doit être positif, et sa valeur maximale est $\frac{1}{2}$, soit la moyenne de la population. La borne supérieure est identifiée par le trait vertical, à 0.5. Quelques exemples de courbes possibles de $\varphi\left(\overline{\theta}_B\right)$ y sont illustrés.

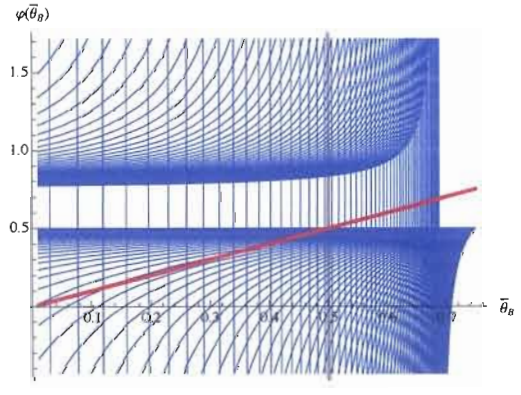


Figure 2.2 : Ensemble des valeurs possibles de $\varphi\left(\overline{\theta}_B\right)$ selon les différentes valeurs des paramètres

La figure 2.2 suggère l'existence de solutions : une partie des courbes croise la bissectrice. Cependant, il est évident que toutes les valeurs des paramètres β et s ne sont pas admissibles. Cette figure n'est cependant pas très claire. Afin de simplifier, la figure 2.3 montre les courbes pour seulement quelques valeurs de β et de s qui satisfont aux restrictions qui leur sont attribuées.

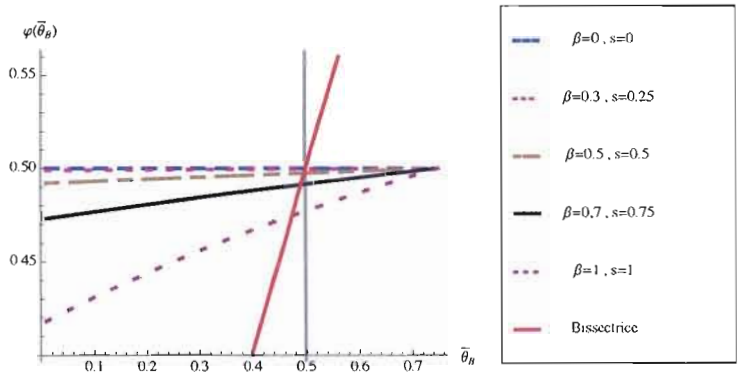


Figure 2.3 : Exemples de $\varphi\left(\overline{\theta}_B\right)$
pour quelques valeurs de β et s

La première chose à remarquer est que plus les valeurs de β et de s augmentent, plus la courbe se déplace vers le bas. Par exemple, lorsque β et/ou s est égal à zéro, la courbe est égale à $\frac{1}{2}$. Ceci implique qu'en l'absence d'intérêt de la part de la communauté, de la part des programmeurs face à leur réputation ou des deux, on ne peut pas tirer de conclusion quant au type du programmeur ; on ne peut que lui attribuer la moyenne de la population. Mais dans tous les cas de la figure 2.3, il y a croisement avec la bissectrice. Les valeurs admissibles pour β et s sont déterminées de telle sorte que (1) $\overline{\theta}_B$ soit comprise entre ses bornes $\left(0 < \overline{\theta}_B < \frac{1}{2}\right)$ et (2) les croisements avec la bissectrice sont assurés. Il faut donc que $\varphi(0) > 0$ et que $\varphi\left(\frac{1}{2}\right) > \frac{1}{2}$. Afin de simplifier la démonstration, l'équation $\varphi\left(\overline{\theta}_B\right)$ sera réécrite en développant les termes :

$$\varphi\left(\overline{\theta}_B\right) = \frac{\frac{1}{2} - \frac{1}{4}\beta s^2\left(\frac{3}{4} - \overline{\theta}_B\right)}{1 - \frac{1}{3}\beta s^2\left(\overline{\theta}_G - \overline{\theta}_B\right)} = \frac{\frac{1}{2} + \frac{1}{4}\beta s^2\overline{\theta}_B - \frac{3}{16}\beta s^2}{1 + \frac{1}{3}\beta s^2\overline{\theta}_B - \frac{1}{4}\beta s^2} \tag{2.33}$$

Ainsi, il sera plus facile d'évaluer la fonction aux bornes de $\overline{\theta}_B$: il faut que $\varphi(0)$ se situe au dessus de la bissectrice ete que $\varphi\left(\frac{1}{2}\right)$ soit sous la bissectrice afin d'assurer un croisement avec celle-ci. Ceci sera assuré en déterminant les valeurs maximales et minimales des paramètres β et s dans l'équation 2.33.

Il faut premièrement que $\varphi(0) > 0$

Puisque

$$\varphi(0) = \frac{\frac{1}{2} - \frac{3}{16}\beta s^2}{1 - \frac{1}{4}\beta s^2}$$

Alors il faut résoudre

$$\frac{\frac{1}{2} - \frac{3}{16}\beta s^2}{1 - \frac{1}{4}\beta s^2} > 0$$

Comme β et s sont contraints à être positifs, il faut que le numérateur et le dénominateur soient positifs :

Numérateur :

$$\begin{aligned} \frac{1}{2} - \frac{3}{16}\beta s^2 &> 0 \\ \frac{1}{2} &> \frac{3}{16}\beta s^2 \\ \beta s^2 &< \frac{8}{3} \\ \beta &< \frac{8}{3s^2} \end{aligned}$$

Dénominateur :

$$\begin{aligned} 1 - \frac{1}{4}\beta s^2 &> 0 \\ 1 &> \frac{1}{4}\beta s^2 \\ \beta s^2 &< 4 \\ \beta &< \frac{4}{s^2} \end{aligned}$$

Ainsi, $\beta s^2 < \frac{3}{8} < 4$ assure que $\varphi(0) > 0$

Il faut de plus que $\varphi\left(\frac{1}{2}\right) < \frac{1}{2}$

$$\varphi\left(\frac{1}{2}\right) = \frac{\frac{1}{2} + \frac{1}{8}\beta s^2 - \frac{3}{16}\beta s^2}{1 + \frac{1}{6}\beta s^2 - \frac{1}{4}\beta s^2} = \frac{\frac{1}{2} - \frac{1}{16}\beta s^2}{1 - \frac{1}{12}\beta s^2}$$

$$\begin{aligned}
\frac{\frac{1}{2} - \frac{1}{16}\beta s^2}{1 - \frac{1}{12}\beta s^2} &< \frac{1}{2} \\
\frac{1}{2} - \frac{1}{16}\beta s^2 &< \frac{1}{2} - \frac{1}{24}\beta s^2 \\
\frac{1}{48}\beta s^2 &> 0 \\
\beta s^2 &> 0
\end{aligned}$$

Ainsi, pour assurer une solution de $\overline{\theta}_B$ tel que $0 > \overline{\theta}_B > \frac{1}{2}$, il faut que $0 < \beta s^2 < \frac{8}{3}$.

Or, puisque $s \in (0, 1]$, il s'ensuit que $\beta \in \left(0, \frac{8}{3}\right)$. Comme il n'y avait pas, *a priori*, de restriction sur β, ce n'est pas un problème d'ajouter celle-ci.

Lemme 2 $\overline{\theta}_B$ ***possède une ou deux solutions***

Soit

$$\psi\left(\overline{\theta}_B\right) = \varphi\left(\overline{\theta}_B\right) - \overline{\theta}_B$$

Posons la valeur $\overline{\theta}_B^*$ *telle que* $\psi\left(\overline{\theta}_B^*\right) = 0$

Alors, comme

$\psi\left(\overline{\theta}_B\right)$ *est continue ;*

$\psi(0) > 0$;

$\psi\left(\frac{1}{2}\right) < 0$.

Par le théorème des valeurs intermédiaires

$$\exists\, \overline{\theta}_B \in \left(0, \frac{1}{2}\right]$$

qui satisfait $\overline{\theta}_B^*$ *tant que* $\beta s^2 \in \left(0, \frac{8}{3}\right]$

2.4.2 Courbure et croissance de $\overline{\theta}_B$

L'équation du gain en cas d'échec étant quadratique en $\overline{\theta}_B$, il y aura au plus 2 solutions, soit les 2 racines quadratiques. Graphiquement, il est cependant clair qu'un nombre pair de solutions correspondant à la situation envisagée dans le lemme 2 est impossible lorsque $\overline{\theta}_B > 0$.

Puisque $\varphi(0) \in (0, \frac{1}{2}]$ et que $\varphi\left(\frac{1}{2}\right) \in (0, \frac{1}{2}]$, il ne peut y avoir exactement deux croisements entre $\varphi\left(\overline{\theta}_B\right)$ et la bissectrice. Il doit donc y en avoir un seul $\forall\ \overline{\theta}_B > 0$.

Avant de calculer la valeur de $\overline{\theta}_B(\beta, s)$, vérifions sa croissance et sa courbure.

$$\overline{\theta}_B = \frac{\frac{1}{2} - \frac{3}{16}\beta s^2 + \frac{1}{4}\beta s^2 \overline{\theta}_B}{1 - \frac{1}{4}\beta s^2 + \frac{1}{3}\beta s^2 \overline{\theta}_B} \tag{2.34}$$

L'équation 2.34, égalisée à zéro, peut être réécrite comme

$$0 = \frac{1}{3}\beta s^2 \overline{\theta}_B^2 + \overline{\theta}_B\left(1 - \frac{1}{2}\beta s^2\right) - \frac{1}{2} + \frac{3}{16}\beta s^2 \tag{2.35}$$

La dérivée première est

$$= \frac{2}{3}\beta s^2 \overline{\theta}_B + 1 - \frac{1}{2}\beta s^2 \tag{2.36}$$

La dérivée seconde est

$$= \frac{2}{3}\beta s^2 \tag{2.37}$$

Une évaluation numérique de l'équation 2.36 telle que les précédentes restrictions sur les paramètres sont respectées montre que son résultat est compris entre 0.56667 et 1 ; la fonction est donc strictement croissante.

La dérivée seconde, soit l'équation 2.37, est pour sa part strictement négative puisque les deux paramètres β et s sont compris entre zéro et un. La fonction est donc strictement concave par rapport à l'origine.

2.4.3 Évaluation de $\overline{\theta}_B$

L'équation 2.35 étant quadratique en $\overline{\theta}_B$, et l'existence d'une solution satisfaisante pour $\overline{\theta}_B$ étant démontrée, elle peut être résolue en déterminant ses racines, obtenues par

$$\overline{\theta}_B = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

où

$$a = \tfrac{1}{3}\beta s^2$$
$$b = 1 - \tfrac{1}{2}\beta s^2$$
$$c = \tfrac{3}{16}\beta s^2 - \tfrac{1}{2}$$

Alors les racines de l'équation sont

$$\overline{\theta}_B = \frac{\frac{1}{2}\beta s^2 - 1 \pm \sqrt{1 - \frac{1}{3}\beta s^2}}{\frac{2}{3}\beta s^2} \tag{2.38}$$

Comme les valeurs recherchées pour $\overline{\theta}_B$ doivent être comprises entre 0 et $\frac{1}{2}$, alors la racine

$$\overline{\theta}_B = \frac{\frac{1}{2}\beta s^2 - 1 \pm \sqrt{1 - \frac{1}{3}\beta s^2}}{\frac{2}{3}\beta s^2} \tag{2.39}$$

peut être rejetée puisqu'elle donne des résultats négatifs.

Ainsi

$$\overline{\theta}_B^* = \frac{\frac{1}{2}\beta s^2 - 1 + \sqrt{1 - \frac{1}{3}\beta s^2}}{\frac{2}{3}\beta s^2} \tag{2.40}$$

Les fonctions de gain en cas d'échec ou de non-réussite et en cas de succès d'un projet permettent de déterminer la fonction de meilleure réponse *ex post*, soit l'effort optimal fourni *ex ante* par un programmeur selon son type (sa compétence, θ) et son gain espéré est

$$e^*\left(\theta; \overline{\theta}_G, \overline{\theta}_B\right) = \beta s\theta\left(\overline{\theta}_G - \overline{\theta}_B\right) \tag{2.41}$$

Une fois les croyances insérées, l'effort obtimal devient

$$e^*(\theta) = \beta s\theta\left(\frac{3}{4} - \frac{\frac{1}{2}\beta s^2 - 1 + \sqrt{1 - \frac{1}{3}\beta s^2}}{\frac{2}{3}\beta s^2}\right) \tag{2.42}$$

$$= \theta\left(\frac{3 - 3\sqrt{1 - \frac{1}{3}\beta s^2}}{2s}\right) \tag{2.43}$$

2.5 Équilibre

Tel qu'il a été défini dans la section 2.4, un équilibre comporte un ensemble de stratégies, un gain en cas d'échec et un gain en cas de succès. Une fois les valeurs connues, cet équilibre s'écrit

$$Eq : \left\{ e^*(\theta) = \theta \left(\frac{3 - 3\sqrt{1 - \frac{1}{3}\beta s^2}}{2s} \right) , \overline{\theta}_G^* = \frac{3}{4} , \overline{\theta}_B^* = \frac{\frac{1}{2}\beta s^2 - 1 + \sqrt{1 - \frac{1}{3}\beta s^2}}{\frac{2}{3}\beta s^2} \right\} \quad (2.44)$$

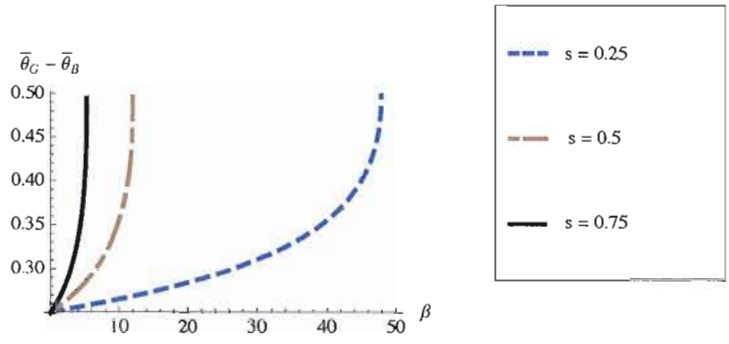
Cet équilibre dépend des divers paramètres exogènes qui sont analysés à la section 2.6. Il est cohérent et séquentiellement rationnel, puisqu'il a été résolu à l'aide de la règle de Bayes et qu'il respecte les hypothèses posées sur les divers paramètres et variables. On peut l'interpréter ainsi : sachant qu'un programmeur a un talent θ, celui-ci fournira un effort $e^*(\theta)$ et verra sa réputation majorée de $\overline{\theta}_B$ si son projet n'aboutit pas ou qu'il ne se distingue pas de la masse, et de $\overline{\theta}_G$ si son projet obtient un succès.

2.6 Statiques comparatives

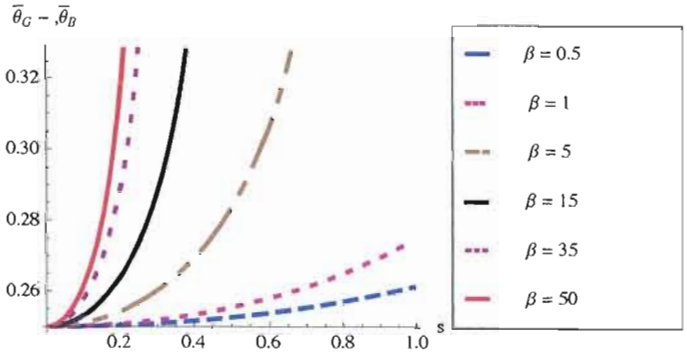
L'équilibre est maintenant défini, mais étant donné la forme de ses composantes, il n'est pas du tout évident de prédire ce qui se produit dans différentes situations. Par exemple, est-ce qu'une augmentation de l'intérêt des programmeurs envers leur réputation aura une incidence sur l'effort optimal à fournir dans un projet ? *A priori*, il semble évident que c'est le cas. Mais étant donné la forme des équations de gain en cas d'échec et d'effort optimal, il faut le vérifier graphiquement.

2.6.1 Le gain en cas d'échec ou de non-réussite

Une première question à analyser à l'aide du modèle est l'impact des deux paramètres sur le gain anticipé : est-il plus payant de programmer en présence d'une motivation



(a) Impact de β



(b) Impact de s

Figure 2.4 : Impact des paramètres β et θ sur $\left(\overline{\theta}_G - \overline{\theta}_B\right)$ selon différentes valeurs pour l'autre paramètre

réputationnelle forte ? Est-ce que l'intérêt de la communauté Open Source (la visibilité d'un projet) a un rôle important dans les gains réputationnels des programmeurs ?

Les graphiques de la figure 2.4 illustrent les valeurs possibles du gain en termes de réputation, en cas d'échec ou de non-succès. Le graphique 2.4a montre l'impact de l'intérêt des programmeurs face à leur réputation pour différentes probabilités de se distinguer parmi leurs pairs, alors que l'inverse est présenté dans le graphique 2.4b. On peut constater que plus ces paramètres augmentent, plus le gain en cas de non-succès sera faible. De plus, on remarque que cette diminution se produit d'autant plus

rapidement que les paramètres sont élevés, ensemble ou séparément, mais que cette diminution est de faible magnitude.

L'intérêt de présenter le gain en cas de succès relativement au gain en cas d'insuccès tient du fait que $\overline{\theta}_G$ est constant $\left(\overline{\theta}_G = \frac{3}{4}\right)$. Puisqu'un succès procure toujours le même gain réputationnel, pour tout programmeur, il est intéressant de vérifier si l'importance relative de ce gain a un impact sur les programmeurs.

Une première analyse de la figure 2.4a révèle que β a le même effet sur le gain relatif en cas de succès, c'est-à-dire que plus l'intérêt des programmeurs vis-à-vis leur réputation est élevé, plus la « prime » réputationnelle qu'ils reçoivent suivant un succès sera importante. Alors que la valeur maximale qu'il est possible d'atteindre ne semble pas varier (toutes les courbes du graphique 2.4a se rendent à 0.5), l'intérêt de la communauté Open Source envers un projet a un effet accélérateur sur cette prime ; plus s est élevé, moins l'intérêt des programmeurs a besoin d'être élevé pour atteindre la valeur maximale de $\left(\overline{\theta}_G - \overline{\theta}_B\right)$. En fait, l'intérêt des programmeurs est en quelque sorte limité ; pour toute valeur de s, il y a une borne maximale à l'intérêt que les programmeurs peuvent présenter. Si β dépasse un certain seuil (voir la section 2.4.1), il n'y a plus aucun gain possible ; ainsi, un programmeur qui manifeste un intérêt trop grand envers sa propre réputation doit choisir un projet qui ne suscite qu'un faible intérêt de la part de la communauté Open Source. En fait, comme il a été mentionné, le gain en cas de succès est fixe, donc ne changera pas selon l'intérêt du programmeur, mais son gain relativement à ce qu'il aurait eu suivant un échec devient pratiquement inexistant[3].

Le graphique 2.4b présente la même situation, mais présentée de « l'autre » point de vue : un accroissement de la probabilité de distinction (s) a un impact semblable à un accroissement de l'intérêt des programmeurs, mais qui se fait ressentir plus rapidement. S'il est plus facile d'atteindre la visibilité lorsqu'un projet est réussi, il

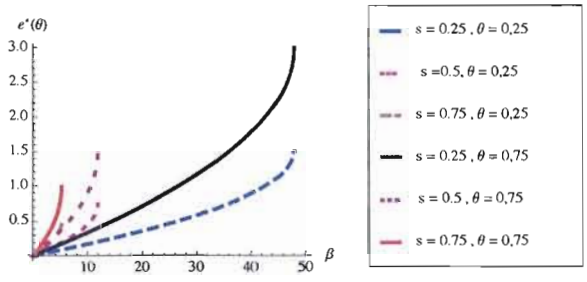
3. En fait, le gain en cas d'échec se retrouve dans les nombres imaginaires.

semble logique de croire qu'un échec ou une non-distinction rapporte encore moins en termes de réputation, et ce, de manière plus prononcée. Le contraire est aussi vrai : le gain en cas de réussite (relativement à $\overline{\theta}_B$) augmente d'autant plus fortement lorsque l'intérêt des programmeurs est élevé. On retrouve donc la même situation que dans le graphique 2.4a : un intérêt élevé de la part des programmeurs limite l'intérêt que la communauté peut offrir. Encore une fois, le gain en cas de succès ne changera pas, mais il perd toute valeur en comparaison du gain en cas d'échec.

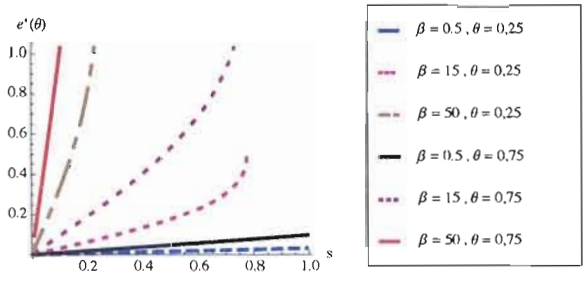
En somme, on remarque que (1) un accroissement de l'intérêt des programmeurs pour la communauté Open Source et que (2) une augmentation de la probabilité de se distinguer advenant une réussite, ensemble ou séparément, font diminuer le gain en cas de réussite relativement au gain en cas d'échec. L'interprétation de ce phénomène est relativement simple : si « l'intérêt » est trop élevé (tant de la part du programmeur envers sa réputation que de la communauté envers un projet), il y aura fort probablement plus de programmeurs qui souhaiteront y participer : c'est une extrapolation de ce que Tirole et Lerner (2002) ont appelé « complémentarités stratégiques » : le succès attire le succès. Donc, si plus de gens participent, la contribution individuelle diminue, réduisant ainsi le gain en cas de succès relativement au gain en cas d'échec.

2.6.2 L'effort optimal

Il faut aussi vérifier ce qui se produit avec l'effort que fournissent les programmeurs. Est-ce que l'intérêt de part ou d'autre favorise l'effort? On peut ainsi déterminer, à l'aide de la fonction d'effort optimal *ex post*, que le type du programmeur (θ) fait varier linéairement l'effort fourni. Ceci est représenté à la figure 2.5 : on peut remarquer, dans les deux graphiques, que la forme des courbes ne change pas selon le type du programmeur. Cependant, pour un même intérêt des programmeur et de la communauté, l'effort fourni quand le talent est élevé est supérieur au niveau d'effort fourni pour un talent moindre. Ainsi, le modèle suggère que les programmeurs talentueux fourniront un plus grand niveau d'efforts.



(a) Impact de β



(b) Impact de s

Figure 2.5 : Impact des différents paramètres sur l'effort optimal selon différentes valeurs pour les autres paramètres

Le graphique 2.5a montre un résultat attendu : plus l'intérêt des programmeurs envers leur réputation est élevé, plus leur niveau d'efforts optimal sera élevé. Cependant, il faut noter, dans ce même graphique, la diminution de l'effort *maximal* fourni lorsque la probabilité de se distinguer ou le talent sont élevés. Autrement dit, l'intérêt de la communauté limite l'intérêt que les programmeurs peuvent avoir : à talent égal, si s est élevé, β ne peut dépasser une certaine valeur, sans quoi l'effort optimal devient « imaginaire ».

Dans le graphique 2.5b, une situation similaire est constatée : un intérêt accru de la part de la communauté Open Source amène un niveau d'effort optimal supérieur. L'effort maximal croît de plus en plus rapidement lorsque β est élevé. Cependant, comme dans la situation précédente, un intérêt élevé de la part de la communauté Open Source limite l'intérêt que le programmeurs peuvent présenter envers leur propre réputation.

Ceci suggère, par exemple, que lorsque la communauté est très intérressée par un projet, elle encouragera un plus grand nombre de programmeurs à y participer, diminuant ainsi l'effort optimal maximal que chacun doit fournir. Une autre explication réside dans les intérêts en quelque sorte divergents des programmeurs et de la communauté Open Source : les programmeurs recherchent leur propre gain tandis que la communauté recherche un « meilleur » projet ou logiciel. Il est donc souhaitable, du point de vue de la communauté, que les programmeurs désirant très fortement une meilleure réputation (β élevé) participent à des projets auxquels la communauté s'intéresse moins.

Néanmoins, ces deux paramètres s'interprètent, dans ce modèle, de manière similaire : si un programmeur a plus d'intérêt envers sa réputation, il désirera plus ardemment réussir son projet ; il fournira plus d'efforts. De même, si un projet réussi a plus de chances de se distinguer parmi les autres projets du même acabit, le programmeur fournira plus d'efforts afin d'empocher le gain associé à son éventuel succès. Il faut donc tenir pour acquis que seulement les plus acharnés, les plus talentueux ou les plus intéressés par leur réputation obtiendront un succès. C'est ce qui explique que dans ce paradigme, il est attendu que les programmeurs qui obtiennent un succès ont un grand talent : $\overline{\theta}_G$ le gain en cas de succès, est la densité conditionnelle du talent sachant un succès. Elle est égale à $\frac{3}{4}$, donc les programmeurs se voient décerner une réputation, suivant un succès, de $\frac{3}{4}$.

CHAPITRE III

APPLICATION DU MODÈLE DE BASE

Puisqu'il existe des dizaines de licences libres, une seconde licence est ajoutée au modèle du chapitre précédent afin de refléter plus fidèlement la réalité. Le but de cet ajout est de créer une « auto-sélection » des programmeurs. Ceux-ci on maintenant à choisir un type de projet lorsqu'ils se joignent à la communauté Open Source. Il a été expliqué précedemment que les programmeurs, peu importe leur motivation personelle, participent à un projet Open Source dans le but d'obtenir un gain réputationnel. Dorénavant, le choix du projet permettra aux programmeurs de choisir un projet correspondant à leur talent afin d'optimiser le signal qu'ils envoient. Par exemple, il semble logique de croire qu'un programmeur de talent faible aurait intérêt à participer à un projet facile.

L'ajout d'une seconde classe de projet se fait à l'aide des licences libres. Dans le modèle précédent, la seule licence retenue était la GPL étant donné son caractère viral, qui fait que le signal que les programmeurs envoient est en quelque sorte amplifié. Ici, toujours en considérant que les programmeurs visent à avoir une meilleure réputation, leurs motivations personnelles seront prises en compte. La licence BSD n'a pas le caractère viral de la GPL, de sorte qu'il est beaucoup plus difficile pour un projet assujetti à la licence BSD de se faire remarquer, comme il a été expliqué à la section 1.2. Ainsi, pour un programmeur de faible talent, mais qui souhaite néanmoins une meilleure réputation, il sera plus intéressant de participer à un projet assujetti à la licence BSD, alors qu'un programmeur de grand talent préférera sans doutes la licence GPL.

Puisque les programmeurs sont différenciés à l'aide de leur talent, les projets assujettis aux différentes licences pourront en plus être qualifiés de « faciles » ou de « difficiles », selon le talent qu'un programmeur doit avoir pour le mener à bien. La notation du modèle est adaptée pour refléter la présence du second projet : les gains seront dorénavant dénotés $\overline{\theta}_B$ pour le gain en cas d'échec ou d'insuccès, $\overline{\theta}_{G_1}$ pour le gain en cas de succès du projet avec la licence BSD (facile) et $\overline{\theta}_{G_2}$ pour le gain en cas de succès du projet asujetti à la licence GPL (dificile). Par souci de cohérence, une restriction (résumée dans l'équation 3.1) est imposée aux gains réputationnels.

$$\overline{\theta}_B < \overline{\theta}_{G_1} < \overline{\theta}_{G_2} \tag{3.1}$$

Ainsi, le projet facile sera choisi par les programmeurs pour qui le gain anticipé dans le projet facile est supérieur au gain anticipé dans le projet difficile. Il est attendu que ce soit ceux ayant un talent (θ) faible.

3.1 Déroulement du jeu

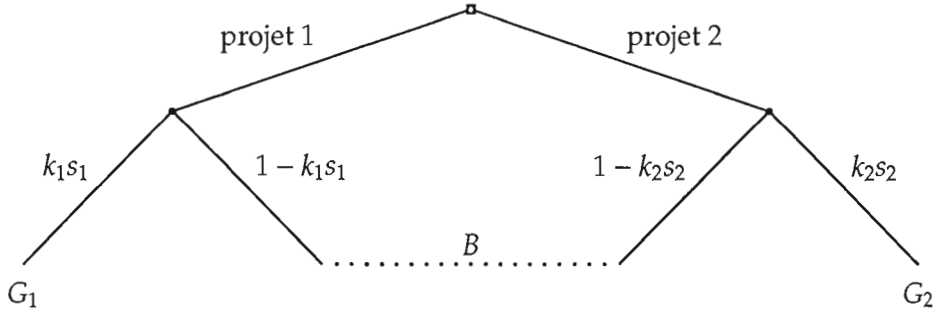


Figure 3.1 : Déroulement dans le cas multilicences

La figure 3.1 illustre la manière dont les choses se déroulent, tel qu'il a été décrit ci-dessus. La première étape du jeu est appelée *réussite* (ou *échec*), et servira à montrer si un programmeur complète son projet d'une manière satisfaisante ou non. La seconde étape est la *distinction*. Indépendante de la réussite, la distinction indiquera si un programmeur qui a réussi son projet parvient à se distinguer des autres programmeurs

ou projets. Pour qu'un programmeur obtienne un gain en termes de réputation, il devra obtenir un *succès*, soit le cumul d'une réussite et d'une distinction. Les paramètres utilisés ici diffèrent de ceux du modèle de base :

$i \in \{1, 2\}$: Nature des projets
- $i = 1$: Projet facile, licence BSD
- $i = 2$: Projet difficile, licence GPL

k_i : Probabilité de réussite ; $k_1 > k_2$
- $i = 1$: réussite facile
- $i = 2$: réussite malaisée

s_i : Probabilité de distinction ; $s_1 < s_2$
- $i = 1$: distinction malaisée
- $i = 2$: distinction facile

La probabilité de réussite d'un projet est semblable à celle du modèle précédent : elle dépend linéairement de l'effort et du talent d'un programmeur.

$$p(e, \theta) = \theta e \tag{3.2}$$

Ce qui change ici est la probabilité de succès, puisqu'elle dépend désormais du projet entrepris. En intégrant les restrictions posées ci-dessus sur les différents paramètres, les probabilité de succès peuvent être définis comme suit :

$$p[G_1] = s_1 k_1 \cdot p(e_1, \theta) \tag{3.3}$$

$$p[G_2] = s_2 k_2 \cdot p(e_1, \theta) \tag{3.4}$$

On peut remarquer que la probabilité de réussite varie selon le projet envisagé et qu'elle sera croissante dans chacun de ses arguments. Le talent initial et l'effort fourni,

comme dans le modèle de base, font augmenter les chances de succès. Les différentes probabilités (réussite, visibilité) aussi, mais il ne faut pas oublier qu'elles sont opposées l'une à l'autre. Tel qu'il a été défini ci-dessus, lorsqu'une de ces probabilités est élevée, l'autre est faible.

Il est aussi important de noter que les deux projets sont mutuellement exclusifs. Ceci implique que le choix d'une catégorie de programme annule le gain potentiel lié à l'autre projet. Si un programmeur participe à un projet assujetti à une licence BSD, son gain anticipé (dénoté $\widetilde{\theta}_S$ comme au chapitre précédent) est $\widetilde{\theta}_S \in \{\overline{\theta}_{G_1}, \overline{\theta}_B\}$, tandis que s'il participe au projet assujetti à la GPL, son gain anticipé sera plutôt $\widetilde{\theta}_S \in \{\overline{\theta}_{G_2}, \overline{\theta}_B\}$. En bref, une fois qu'un programmeur a décidé de participer à un projet en particulier, l'espérance de gain en cas de succès lié à l'autre classe de projet est nulle. Les gains anticipés sont donc définis comme suit :

$$E\left[\widetilde{\theta}_S \mid \text{ le projet } i\right] = E\left(\overline{\theta}_{G_i}\right) + E\left(\overline{\theta}_B\right) \qquad i \in 1,2 \tag{3.5}$$

$$= p(e_i, \theta)\overline{\theta}_{G_i} + (1 - p(e_i, \theta))\,\overline{\theta}_B \qquad i \in 1,2 \tag{3.6}$$

$$= k_i s_i \theta e_i \overline{\theta}_{G_1} + (1 - k_i s_i \theta e_i)\,\overline{\theta}_B \qquad i \in 1,2 \tag{3.7}$$

3.2 Fonction d'utilité du programmeur

La même fonction d'utilité pour les programmeurs que dans le modèle précédant est employée :

$$U = \mathscr{Y} - c(e) + \beta\theta_S \tag{3.8}$$

On retrouve encore la richesse initiale, le coût lié à l'effort d'apprentissage d'un projet, ainsi que le paramètre de retour sur l'effort (β). Cependant, les gains en termes de réputation sont définis autrement, afin de respecter la nouvelle représentation du jeu.

Le même coût de participation lié à l'effort est utilisé, afin d'éliminer des sources de divergence ; il sera ainsi plus aisé de comparer les résultats des deux modèles.

$$c(e) = \frac{1}{2}e^2 \tag{3.9}$$

On peut alors, en substituant les équations précédentes dans l'équation 3.2, déterminer la fonction d'utilité anticipée des programmeurs participant à un projet Open Source :

$$E[U] = \mathscr{Y} - c(e) + \beta E\left[\widetilde{\theta}_S\right] \tag{3.10}$$

$$= \mathscr{Y} - \frac{1}{2}e^2 + \beta\left[k_i s_i \cdot p(e,\theta)\overline{\theta}_{G_i} + (1 - k_i s_i \cdot p(e,\theta))\,\overline{\theta}_B\right] \qquad i \in \{1,2\} \tag{3.11}$$

$$= \mathscr{Y} - \frac{1}{2}e^2 + \beta\left[k_i s_i \cdot p(e,\theta)\,\overline{\theta}_{G_1} + (1 - s_i k_i \cdot p(e,\theta))\,\overline{\theta}_B\right] \qquad i \in \{1,2\} \tag{3.12}$$

$$\overline{U}(e,i) = \mathscr{Y} - \frac{1}{2}e^2 + \beta\overline{\theta}_B + \beta k_i s_i \theta e\left(\overline{\theta}_{G_i} - \overline{\theta}_B\right) \qquad i \in \{1,2\} \tag{3.13}$$

Les gains anticipés en termes de réputation sont, comme dans le modèle précédent, l'espérance du talent d'un programmeur conditionelle au succès obtenu :

$$\overline{\theta}_{G_1} = E[\theta|G_1] \tag{3.14}$$

$$\overline{\theta}_{G_2} = E[\theta|G_2] \tag{3.15}$$

3.3 Effort optimal

L'effort optimal que fournit un programmeur participant à un projet Open Source est toujours donné par sa stratégie de meilleure réponse pour un programmeur maximisant son utilité :

$$\frac{\partial \overline{U}}{\partial e_i} = 0 \tag{3.16}$$

$$= \begin{cases} \widehat{e_1}(\theta, \overline{\theta}_{G_1}, \overline{\theta}_B) = \beta\theta k_1 s_1 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) \\ \widehat{e_2}(\theta, \overline{\theta}_{G_2}, \overline{\theta}_B) = \beta\theta k_2 s_2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) \end{cases} \tag{3.17}$$

Les probabilités de succès deviennent donc :

$$p\left(\widehat{e_1}\left(\theta, \overline{\theta}_{G_1}, \overline{\theta}_B\right), \theta\right) = \theta \cdot \widehat{e_1}\left(\theta, \overline{\theta}_{G_1}, \overline{\theta}_B\right) \tag{3.18}$$

$$= \beta\theta^2 k_1 s_1 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) \tag{3.19}$$

$$p\left(\widehat{e_2}(\theta, \overline{\theta}_{G_2}, \overline{\theta}_B), \theta\right) = \theta \cdot \widehat{e_2}\left(\theta, \overline{\theta}_{G_2}, \overline{\theta}_B\right) \tag{3.20}$$

$$= \beta\theta^2 k_2 s_2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) \tag{3.21}$$

L'effort optimal est toujours modulé par le talent du programmeur, ainsi que les probabilités de réussite ou d'échec. Cependant, l'effort est aussi déterminé par le projet choisi : les paramètres de succès et de visibilité font partie de la fonction d'effort. La valeur relative du gain en cas de succès, par rapport au gain en cas d'échec, y est aussi toujours présente. Pour assurer la cohérence du modèle, il est nécéssaire que $\left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) > \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)$.

3.4 Détermination de l'équilibre

Dans ce modèle, l'équilibre est toujours un ensemble de stratégies, qui servira à déterminer l'effort optimal de chaque programmeur. Cet effort sera déterminé en fonction des gains anticipés, selon le projet choisit. Ces gains, composés du gain anticipé en cas d'échec et du gain anticipé en cas de succès, sont déterminés de manière exogène pour le programmeur, mais de manière endogène par rapport au modèle.

Un problème supplémentaire survient ici : l'auto-sélection des programmeurs quant au choix de projet à entreprendre. Il est, à ce stade, souhaité que les programmeurs de talent faible choisissent un projet assujetti à la licence BSD et que que les programmeurs au talent élevé, un projet avec une licence GPL. Or, puisque $\overline{\theta}_{G_2}$ est supérieur à $\overline{\theta}_{G_1}$, il n'est pas clair que ce sera le cas. Il est possible que tous les programmeurs choisissent le projet difficile. Il faut introduire un moyen de déterminer comment les programmeurs choisiront le projet auquel ils participeront. Chaque programmeur qui choisit de s'impliquer dans un projet Open Source a deux actions, ou choix, à faire. Soit (d, e) ces actions, où $d \in \{1, 2\}$ est le choix de la classe de projet et e l'effort qu'il y consacrera. Un équilibre se caractérise donc par

$$Eq = \left\{d^*(\theta), e_1^*(\theta), e_2^*(\theta), \overline{\theta}_{G_1}, \overline{\theta}_{G_2}, \overline{\theta}_B\right\} \tag{3.22}$$

Le programmeur, qui connaît son propre talent $\overline{\theta}$, choisira le projet auquel il participe en fonction de ce talent. Afin de situer chaque programmeur par rapport au talent requis pour chacune des classes de projet, il faudra comparer ce talent personnel θ à une valeur critique, $\widehat{\theta}$, qui représente le niveau de talent pour lequel un programmeur est tout juste indifférent entre participer à un projet facile ou un projet difficile. Puisque le talent est toujours $\theta \in [0, 1]$, cette situation est représentée graphiquement à la figure 3.2.

Dans les équilibres considérés ici, il devient clair que $d^*(\theta) = 1$ si $\theta < \widehat{\theta}$ et que $d^*(\theta) = 2$ si $\theta > \widehat{\theta}$. La valeur critique $\widehat{\theta}$ définit donc complètement la stratégie de choix du

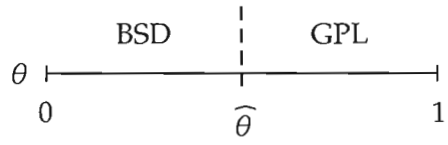


Figure 3.2 : Représentation du seuil critique et de l'auto-sélection des projets par les programmeurs

projet par un programmeur. Un équilibre, dans ce modèle, sera donc écrit de la manière suivante :

$$Eq = \left\{ \widehat{\theta}, e_1^*(\theta), e_2^*(\theta), \overline{\theta}_{G_1}, \overline{\theta}_{G_2}, \overline{\theta}_B \right\} \tag{3.23}$$

Les croyances sont rationelles étant donné les ensembles de stratégies, et sont obtenues à l'aide de la règle de Bayes. Les ensembles de stratégies sont, quant à eux, rationnels par rapport aux croyances. La résolution du modèle à deux licences est effectuée essentiellement comme celle du modèle de base. La présence d'une deuxième licence rend les équations plus lourdes, mais n'influence la résolution que dans l'ajout d'une étape : plutôt que de considérer le succès dans son ensemble, il faut maintenant considérer le succès d'un projet facile, ensuite celui d'un projet difficile.

La probabilité d'un échec (soit la probabilité de se retrouver dans l'événement B) est maintenant la probabilité de ne se retrouver ni dans G_1 ni dans G_2, ce qui donne un ensemble d'informations pour B. Cet ensemble d'informations se retrouve en quelque sorte divisé en deux parties autour de $\widehat{\theta}$, le talent minimal requis pour réussir un projet difficile. La résolution du modèle se trouve donc complexifiée par ce fait.

La résolution des gains en cas d'échec s'effectue de manière semblable à ce qui a été fait dans le chapitre précédent. Il faut d'abord déterminer la probabilité de succès dans la population, mais cette fois en supposant l'auto-sélection réussie ; autrement dit, les

probabilités de succès pour les deux projets sont calculées selon la répartition du talent telle qu'elle est illustrée à la figure 3.2.

Les éléments de l'équilibre peuvent être représentés ainsi :

$$e_1^*(\theta) = \widehat{e_1}\left(\theta, \overline{\theta}_{G_1}^*, \overline{\theta}_B^*\right) \tag{3.24}$$

$$e_2^*(\theta) = \widehat{e_2}\left(\theta, \overline{\theta}_{G_2}^*, \overline{\theta}_B^*\right) \tag{3.25}$$

$$\overline{\theta}_{G_1}^* = \int_0^{\widehat{\theta}} \theta \cdot f(\theta|G_1)\, d\theta \tag{3.26}$$

$$\overline{\theta}_{G_2}^* = \int_{\widehat{\theta}}^1 \theta \cdot f(\theta|G_2)\, d\theta \tag{3.27}$$

$$\overline{\theta}_B^* = \int_0^1 \theta \cdot f(\theta|B)\, d\theta \tag{3.28}$$

La résolution des deux gains en cas de succès $\left(\overline{\theta}_{G_1}^*, \overline{\theta}_{G_2}^*\right)$ s'effectue de la même manière, soit sensiblement comme le gain en cas de succès du modèle précédent. Il faut premièrement déterminer la probabilité du succès dans la population, selon l'effort optimal dans le projet concerné. La probabilité d'échec, comme il a été mentionné ci-dessus, est donc la probabilité de ne se retrouver ni dans G_1, ni dans G_2 :

$$p[G_1] = \int_0^{\widehat{\theta}} k_1 s_1 \cdot p\left(e_1^*(\theta), \theta\right) f(\theta) d\theta \tag{3.29}$$

$$= \int_0^{\widehat{\theta}} \beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) f(\theta) d\theta \tag{3.30}$$

$$p[G_2] = \int_{\widehat{\theta}}^1 k_2 s_2 \cdot p\left(e_2^*(\theta), \theta\right) f(\theta) d\theta \tag{3.31}$$

$$= \int_{\widehat{\theta}}^1 \beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) f(\theta) d\theta \tag{3.32}$$

$$p[B] = \int_0^{\widehat{\theta}} \left[1 - k_1 s_1 \cdot p\left(e_1^*(\theta), \theta\right) f(\theta)\right] d\theta + \int_{\widehat{\theta}}^1 \left[1 - k_2 s_2 \cdot p\left(e_2^*(\theta), \theta\right) f(\theta)\right] d\theta \quad (3.33)$$

$$= \int_0^{\widehat{\theta}} \left[1 - \beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) f(\theta)\right] d\theta + \int_{\widehat{\theta}}^1 \left[1 - \beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) f(\theta)\right] d\theta \quad (3.34)$$

Ces équations permettent de déterminer les fonctions de probabilités conditionelles du talent sachant la réalisation d'un événement, à l'aide de la règle de Bayes :

$$f(\theta|G_1) = \frac{k_1 s_1 \cdot p\left(e_1^*(\theta), \theta\right)}{p[G_1]} \quad (3.35)$$

$$= \frac{\beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)}{p[G]_1} \quad (3.36)$$

$$f(\theta|G_2) = \frac{k_2 s_2 \cdot p\left(e_2^*(\theta), \theta\right)}{p[G_2]} \quad (3.37)$$

$$= \frac{\beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)}{p[G_2]} \quad (3.38)$$

$$f(\theta|B) = \begin{cases} f(\theta|B)_{\text{II}} : \dfrac{1 - k_1 s_1 \cdot p\left(e_1^*(\theta), \theta\right)}{p[B]} \\ f(\theta|B)_{\text{III}} : \dfrac{1 - k_2 s_2 \cdot p\left(e_2^*(\theta), \theta\right)}{p[B]} \end{cases} \quad (3.39)$$

$$= \begin{cases} f(\theta|B)_{\text{II}} : \dfrac{1 - \beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)}{p[B]} \\ f(\theta|B)_{\text{III}} : \dfrac{1 - \beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)}{p[B]} \end{cases} \quad (3.40)$$

Enfin, les gains réputationnels sont déterminés en appliquant cette fonction de probabilité conditionelle à la population :

$$\overline{\theta}_{G_1} = \int_0^{\widehat{\theta}} \theta \cdot f(\theta|G_1)d\theta \tag{3.41}$$

$$\overline{\theta}_{G_2} = \int_{\widehat{\theta}}^1 \theta \cdot f(\theta|G_2)d\theta \tag{3.42}$$

$$\overline{\theta}_B = \int_0^{\widehat{\theta}} \theta \cdot f(\theta|B)_{\text{I}}d\theta + \int_{\widehat{\theta}}^1 \theta \cdot f(\theta|B)_{\text{III}}d\theta \tag{3.43}$$

Il est clair que, comme dans le modèle précédent, les gains en cas de succès sont déterminés par le type du programmeur, la probabilité de succès et la « prime » du gain en cas de succès par rapport au gain en cas d'échec. La résolution des équations 3.41, 3.42 et 3.43 sont détaillées à la section B, et donnent les résultats suivants :

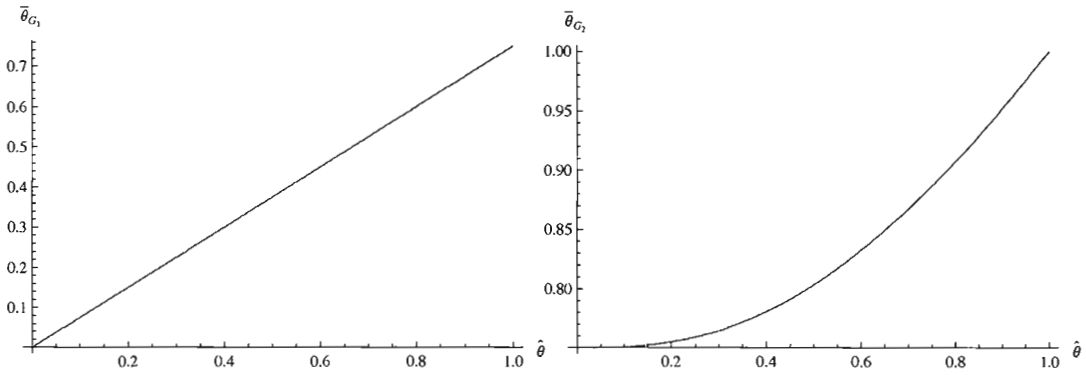
$$\overline{\theta}^*_{G_1} = \frac{3}{4}\widehat{\theta} \tag{3.44}$$

$$\overline{\theta}^*_{G_2} = \frac{3\left(1 - \widehat{\theta}^4\right)}{4\left(1 - \widehat{\theta}^3\right)} \tag{3.45}$$

$$\overline{\theta}_B = \frac{\frac{1}{2} - \frac{1}{4}\beta\left[\widehat{\theta}^4 s_1^2 k_1^2\left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) + \left(1 - \widehat{\theta}^4\right)s_2^2 k_2^2\left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right]}{1 - \frac{1}{3}\beta\left[\widehat{\theta}^3 k_1^2 s_1^2\left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) + (1 - \widehat{\theta}^3)k_2^2 s_2^2\left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right]} \tag{3.46}$$

Les valeurs potentielles des gains en cas de succès sont illustrées à la figure 3.3. On peut y remarquer que ces gains sont clairement « partitionnés » à 0.75 : un succès dans un projet facile ne peut rapporter plus de cette valeur, tandis qu'un succès dans un projet difficile ne peut, lui, rapporter moins de cette valeur à un programmeur.

Cette situation est cohérente avec les hypothèses concernant ces deux valeurs : $\overline{\theta}_{G_2} > \overline{\theta}_{G_1}$. À l'équilibre, il n'y aura qu'une partie de ces valeurs qui seront retenues; les graphiques de la figure 3.3 ont été réalisés en permettant au seuil critique entre les projets $\left(\widehat{\theta}\right)$ de varier entre zéro et un. Or, sa valeur sera limitée et, à ce stade, il est



(a) Valeur de $\overline{\theta}_{G_1}$ (b) Valeur de $\overline{\theta}_{G_2}$

Figure 3.3 : Paiement en termes de réputation
en cas de succès

permis de croire sa valeur autour de la moyenne de la population puisque le talent est réparti uniformément.

3.4.1 Preuve de l'existence de solutions pour $\overline{\theta}_B$

Puisque $\overline{\theta}_{G_1}$ et $\overline{\theta}_{G_2}$ sont connus, une fois l'équation pour $\overline{\theta}_B$ trouvée, ces deux gains seront remplacés, dans l'équation pour $\overline{\theta}_B$, par leur expression respective telles que déterminées ci-dessus :

$$\overline{\theta}_B = \frac{\frac{1}{2} - \frac{1}{4}\beta\left[\widehat{\theta}^4 s_1^2 k_1^2\left(\frac{3}{4}\widehat{\theta} - \overline{\theta}_B\right) + \left(1 - \widehat{\theta}^4\right) s_2^2 k_2^2\left(\frac{3\left(1-\widehat{\theta}^4\right)}{4\left(1-\widehat{\theta}^3\right)} - \overline{\theta}_B\right)\right]}{1 - \frac{1}{3}\beta\left[\widehat{\theta}^3 s_1^2 k_1^2\left(\frac{3}{4}\widehat{\theta} - \overline{\theta}_B\right) + \left(1 - \widehat{\theta}^3\right) s_2^2 k_2^2\left(\frac{3\left(1-\widehat{\theta}^4\right)}{4\left(1-\widehat{\theta}^3\right)} - \overline{\theta}_B\right)\right]} \tag{3.47}$$

Dans un premier temps, il faut déterminer la borne supérieure pour $\overline{\theta}_B$. Il n'est pas nécessaire de chercher une borne inférieure ; le gain en cas d'échec doit être positif puisque le talent se situe entre zéro et un.

Lemme 3 : *$\overline{\theta}_B$ est inférieur à la moyenne de la population, à $\overline{\theta}_{G_1}$ et $\overline{\theta}_{G_2}$*

Puisque la distribution du talent est uniforme, et que la somme des gains anticipés est égale à la cumulative du talent,

$$\overline{\theta} = \frac{1}{2} = p\left(\overline{\theta}_{G_1}\right)\overline{\theta}_{G_1} + p\left(\overline{\theta}_{G_2}\right)\overline{\theta}_{G_2} + p\left(\overline{\theta}_B\right)\overline{\theta}_B$$

il s'ensuit que

$$p\left(\overline{\theta}_B\right)\overline{\theta}_B = \frac{1}{2} - \left[p\left(\overline{\theta}_{G_1}\right)\overline{\theta}_{G_1} + p\left(\overline{\theta}_{G_2}\right)\overline{\theta}_{G_2}\right]$$

donc

$$\overline{\theta}_B < \frac{1}{2}$$

Comme il a été fait précédemment, il faut d'abord vérifier s'il y a au moins une solution pour $\overline{\theta}_B$. Posons d'abord $\varphi\left(\overline{\theta}_B\right)$, qui est égal au membre de droite de l'équation 3.47. La figure 3.4 montre cette équation, ainsi que la bissectrice $\varphi\left(\overline{\theta}_B\right) = \overline{\theta}_B$.

$$\varphi\left(\overline{\theta}_B\right) = \frac{\frac{1}{2} - \frac{1}{4}\beta\left[\widehat{\theta}^4 s_1^2 k_1^2\left(\frac{3}{4}\widehat{\theta} - \overline{\theta}_B\right) + \left(1 - \widehat{\theta}^4\right)s_2^2 k_2^2\left(\frac{3\left(1-\widehat{\theta}^4\right)}{4\left(1-\widehat{\theta}^3\right)} - \overline{\theta}_B\right)\right]}{1 - \frac{1}{3}\beta\left[\widehat{\theta}^3 s_1^2 k_1^2\left(\frac{3}{4}\widehat{\theta} - \overline{\theta}_B\right) + \left(1 - \widehat{\theta}^3\right)s_2^2 k_2^2\left(\frac{3\left(1-\widehat{\theta}^4\right)}{4\left(1-\widehat{\theta}^3\right)} - \overline{\theta}_B\right)\right]} \tag{3.48}$$

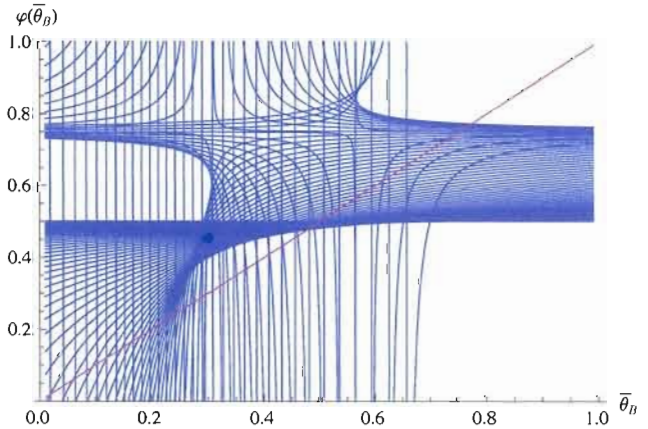


Figure 3.4 : Évolution de $\varphi\left(\overline{\theta}_B\right)$ en fonction de $\overline{\theta}_B$ avec sa bissectrice pour $\theta \in [0, 1]$

La confrontation de $\varphi\left(\overline{\theta}_B\right)$ à la bissectrice permet de tester s'il est réaliste de croire en l'existence de solutions. La présence de plusieurs courbes provient des différentes valeurs de $\overline{\theta}_B$ pour différents niveaux de paramètres. Ceux-ci $\left(\beta,\ \widehat{\theta}, k_1, k_2, s_1, s_2\right)$ varient entre $[0,1]$. Puisque chacune des courbes croise la bissectrice (en rouge) au moins une fois, il semble exister au moins une solution. Cette figure n'est cependant pas très claire. La figure 3.5 présente quelques-unes de ces courbes avec des paramètres choisis tels qu'ils respectent les hypothèses.

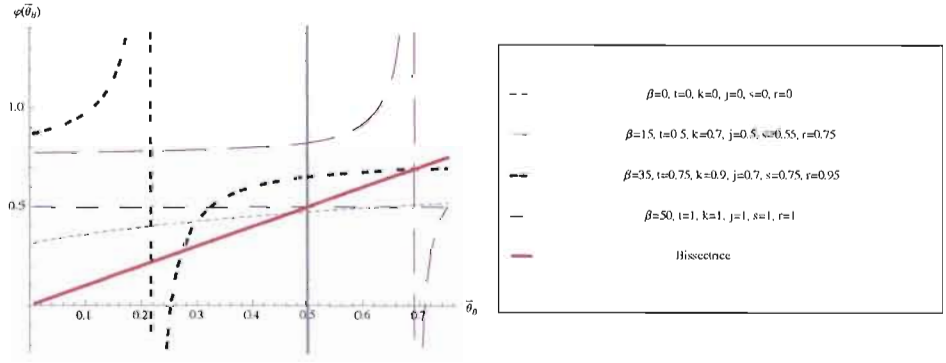


Figure 3.5 : Exemples de $\varphi\left(\overline{\theta}_B\right)$
selon diverse valeurs des paramètres

La figure 3.5 montre deux choses. Premièrement, il y a un croisement entre $\varphi\left(\overline{\theta}_B\right)$ et la bissectrice dans l'intervalle $\overline{\theta}_B \in [0, 0.5]$. Deuxièmement, les courbes sont de forme hyperbolique ; le choix des valeurs admissibles pour les diffèrents paramètres devient donc primordial. Cependant, étant donné le nombre de ces paramètres, et leurs nombreuses interactions, il ne sera pas possible de déterminer des valeur numériques pour chacun. À la place, une relation sera déterminée, de telle sorte que $\varphi(0) > 0$ et que $\varphi\left(\frac{1}{2}\right) < \frac{1}{2}$, soit ce qui est nécéssaire pour assurer l'existence d'au moins une solution.

$$\varphi(0) = \frac{\frac{1}{2} - \frac{3}{16}\left[\widehat{\theta}^5 s_1^2 k_1^2 + \left(\frac{\left(1-\widehat{\theta}^4\right)^2}{\left(1-\widehat{\theta}^3\right)}\right) s_2^2 k_2^2\right]}{1 - \frac{1}{4}\left[\widehat{\theta}^4 s_1^2 k_1^2 + \left(1 - \widehat{\theta}^4\right) s_2^2 k_2^2\right]} \tag{3.49}$$

$$\varphi\left(\frac{1}{2}\right) = \frac{\frac{1}{2} + \frac{3}{32}\beta\left[\widehat{\theta}^4 s_1^2 k_1^2 + \left(1 - \widehat{\theta}^4\right)\right] - \frac{3}{16}\left[\widehat{\theta}^5 s_1^2 k_1^2 + \left(\frac{\left(1-\widehat{\theta}^4\right)^2}{\left(1-\widehat{\theta}^3\right)}\right) s_2^2 k_2^2\right]}{1 + \frac{1}{8}\beta\left[\widehat{\theta}^3 s_1^2 k_1^2 + \left(1 - \widehat{\theta}^3\right)\right] 1 - \frac{1}{4}\left[\widehat{\theta}^4 s_1^2 k_1^2 + \left(1 - \widehat{\theta}^4\right) s_2^2 k_2^2\right]} \tag{3.50}$$

Il faut que $\varphi(0) \geq 0$ afin d'assurer que la courbe soit au-dessus de la bissectrice à la valeur minimale que peut prendre $\overline{\theta}_B$, et que $\varphi\left(\frac{1}{2}\right)$ *leq* $\frac{1}{2}$ pour être sous la bissectrice à la plus haute valeur que peut prendre $\overline{\theta}_B$. Pour ce faire, il faudra que le numérateur et le dénominateur de $\varphi(0)$ soient positifs ; ceci assurera le respect des restrictions sur les paramètres et sur $\overline{\theta}_B$.

$\varphi(0)$ – Numérateur :

$$\frac{1}{2} \geq \frac{3}{16}\left[\widehat{\theta}^5 s_1^2 k_1^2 + \left(\frac{\left(1 - \widehat{\theta}^4\right)^2}{\left(1 - \widehat{\theta}^3\right)}\right) s_2^2 k_2^2\right] \tag{3.51}$$

$$s_1^2 k_1^2 \leq \frac{8}{3\beta\,\widehat{\theta}^5} - \left(\frac{\left(1 - \widehat{\theta}^4\right)^2}{\left(1 - \widehat{\theta}^3\right)\widehat{\theta}^5}\right) s_2^2 k_2^2 \tag{3.52}$$

$$s_2^2 k_2^2 \leq \frac{8\left(1 - \widehat{\theta}^3\right)}{3\beta\left(1 - \widehat{\theta}^4\right)^2} - \left(\frac{\left(1 - \widehat{\theta}^3\right)\widehat{\theta}^5}{\left(1 - \widehat{\theta}^4\right)^2}\right) s_1^2 k_1^2 \tag{3.53}$$

$\varphi(0)$ – Dénominateur :

$$1 \geq \frac{1}{4}\left[\widehat{\theta}^4 s_1^2 k_1^2 + \left(1 - \widehat{\theta}^4\right) s_2^2 k_2^2\right] \tag{3.54}$$

$$s_1^2 k_1^2 \leq \frac{4\beta}{\widehat{\theta}^4} - \left(\frac{\left(1 - \widehat{\theta}^4\right)}{\widehat{\theta}^4}\right) s_2^2 k_2^2 \tag{3.55}$$

$$s_2^2 k_2^2 \leq \frac{4\beta}{\left(1 - \widehat{\theta}^4\right)} - \left(\frac{\widehat{\theta}^4}{\left(1 - \widehat{\theta}^4\right)}\right) s_1^2 k_1^2 \tag{3.56}$$

$\varphi\left(\frac{1}{2}\right)$

$$\frac{1}{2} \geq \frac{\frac{1}{2}+\frac{3}{32}\beta\left[\widehat{\theta}^4 s_1^2k_1^2+\left(1-\widehat{\theta}^4\right)\right]-\frac{3}{16}\beta\left[\widehat{\theta}^5 s_1^2k_1^2+\left(\frac{\left(1-\widehat{\theta}^4\right)^2}{\left(1-\widehat{\theta}^3\right)}\right)s_2^2k_2^2\right]}{1+\frac{1}{8}\beta\left[\widehat{\theta}^3 s_1^2k_1^2+\left(1-\widehat{\theta}^3\right)\right]1-\frac{1}{4}\beta\left[\widehat{\theta}^4 s_1^2k_1^2+\left(1-\widehat{\theta}^4\right)s_2^2k_2^2\right]} \tag{3.57}$$

$$s_1^2k_1^2 \leq \frac{\left[\frac{1}{16}\left(1-\widehat{\theta}^3\right)-\frac{7}{32}\left(1-\widehat{\theta}^4\right)-\frac{3}{16}\left(\frac{\left(1-\widehat{\theta}^4\right)^2}{\left(1-\widehat{\theta}^3\right)}\right)\right]s_2^2k_2^2}{\frac{7}{32}\widehat{\theta}^4-\frac{3}{16}\widehat{\theta}^5-\frac{1}{16}\widehat{\theta}^3} \tag{3.58}$$

$$s_2^2k_2^2 \leq \frac{\left[\frac{1}{16}\widehat{\theta}^3+\frac{3}{16}\widehat{\theta}^5-\frac{7}{32}\widehat{\theta}^4\right]s_1^2k_1^2}{\frac{7}{32}\left(1-\widehat{\theta}^4\right)+\frac{3}{16}\left(\frac{\left(1-\widehat{\theta}^4\right)^2}{\left(1-\widehat{\theta}^3\right)}\right)-\frac{1}{16}\left(1-\widehat{\theta}^3\right)} \tag{3.59}$$

Les différents paramètes, dans les équations 3.51 à 3.56, ne sont pas tous séparés parce qu'ils sont toujours présents par paires $s_i^2k_i^2$. Le tableau 3.1 présente une solution numérique possible pour chacune de ces équation, en considérant les différentes restrictions sur les paramètres.

Tableau 3.1 : Borne supérieure des paramètres pour assurer l'existence d'une solution à $\overline{\theta}_B$

	$s_1^2k_1^2$	$s_2^2k_2^2$	$\widehat{\theta}$	β
equation 3.52	$\frac{1}{2}$	$\frac{1}{64}$	$\frac{1}{2}$	$\frac{51}{2}$
equation 3.53*	n/d	n/d	n/d	n/d
equation 3.55	$\frac{1}{2}$	$\frac{127}{128}$	$\frac{543}{2048}$	$\frac{25551}{512}$
equation 3.55	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{8}$
equation 3.58	$\frac{1}{2}$	$\frac{1}{1024}$	$\frac{1}{6}$	$\frac{99}{2}$
equation 3.59	$\frac{15}{16}$	$\frac{1}{1024}$	$\frac{1}{3}$	$\frac{99}{2}$

* Mathematica n'est pas parvenu à résoudre cette équation; cependant les valeurs pour l'équation 3.52 conviennent.

Une fois ces valeurs déterminées, il est possible de montrer formellement l'existence d'au moins une solution pour $\overline{\theta}_B$.

Lemme 4 *[$\overline{\theta}_B$ possède une ou deux solutions]*

Soit

$$\psi\left(\overline{\theta}_B\right) = \varphi\left(\overline{\theta}_B\right) - \overline{\theta}_B$$

Posons la valeur $\overline{\theta}_B^*$ *telle que* $\psi\left(\overline{\theta}_B^*\right) = 0$ *Alors, comme*

$\psi\left(\overline{\theta}_B\right)$ *est continue ;*

$\exists\{s_1^2k_1^2, s_2^2k_2^2, \widehat{\theta}, \beta\}$ *t.q.* $\psi(0) > 0$;

$\psi(1) < 0$.

Par le théorème des valeurs intermédiaires

$$\exists\, \overline{\theta}_B \in (0,1]$$

qui satisfait $\overline{\theta}_B^*$

L'existence d'au moins une solution étant assurée, il faut déterminer les racines quadratiques de $\overline{\theta}_B$. Celles-ci ont été déterminées, encore une fois, à l'aide de Mathematica. La racine à retenir sera déterminée à la section suivante, tout comme les bornes des paramètres à utiliser ; étant donné l'allure du graphique de la figure 3.4, la solution devrait se trouver au voisinage de $\overline{\theta}_B = 0.5$, puisqu'il y a là la plus forte densité de croisements de $\varphi(\overline{\theta}_B)$ avec la bissectrice. La résolution de l'équation quadratique donne :

$$\frac{1}{\left(4\beta\left(1+\widehat{\theta}+\widehat{\theta}^2\right)\left(k_1^2s_1^2\,\widehat{\theta}^3 - k_2^2s_2^2\left(-1+\widehat{\theta}^3\right)\right)\right)} \times$$
$$\left(-6 - 6\,\widehat{\theta}(1+\widehat{\theta}) - 3\beta\left(1+\widehat{\theta}+\widehat{\theta}^2\right)\left(-k_1^2s_1^2\,\widehat{\theta}^4 + k_2^2s_2^2\left(-1+\widehat{\theta}^4\right)\right)\right.$$
$$\pm\sqrt{3}\sqrt{\left(\left(1+\widehat{\theta}+\widehat{\theta}^2\right)\left(3\beta^2k_2^2k_1^2s_2^2s_1^2(-1+\widehat{\theta})\,\widehat{\theta}^3 + 12\left(1+\widehat{\theta}+\widehat{\theta}^2\right)\right.\right.}$$
$$\left.\left.\left.+\ 4\beta\left(1+\widehat{\theta}+\widehat{\theta}^2\right)\left(k_1^2s_1^2(2-3\,\widehat{\theta})\,\widehat{\theta}^3 + k_2^2s_2^2\left(-1+\widehat{\theta}^3(-2+3\,\widehat{\theta})\right)\right)\right)\right)\right)$$

La bonne racine sera déterminée graphiquement, lors de la détermination de l'équilibre ; une seule devrait donner un équilibre bayesien satisfaisant aux critères déterminés au début du présent chapitre.

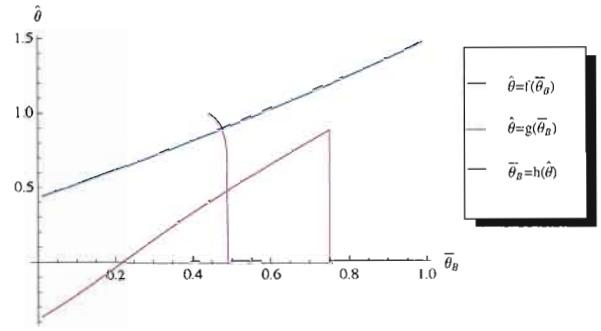
3.4.2 Détermination de $\widehat{\theta}$ à l'aide du critère d'indifférence

La dernière variable d'intérêt qu'il reste à déterminer est $\widehat{\theta}$, la limite entre un talent fort et un talent faible. Autrement dit, c'est le talent minimal requis pour qu'un programmeur soit à même d'entreprendre un projet difficile. Cette variable sera définie à l'aide du critère d'indifférence, soit la valeur de θ pour laquelle l'utilité anticipée est la même, quel que soit le projet entrepris $\left(\theta = \widehat{\theta}\right)$. Il faut donc déterminer les utilités anticipées des deux projets en fonction de $\overline{\theta}_B$ et de $\widehat{\theta}$. Ceci est réalisé à l'aide des fonctions de meilleure réponse. Puisque $\overline{\theta}_{G_1}$ et $\overline{\theta}_{G_2}$ sont connus, il sera possible de les substituer dans les fonctions meilleure réponse, et celles-ci dans les fonctions d'utilité anticipée.

Une fois ces fonctions d'utilité trouvées, elles sont égalisées, puis résolues pour $\widehat{\theta}$. Par contre, étant donné la complexité de l'équation résultante (l'équation est du huitième degré pour plusieurs des paramètres), la résolution symbolique n'est pas présentée. Une fois les huit solutions pour $\widehat{\theta}$ déterminées, plusieurs d'entre elles peuvent être éliminées sans plus de considération. Comme, dans le présent modèle, le domaine et l'image sont dans $\mathbb{R}$, les quatre racines complexes peuvent être éliminées. Deux racines sont les racines triviales. Sans être nécessairement éliminées, elles peuvent être ignorées pour le moment. $\left(\overline{\theta}_B = 0,\ \widehat{\theta} = 0\right)$

Il reste deux racines à évaluer[1]. Afin d'évaluer le critère d'indifférence, la figure 3.6, comprenant sur un axe $\overline{\theta}_B = f\left(\widehat{\theta}\right)$ et sur l'autre $\widehat{\theta} = g\left(\overline{\theta}_B\right)$, présente la confrontation de ces fonctions. Il est à noter ici que malgré son apparence, la courbe définissant $\overline{\theta}_B$ est bel et bien une fonction, celle-ci étant par rapport à $\widehat{\theta}$, soit en quelque sorte $x = f(y)$. À

1. *c.f.* annexe mathématique, section B.3.



(a) $k_1 = 0.85, k_2 = 0.15, s_1 = 0.3, s_2 = 0.7, \beta = 30$

Figure 3.6 : Évaluation des équilibres potentiels selon les hypothèses sur $\overline{\theta}_{G_1}, \overline{\theta}_{G_2}, \overline{\theta}_B$ et $\widehat{\theta}$

la section 3.1, certaines restrictions ont été posées sur les gains. Ces dernières stipulent que $\overline{\theta}_B < \overline{\theta}_{G_1} < \overline{\theta}_{G_2}$ doit être respectée. Les valeurs que l'on retrouve dans la figure 3.6 sont données dans le tableau 3.2.

Tableau 3.2 : Détermination numérique de la bonne racine pour $\widehat{\theta}$

	$\widehat{\theta}_1\left(\overline{\theta}_B\right)$	$\widehat{\theta}_2 x\left(\overline{\theta}_B\right)$
	0.893694	0.482537
$\overline{\theta}_B$	0.475276	0.489426
$\overline{\theta}_{G_1}$	0.67027	0.361903
$\overline{\theta}_{G_2}$	0.948834	0.799124

Ceci démontre que la bonne valeur de $\widehat{\theta}$ est celle de la courbe en bleu dans le graphique de la figure 3.6, soit $\widehat{\theta} = \widehat{\theta}_1$ tel qu'il est défini dans l'annexe mathématique, à la section C.

3.5 Statiques comparatives

Une fois les fonctions de gain et la limite de talent fort/faible déterminées, il faut analyser l'impact des différents paramètres sur ces fonctions. En d'autres mots, comment est-ce que la probabilité de réussite d'un projet affecte les gains attendus en cas d'échec ? Est-ce que les chances de réussir un projet ont un impact sur le talent minimal requis pour

Tableau 3.3 : Valeurs retenues pour les différents paramètres dans les statiques comparatives

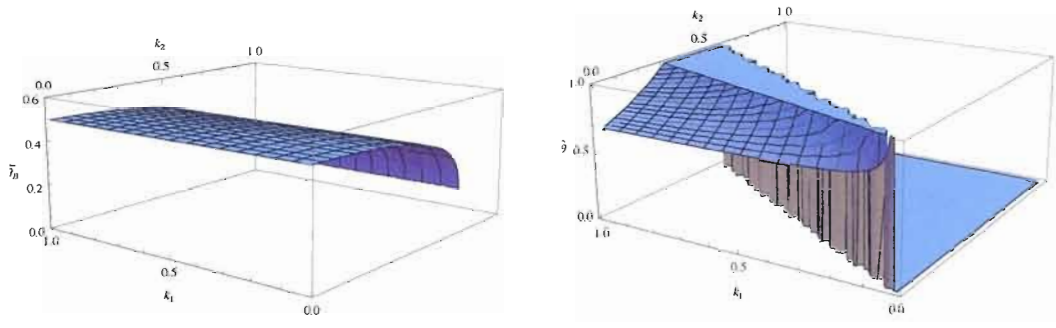
paramètre	valeur
k_1	0.65
k_2	0.15
s_1	0.3
s_2	0.5
β	20
θ	0.5

entreprendre un projet difficile ? Et l'effort optimal est-il, lui, affecté par ces probabilités ? Et dans tous les cas, est-ce que ces probabilités ont le même effet ? Comme la plupart de ces paramètres viennent en paire ($k_1 s_2, k_2 s_2$), l'analyse sera effectuée graphiquement. Chaque sous-section présentera un graphique démontrant l'effet de ces paramètres sur les variables d'intérêt, selon certaines valeurs de chacun des autres paramètres telles que les restrictions déterminées plus tôt sont respectées.

Ces graphiques sont présentés en fonction de deux paramètres à la fois ; le but de ce type de graphique n'est pas de présenter la variation simultanée de ces paramètres, mais plutôt d'illustrer les bornes des différents paramètres. Chaque graphique peut être analysé en fixant l'un des deux paramètres, et en analysant la « tranche » correspondante lorsque l'autre paramètre varie. Ainsi, pour une valeur donnée d'un paramètre, il sera possible d'observer à la fois les valeurs maximale et minimale de l'autre paramètre, ainsi que son impact individuel. Les valeurs retenues pour les paramètres (qui ne sont pas sur un axe) dans les graphiques sont identifiées dans le tableau 3.3.

3.5.1 La difficulté de réussite d'un projet – k_1, k_2

La première chose que l'on peut remarquer dans la figure 3.7a est le faible impact de la probabilité de réussir un projet facile (k_1). En fait, cet impact est nul ; augmenter la probabilité de réussir un projet facile, pour toute valeur de k_2, n'a aucun impact sur le gain en cas d'échec ou de non-réussite ($\overline{\theta}_B$). Ceci correspond bien à l'idée derrière le projet facile : licence BSD ne maximise pas la visibilité des programmeurs, mais comme



(a) Impact sur $\overline{\theta}_B$ (b) Impact sur $\widehat{\theta}$

Figure 3.7 : Impact de la probabilité de réussite d'un projet sur $\overline{\theta}_B$ et $\widehat{\theta}$

il est supposé que les programmeurs choisissant ce projet ont un talent faible, il est à supposer que la visibilité n'est pas leur motivation principale. On peut percevoir ces paramètres (k_1, k_2) comme le niveau de difficulté propre à un projet : plus la probabilité de réussite est élevée, plus le projet est facile. La probabilité de réussite d'un projet difficile a un effet important, qui peut être divisé en trois phases. Lorsque $k_2 \in [0, 0.5]$, il n'y a qu'une faible diminution de $\overline{\theta}_B$, entre $[0.5, 0.75]$, il y a une forte décroissance du gain en cas d'échec, et lorsque k_2 dépasse 0.75, le gain en cas d'échec devient « imaginaire » ; en fait, l'image de l'équation est dans les nombres complexes.

Ceci implique que si la probabilité de réussir un projet classé difficile devient trop grande, il n'y a plus aucun gain si le projet n'est pas réussi ou si le programmeur ne se démarque pas ; la réputation du programmeur devient nulle. Les programmeurs participant à un projet Open Source recherchent l'amélioration de leur réputation. Ceux qui choisissent un projet facile ne recherchent pas autant que les autres à envoyer un signal à l'industrie commerciale ; ainsi leurs chances de réussite n'affectent pas le gain qu'un programmeur reçoit lorsque sont projet échoue ou n'est pas remarqué. Tandis que les chances de réussite d'un projet difficile ont un impact sur la *qualité* du signal envoyé à l'industrie commerciale : si le projet difficile a une grande probabilité de

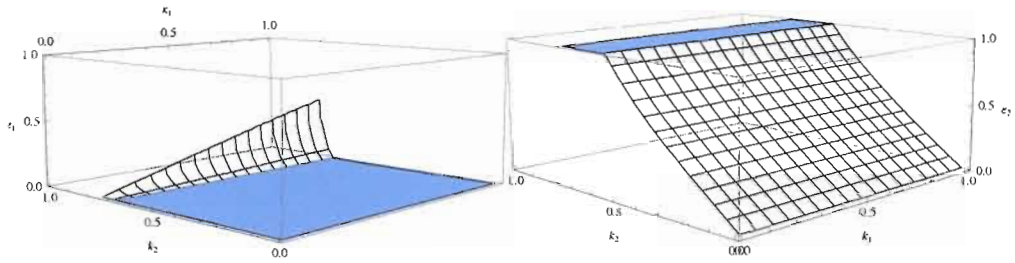
succès, son échec doit avoir un impact *négatif* fort sur la réputation du programmeur participant à ce projet.

Le graphique 3.7b montre que les valeurs admissibles de $\{k_1, k_2\}$ ont une borne supérieure, et que chacune est relative à la valeur de l'autre paramètre. Par exemple, si $k_1 = 0.5$, il ne faut pas que k_2 soit supérieur à environ 0.25 pour que $\widehat{\theta} \in (0, 1)$. Si k_2 est légèrement supérieur à cette valeur, alors tous les projet sont faciles puisque $\widehat{\theta} > 1$, et si k_2 est largement supérieur à cette valeur, alors tous les projets sont difficiles : $\widehat{\theta} < 0$. Autrement dit, si la probabilité de réussir un projet difficile est très élevée, ce projet est, de fait, un projet facile. L'impact de k_2 est sensiblement le même, mais en plus accentué : lorsque ce paramètre passe de 0 à 0.3, $\widehat{\theta}$ augmente jusqu'à être égal (voire à dépasser) à 1 et ce, lorsque k_1 est nul.

On peut donc comprendre ici que la probabilité de réussite d'un projet a deux impacts sur les projets difficiles : une augmentation de k_2 fera diminuer le talent nécéssaire pour l'entreprendre (ce projet difficile est donc plus facile) et, en cas d'échec, la réputation des programmeurs participant à ce projet sera diminuée d'autant plus fortement que la probabilité de réussite était élevée. Ceci concorde avec l'idée que des programmeurs entreprenant une projet assujetti à la GPL (donc un projet difficile) souhaitent envoyer un signal à l'industrie commerciale quant à leur talent. Les programmeurs participant à un projet utilisant une licence BSD ne recherchent que l'amélioration de leur réputation au sein de la communauté Open Source, et les probabilités de réussite n'ont qu'un faible impact sur leur espérance de gain en cas d'échec.

Impacts sur la fonction d'effort

L'ensemble de stratégie d'équilibre, dans ce modèle, est la fonction d'effort optimal ; il est donc nécéssaire d'évaluer l'impact des différents paramètres sur la stratégie d'équilibre des programmeurs.



(a) Impact sur le projet facile (b) Impact sur le projet difficile

Figure 3.8 : Impact de la probabilité de réussite d'un projet sur les fonctions d'effort optimal

Les deux graphiques de la figure 3.8 montrent que les probabilités de réussite influent positivement sur l'effort qui est attendu (*ex ante*) de la part d'un programmeur. Autrement dit, plus les probabilités de réussite d'un projet (peu importe sa licence) sont grandes, plus l'effort optimal que devrait choisir un programmeur lorsqu'il s'embarque dans un projet est grand ; le coût de la participation est plus élevé, mais le gain espéré aussi.

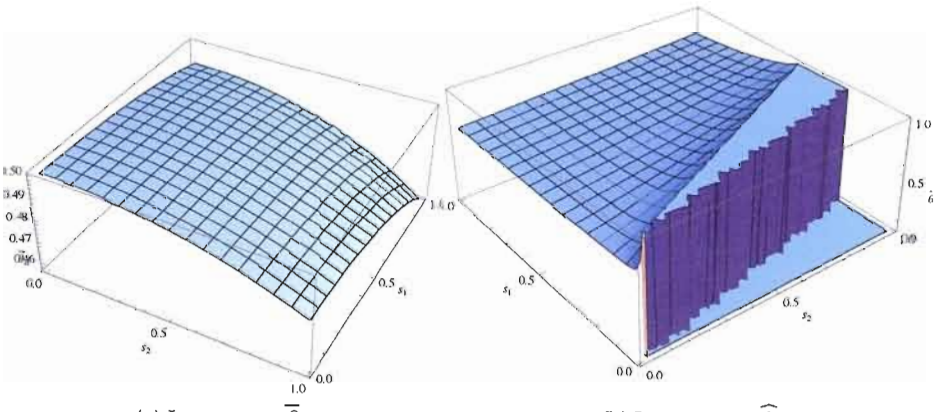
Par contre, pour qu'un programmeur fournisse un effort optimal positif ($e_1(\theta) > 0$), il faut que la probabilité de succès d'un projet difficile soit assez élevée, mais pas trop. Autrement dit, si k_2 n'est pas situé entre certaines bornes, l'effort optimal dans un projet facile sera nul [2]. Ainsi, aucun programmeur n'est intéressé à participer à un projet facile s'il n'est pas relativement aisé de réussir un projet difficile. Pour le cas de l'effort optimal dans un projet difficile, on remarque que la probabilité de réussite d'un projet facile n'a aucun impact sur l'effort fourni par les programmeurs de grand talent. Autrement dit, les programmeurs désirant envoyer un signal à l'industrie du logiciel commercial ne se soucient pas de la probabilité de réussir un projet facile. Quant à la probabilité de réussir leur propre projet, plus elle est élevée, plus l'effort optimal fourni sera grand. Si, cependant, cette probabilité dépasse un certain seuil, l'effort fourni disparaît puisque le projet est trop facile. Cette situation s'explique par la motivation réputationelle de

2. Il sera considéré que toute valeur négative est égale à 0.

chaque programmeur : si la probabilité de réussite d'un projet difficile est trop élevée, il devient plus intéressant pour tous les programmeurs de se joindre à un projet difficile. Au contraire, si elle est trop basse, tous les programmeurs se joindront à un projet facile, « diluant » l'effort optimal requis.

Cette situation provient du gain en cas de succès des différents types de projet. Comme il a été montré dans la figure 3.3, les paiements sont partitionnés à $\frac{3}{4}$: $\overline{\theta}_{G_1} < 0.75$ et $\overline{\theta}_{G_2} > 0.75$.

3.5.2 La facilité de se distinguer des autres – s_1, s_2



(a) Impact sur $\overline{\theta}_B$ (b) Impact sur $\widehat{\theta}$

Figure 3.9 : Impact de la probabilité de réussite d'un projet sur $\overline{\theta}_B$ et $\widehat{\theta}$

Les impacts des probabilités de se distinguer dans un projet facile comme dans un projet difficile sont sensiblement les mêmes. Le graphique 3.9a montre une légère diminution du gain en cas de non-réussite si la probabilité de se faire remarquer est plus élevée dans un projet facile comme dans un projet difficile. Ces probabilités peuvent s'interpréter comme l'intérêt que la communauté Open Source porte à un projet. Si cet intérêt est très élevé, il sera évidemment plus facile pour les programmeurs de ce projet de se distinguer des autres. Si un programmeur n'y parvient pas, sa réputation souffrira d'autant plus que le gain en cas d'échec ou de non-réussite est élevé. Ainsi, dans un

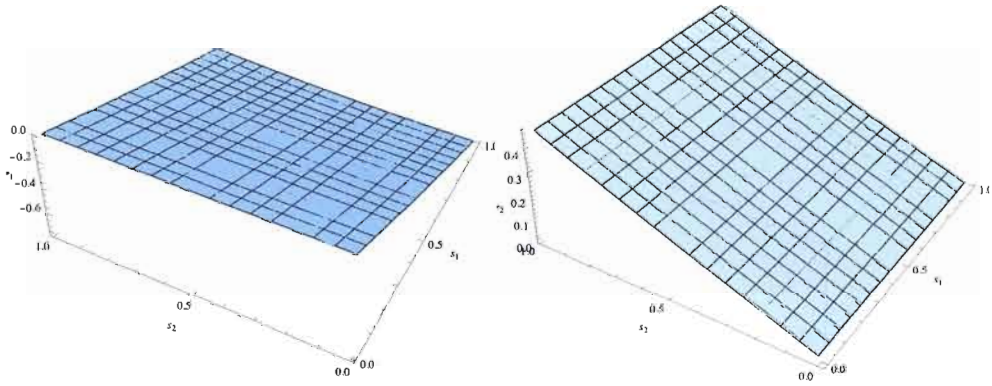
projet facile, plus les chances d'être remarqué sont élevées, plus le gain réputationnel en cas d'échec sera faible.

Le graphique 3.9b est, lui, un peu plus difficile à interpréter. Comme il a été expliqué ci-dessus, le but de ce graphique n'est pas d'analyser l'impact d'un mouvement simultané des paramètres, mais plutôt d'illustrer les bornes de chacun. Ici, on peut voir que pour toute valeur que peut prendre s_2, s_1 doit être supérieur à une borne inférieure pour permettre l'existence des deux classes de projet. Par exemple, si $s_2 = 0.5$, s_1 doit être supérieur à environ 0.23 pour que $\widehat{\theta}$ soit inférieur à 1 mais supérieur à 0. En d'autres termes, lorsqu'un projet difficile a de grandes chances d'être remarqué (suivant une réussite), il ne faut pas qu'un projet facile, lui, soit remarqué avec une faible probabilité ni une probabilité très faible. Les programmeurs se répartiront entre les deux classes de projets seulement si les chances d'être remarqué sont suffisament élevées dans chacun des projets. Sinon, il y a deux situations possibles. Par exemple, lorsque s_2 est très élevé mais que s_1 est très faible, tous les projets sont difficiles $\left(\widehat{\theta} < 0\right)$ et personne ne participe à un projet facile. Toujours lorsque s_2 est très élevé, si s_1 est moyennement élevé, il n'y aura personne pour participer à un projet difficile $\left(\widehat{\theta} > 1\right)$. Tous les programmeurs se joindront à un projet facile, puisque, comme il a été mentionné précédemment, une possibilité trop élevée d'être remarquée nuit à la qualité du signal envoyé à l'industrie.

Impacts sur la fonction d'effort

La première chose à remarquer dans la figure 3.10 est que seulement la probabilité de distinction propre à une classe de projets a une influence sur l'effort optimal. Cependant, ce paramètre a un effet contraire selon la classe en question. Dans le cas du projet facile, plus la probabilité de se distinguer est élevée, plus l'effort optimal diminue, tandis que dans le projet difficile, l'effort optimal croît avec la probabilité de se distinguer.

Ceci correspond aux motivations des programmeurs : ceux qui participent à un projet facile ne recherchent que l'amélioration de leur propre réputation, tandis que les programmeurs participant à un projet assujetti à la licence GPL cherchent à envoyer



(a) $k_1 = 0.65, k_2 = 0.15, \beta = 20$ (b) $k_1 = 0.65, k_2 = 0.15, \beta = 20$

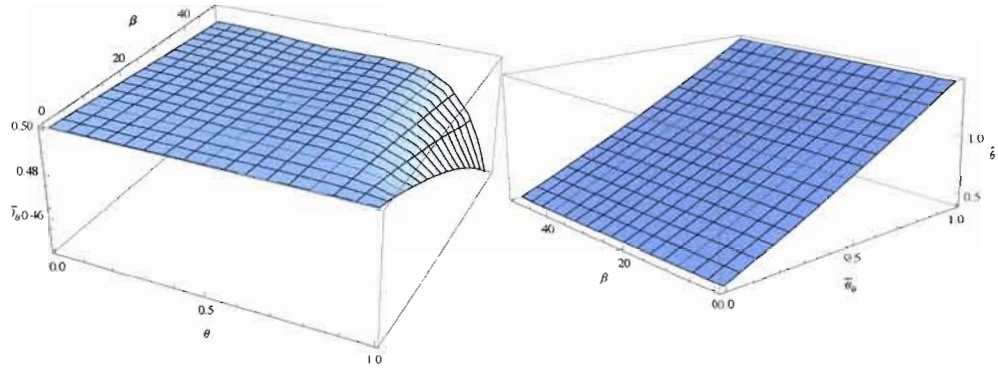
Figure 3.10 : Impact de la probabilité de distinction d'un projet sur $\overline{\theta}_B$ et $\widehat{\theta}$

un signal à l'industrie commerciale. Ainsi, dans un projet difficile, s'il est plus facile de se distinguer, il est aussi plus facile d'envoyer un signal à l'industrie. La facilité de se distinguer est donc, pour les programmeurs de projets difficiles, un motivateur.

3.5.3 Les autres paramètres : l'intérêt et le talent des programmeurs – β, $\widehat{\theta}$

À la différence des sections précédentes, les deux graphiques de la figure 3.12 ne présentent pas les mêmes paramètres. La raison en est bien simple : l'équation de gain en cas d'échec ne contient pas $\overline{\theta}_B$ comme variable, et l'équation pour la limite entre le talent faible et fort ne contient pas $\widehat{\theta}$. Donc, chacun des graphiques présente les impacts de β, l'intérêt des programmeurs face à la programmation Open Source et « l'autre » paramètre, soit l'ensemble des valeurs admissibles de l'autre équation dans la figure 3.6.

Le graphique 3.11a montre que chacun de ces deux paramètres n'a qu'un très faible impact sur le gain en cas de non-succès. En fait, ce graphique montre que le talent initial d'un programmeur (θ) n'a aucun impact sur le gain en cas d'échec. Dans le cas



(a) $k_1 = 0.65, k_2 = 0.15, s_1 = 0.3, s_2 = 0.5$ (b) $k_1 = 0.65, k_2 = 0.15, s_1 = 0.3, s_2 = 0.5$

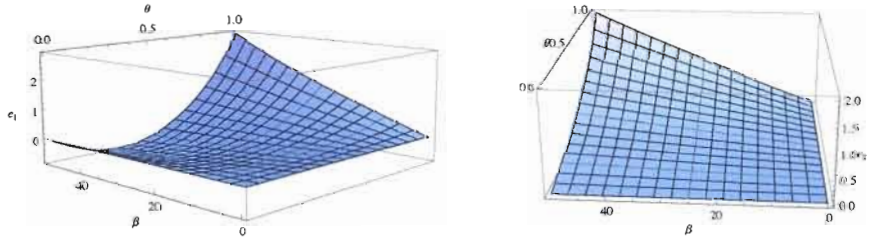
Figure 3.11 : Impact de l'autre variable d'intérêt et de l'intérêt des programmeurs sur $\overline{\theta}_B$ et $\widehat{\theta}$

de l'intérêt que les programmeurs portent à leur participation à la communauté Open Source, il y a, quand β varie de 0 à 50, une décroissance continue de $\overline{\theta}_B$. Autrement dit, de θ, un accroissement de β entraîne une diminution du gain en cas d'échec. Peu importe son talent, un programmeur qui démontre un très grand intérêt envers sa réputation verra cette dernière diminuée en conséquence après un échec. Cette diminution du gain en l'absence de réussite peut aussi être analysée en fonction du gain en cas de succès : les fonctions d'utilité des programmeurs font appel à la « prime » en cas de succès, soit la différence entre le gain en cas de réussite et le gain en cas d'échec $\left(\overline{\theta}_{G_i} - \overline{\theta}_B\right)$. Évidemment, plus le gain en cas d'échec est petit, plus le gain en cas de réussite est élevé. De plus, le gain en cas de succès est basé uniquement sur le talent initial d'un programmeur.

Bien que cela puisse sembler superflu, le graphique 3.11b présente l'impact de l'intérêt des programmeurs envers leur réputation. Cet impact est nul ; en fait, ce paramètre n'entre même pas dans l'équation de $\widehat{\theta}$. Il est présenté ici quand même pour souligner le fait que même si un programmeur manifeste un très grand intérêt envers l'amélioration de sa réputation, le niveau de difficulté d'un projet ne s'en trouve pas affecté.

Cependant, le talent requis pour entreprendre un projet difficile va croissant selon le gain en cas d'échec. Plus celui-ci est élevé, plus un projet difficile demande un talent minimal élevé afin d'être entrepris. Ceci s'explique probablement par le fait que les deux fonctions $\left(\overline{\theta}_B,\ \widehat{\theta}\right)$ sont déterminées simulanément : ce n'est pas l'augmentation de $\overline{\theta}_B$ comme tel qui fait augmenter $\widehat{\theta}$, mais plutôt l'ensemble des paramètres sous-jacents. Il a été expliqué précédemment comment ceux-ci affectent le gain en cas d'échec et la frontière entre un talent faible et un talent fort. Ce qui en ressort est toujours valide : plus les conditions sont difficiles (faible intérêt, faibles chances de réussite et de se distinguer, etc.), plus il est difficile d'atteindre le succès.

Impact sur la fonction d'effort



(a) $k_1 = 0.65, k_2 = 0.15, s_1 = 0.3, s_2 = 0.5$ (b) $k_1 = 0.65, k_2 = 0.15, s_1 = 0.3, s_2 = 0.5$

Figure 3.12 : Impact du talent et de l'intérêt des programmeurs sur $\overline{\theta}_B$ et $\widehat{\theta}$

Le graphique 3.12a montre que β doit être très élevé afin d'avoir un impact significatif sur la fonction de meilleure réponse des programmeurs. En fait, son effet individuel est pratiquement nul. Le talent des programmeurs fait cependant croître exponentiellement l'effort fourni par les programmeurs. Autrement dit, plus les programmeurs d'un projet facile sont intérressés, plus ils fourniront d'efforts. De même, plus ils sont talentueux, plus l'effort optimal sera élevé. Par contre, les programmeurs de faible talent fourniront plus d'efforts que les programmeurs de talent moyen, et les programmeurs avec un talent fort fourniront encore plus d'efforts. Or, il faut rappeler qu'idéalement, un programmeur au talent initial élevé ($\theta \geq \widehat{\theta}$) ne devrait pas participer à un projet facile,

mais plutôt à un projet difficile. Cependant, le modèle n'interdit pas à un programmeur de grand talent de participer à un proje facile « pour le plaisir ». Quant à l'effort optimal dans un projet diffici le, les différents paramètres ont sensiblement le même impact. L'intérêt des programmeurs en soi, tout comme le talent, ont un impact positif : pour tout niveau de talent, un intérêt élevé envers sa réputation incitera un programmeur à fournir un effort optimal supérieur.

CHAPITRE IV

RÉSULTATS : COMPARAISON DE TROIS SITUATIONS

Ce chapitre présente trois situations qui seront analysées à l'aide des modèles présentés précédemment. Par la suite, l'ensemble des situations sera analysé pour déterminer lequel des paradigmes est préférable : s'il n'y a qu'un copyleft fort ou s'il est préférable qu'un copyleft faible soit aussi présent.

Les situations qui sont présentées sont simplifiées, mais réalistes : il y a une situation bonne, une moyenne et une mauvaise. Une situation bonne implique des valeurs élevées pour les différents paramètres : l'intérêt des programmeurs et celui de la communauté est élevé, comme les probabilités de réussite. Dans la situation moyenne, ce sont des valeurs moyennes, et ainsi de suite. Dans chaque cas, trois programmeurs (trois valeurs différentes pour θ) participent à chacune des possibilités : le modèle à une licence d'une part, le projet facile et le projet difficile du modèle à deux licences d'autre part. Ainsi, il est possible d'évaluer les différentes valeurs d'équilibre, l'utilité que procure chacun des projets à chacun des programmeurs et, dans le cas du modèle à deux licences, le talent minimal requis pour entreprendre un projet difficile $(\widehat{\theta})$. Ces différentes valeurs permettront de déterminer, premièrement, si un paradigme dans lequel il y a deux licences est préférable à l'existence d'une seule licence, et deuxièmement, si l'auto-sélection des programmeurs se produit réellement.

Tableau 4.1 : Paramètres définissant les différentes situations

Situation	Paramètres						
	Modèle à une licence		Modèle à deux licences				
	β	s	β	k_1	k_2	s_1	s_2
Bonne	2.5	0.5	25	0.85	0.24	0.6	0.8
Moyenne	1.1	0.35	20	0.65	0.15	0.3	0.5
Mauvaise	0.5	0.15	10	0.55	0.05	0.15	0.35

4.1 Description des paramètres utilisés

Les situations décrites précédemment seront définies à l'aide de l'ensemble des différents paramètres pour chacun des modèles. Bien sûr, ces paramètres ne sont pas exactement les mêmes dans les deux modèles. Les situations sont néanmoins comparables : le talent des programmeurs, la probabilité de se distinguer et l'intérêt des programmeurs sont présents dans les deux modèles.

Aussi, une dotation initiale de $\mathscr{Y} = 100$ est allouée à chacun des programmeurs afin d'établir une base de comparaison pour les différentes situations. Il n'y a que la probabilité de réussite qui soit vraiment différente entre les deux modèles, puisque les projets sont considérés comme sur un pied d'égalité lorsqu'il n'y a que la GPL. La probabilité de distinction peut sembler différente entre les deux modèles, mais ceci ne sert qu'à refléter les deux classes de projets dans le modèle avec les licences GPL et BSD. Le tableau 4.1 présente les paramètres pour les deux modèles, pour chacune des situations.

Il est important de noter ici que les différents niveaux des paramètres ont été déterminés comme étant ceux permettant d'obtenir des résultats compris dans les réels ($\mathbb{R}$) tout en respectant les relations déterminées dans les chapitres précédents. Autrement dit, les niveaux maximums et minimums des différents paramètres sont choisis de sorte qu'aucun résultat (effort, paiement) ne soit imaginaire, comme il a été décrit dans les chapitres précédents.

4.2 Présentation et analyse des résultats

Tableau 4.2 : Valeurs d'équilibre – Situation bonne

Modèle	Variables	$\theta = 0.85$	$\theta = 0.5$	$\theta = 0.25$
Modèle à 1 licence				
GPL seulement	$e^*(\theta)$	0.281121	0.165365	0.0826826
	$\overline{\theta}_G$	0.75		
	$\overline{\theta}_B$	0.485416		
	$\overline{\theta}_G - \overline{\theta}_B$	0.264584		
	U	101.363	101.265	101.226
Modèle à 2 licences				
Projet BSD	$e_1^*(\theta)$	n/d	0.820716	0.67016
	$\overline{\theta}_{G_1}$	0.596036		
Projet GPL	$e_2^*(\theta)$	2.25573	n/d	n/d
	$\overline{\theta}_{G_2}$	0.90515		
Variables communes	$\overline{\theta}_B$	0.440783		
	$\overline{\theta}_{G_1} - \overline{\theta}_B$	0.24376	0.12874	0.123607
	$\overline{\theta}_{G_2} - \overline{\theta}_B$	0.552874	0.437874	0.432721
	$\widehat{\theta}$	0.794714		
	U_1	0	112.019	111.888
	U_2	109.301	0	0

Les valeurs numériques des équilibres sont présentées dans les différents tableaux de ce chapitre. Chacun présente les valeurs d'équilibre, l'utilité procurée et, dans le cas du modèle à deux licences, la limite entre un talent fort et un talent faible. Les équilibres sont cohérents : dans tous les cas, on peut constater que $\overline{\theta}_B < \overline{\theta}_G$ dans le premier modèle et que $\overline{\theta}_B < \overline{\theta}_{G_1} < \overline{\theta}_{G_2}$ dans le second modèle. De plus, les valeurs de gain en cas d'échec et, dans le modèle à deux licences, de limite entre un talent fort et faible sont positives et réelles. Ceci indique que les restrictions déterminées pour chacun des paramètres sont respectées. Les résultats seront analysés dans un premier temps en

comparant les deux modèles, puis en comparant les différentes situations pour chacun des modèles.

Tableau 4.3 : Valeurs d'équilibre – Situation moyenne

Modèle	Variables	$\theta = 0.85$	$\theta = 0.5$	$\theta = 0.25$
Modèle à 1 licence				
GPL seulement	$e^*(\theta)$	0.0827524	0.0486779	0.0243389
	$\overline{\theta}_G$	0.75		
	$\overline{\theta}_B$	0.497128		
	$\overline{\theta}_G - \overline{\theta}_B$	0.252872		
	U	100.56	100.552	100.548
Modèle à 2 licences				
Projet BSD	$e_1^*(\theta)$	0.573672	0.332816	0.165772
	$\overline{\theta}_{G_1}$	0.667112		
Projet GPL	$e_2^*(\theta)$	n/d	n/d	n/d
	$\overline{\theta}_{G_2}$	0.946894		
Variables communes	$\overline{\theta}_B$	0.37353		
	$\overline{\theta}_{G_1} - \overline{\theta}_B$	0.173053	0.170675	0.170023
	$\overline{\theta}_{G_2} - \overline{\theta}_B$	0.452835	0.450456	0.449804
	$\widehat{\theta}$	0.889483		
	U_1	110.046	109.984	109.956
	U_2	0	0	0

Les tableaux 4.2, 4.3 et 4.4 indiquent clairement qu'un modèle à deux licences procure beaucoup plus d'utilité que le modèle à une licence, peu importe le type de programmeur. Ainsi, il est permis de croire qu'un monde où plusieurs licences existent, dans lequel un programmeur est récompensé en fonction de son talent plutôt qu'en fonction de sa production uniquement est préférable pour la société, malgré le fait qu'un gain en cas de succès soit beaucoup plus faible dans le projet facile du modèle à deux licences que dans le modèle à une seule licence.

En outre, on peut constater que les gains en cas de succès, dans le modèle à deux licences, oscillent autour du gain en cas de succès du modèle à une licence. Si $\widehat{\theta}$ est utilisé comme proportion de programmeur choisissant le projet facile, et $\left(1 - \widehat{\theta}\right)$ comme proportion de programmeurs dans le projet difficile, un gain en cas de succès moyen, dans le modèle à deux licences, est défini comme $\overline{G} = \widehat{\theta} \cdot \overline{\theta}_{G_1} + \left(1 - \widehat{\theta}\right) \cdot \overline{\theta}_{G_2}$.

Tableau 4.4 : Valeurs d'équilibre – Situation mauvaise

Modèle	Variables	$\theta = 0.85$	$\theta = 0.5$	$\theta = 0.25$
Modèle à 1 licence				
GPL seulement	$e^*(\theta)$	0.015925	0.00938381	0.0046919
	$\overline{\theta}_G$	0.75		
	$\overline{\theta}_B$	0.499765		
	$\overline{\theta}_G - \overline{\theta}_B$	0.250235		
	U	100.25	100.25	100.25
Modèle à 2 licences				
Projet BSD	$e_1^*(\theta)$	n/d	0.0348867	0.0174383
	$\overline{\theta}_{G_1}$	0.588447		
Projet GPL	$e_2^*(\theta)$	0.0593588	n/d	n/d
	$\overline{\theta}_{G_2}$	0.898713		
Variables communes	$\overline{\theta}_B$	0.448051		
	$\overline{\theta}_{G_1} - \overline{\theta}_B$	0.0847846	0.0845739	0.0845493
	$\overline{\theta}_{G_2} - \overline{\theta}_B$	0.399051	0.39884	0.398815
	$\widehat{\theta}$	0.779263		
	U_1	0	104.999	104.999
	U_2	104.998	0	0

Les résultats obtenus pour chacune des situations et des types de programmeurs sont présentés dans tableau 4.5. On peut y constater qu'en moyenne, le gain en cas de succès est légèrement inférieur au gain en cas de succès dans le modèle à une licence. Il est cependant surprenant de constater que le plus haut gain moyen est dans la situation

moyenne ; il aurait été attendu que ce soit dans la situation mauvaise, où un succès est plus difficile à obtenir. Encore plus étonnant, lorsque les conditions se dégradent, la prime accordée à la suite d'un succès $\left(\overline{\theta}_G - \overline{\theta}_B\right)$ diminue.

Tableau 4.5 : Gains moyens en cas de succès $\left(\overline{\theta}_{G_{moy}}\right)$ pour le modèle à deux licences

	Situation		
Gain Moyen	Bonne	Moyenne	Mauvaise
$\overline{\theta}_{\overline{G}}$	0.659493	0.698033	0.653817

Intuitivement, il aurait été attendu que la prime en cas de succès soit plus élevée lorsqu'il est plus difficile d'obtenir un succès. Or, il semble que ce soit plutôt le gain en cas d'échec qui augmente quand une réussite ou une distinction est plus difficile à obtenir. Il faut donc conclure que lorsqu'il n'y a que la GPL et que la situation est difficile, la communauté encourage plutôt la participation que le succès, et attribue les « récompenses » en conséquence. Ceci permet de croire que des conditions qui se dégradent impliquent un intérêt plus faible de la communauté, pas seulement des programmeurs. De plus, un intérêt moyen présente le $\widehat{\theta}$ le plus élevé ; les conditions moyennes seraient donc celles qui rallient la « majorité », donc un talent plus élevé est requis pour entreprendre un projet difficile. Quant à l'effort optimal, il est clairement supérieur dans un modèle à deux licences : dans tous les cas, l'effort optimal fourni lorsqu'il y a deux licences est près de huit fois l'effort optimal du modèle à une licence. Ainsi, laisser les programmeurs choisir le projet qui corresponde le mieux à leur talent et à leur motivation personelle permet à chacun de s'investir beaucoup plus.

Afin d'évaluer l'importance des valeurs retenues pour les différents paramètres, une analyse des différentes situations pour chacun des modèles s'impose. Si l'on commence par le modèle « GPL seulement », une chose en particulier étonne : la chute rapide de l'effort fourni. Alors que dans la situation bonne, un grand effort est fourni, dans la mauvaise, l'effort devient nul. En comparaison, l'utilité que procure la communauté Open Source ne varie presque pas. Ceci implique que pour qu'un effort soit fourni, il faut que les conditions soient intéressantes.

Si l'on compare les différentes situations dans le modèle où l'on retrouve les licences GPL et BSD, on peut remarquer que tenir compte des motivations des différents types de programmeurs est bénéfique pour la société dans son ensemble : l'utilité, dans la situation bonne, est près de 14 % supérieure à celle obtenue dans la mauvaise. De plus, l'utilité procurée par chacun des types de projets est pratiquement identique. Cette situation peut s'expliquer par les autres paramètres : de mauvaises conditions impliquent aussi un faible intérêt de la part de la communauté Open Source (faible probabilité de distinction). Si cette dernière est moins intéressée par un projet ou un programmeur, la rétribution sera forcément moindre. En somme, on peut constater que le fait de permettre aux programmeurs de choisir le projet qui leur convient le plus est pertinent. L'utilité générée est supérieure, même s'il est considéré que chaque programmeur participe à un projet selon son propre talent, et que, selon la situation, il peut s'attendre à un gain en cas de succès plus faible.

On peut aussi remarquer que plus de programmeurs sont en mesure d'entreprendre un projet assujetti à la licence GPL lorsque la situation se dégrade. Autrement dit, en présence de mauvaises conditions, $\widehat{\theta}$ est plus faible. Si l'intérêt est plus faible (tant celui des programmeurs que celui de la communauté), que la probabilité de succès est plus basse et que les programmeurs sont moins talentueux, une plus grande proportion des programmeurs sera en mesure de s'attaquer à un projet difficile. La situation moyenne, par contre, présente la limite entre le talent faible et le talent fort la plus élevée, comme il a été mentionné plus haut.

Il faut comprendre que ce ne sont pas les programmeurs qui changent, mais le niveau de talent requis pour qu'un projet soit considéré comme difficile, ou qu'il soit assujetti à la licence GPL. Par exemple, si les conditions sont bonnes, un projet requérant un talent initial de $\theta = 0.8$ sera entrepris avec une licence BSD ; ce sera un projet facile. Lorsque les conditions sont moyennes, ce même projet sera considéré comme difficile et son responsable choisira plutôt une licence GPL.

Le tableau 4.6 présente les taux de variation des différentes variables par type de programmeur, selon les conditions. Autrement dit, un changement de classe de projet (dans le projet à deux licences) pour le « bon » programmeur est traité comme une variation de ce que reçoit ou constate ce programmeur lorsque les conditions changent. Il montre aussi que l'utilité décroît sensiblement plus rapidement dans le modèle à deux licences, mais le niveau est beaucoup plus haut, de sorte qu'il est toujours préférable d'avoir deux licences. Ceci laisse entendre que les programmeurs sont beaucoup plus sensibles aux différents paramètres lorsque leurs motivations sont prises en compte. La variation de l'effort qu'ils fournissent n'est apparament pas influencée par la présence d'une seconde licence ; la diminution lorsque les conditions se dégradent est à peu de choses près la même.

Un autre point à souligner est la variation du gain en cas de succès dans le modèle à deux licences. Contrairement à ce qui est attendu, les gains en cas de succès les plus élevés se retrouvent dans la situation moyenne. Autrement dit, afin de maximiser les gains réputationnels en cas de succès, l'intérêt des programmeurs comme celui de la communauté ne doivent pas être trop élevés, tout comme les probabilités de réussite et les chances de se démarquer. L'effort, par contre, va dans le sens attendu ; meilleures sont les conditions, plus grand sera l'effort fourni. Ceci concorde avec les statiques comparatives présentées au chapitre 3.

En résumé, de meilleures conditions amènent de meilleurs résultats ; ceci n'est évidemment pas un résultat en soi. Par contre, il est maintenant clair que, pour obtenir un effort optimal raisonnable, un certain intérêt de la part des programmeurs et de la communauté Open Source est requis, mais il ne doit pas être trop élevé. Lorsqu'il y a deux licences, l'effort optimal fourni est nettement supérieur. La présence de deux classes de projets permet de déterminer une proportion de programmeurs souhaitant envoyer un signal à l'industrie commerciale $(\widehat{\theta})$, et cette proportion semble logique étant donné les hypothèses sur les participants à la communauté Open Source.

Tableau 4.6 : Taux de variation intra-modèle
entre les différentes situations

variation / *(approx.)*	Modèle à une licence		Modèle à deux licences	
	B →Mo.	Mo.→Ma.	B→Mo.	Mo.→Ma.
$e^*(\theta = 0.85)$	-91.7%	-80.8%	-75%	-94%
$e^*(\theta = 0.5)$	-70.6%	-99%	-66.7%	-96.5%
$e^*(\theta = 0.25)$	-97.6%	-99.5%	-83.4%	-98.3%
$\overline{\theta}_G \quad \forall\, \theta$	0	0	$\overline{\theta}_{G_1}$: +11.9% $\overline{\theta}_{G_2}$: +4.6%	$\overline{\theta}_{G_1}$: -11.8% $\overline{\theta}_{G_2}$: -5.1%
$\overline{\theta}_B \quad \forall\, \theta$	+2.4%	+0.5%	-15.3%	+20%
$\overline{U}(\theta = 0.85)$	-0.8%	-0.3%	+0.7%	-4.6%
$\overline{U}(\theta = 0.5)$	-0.7%	-0.3%	-1.8%	-4.5%
$\overline{U}(\theta = 0.25)$	-0.6%	-0.3%	-1.7%	-4.5%

Légende : B = situation bonne ; Mo. = situation moyenne ; Ma. = situation mauvaise

Quant aux gains anticipés en cas de succès, s'il n'y a qu'une seule licence, ce gain est fixe. Autrement dit, s'il n'y a que la licence GPL en jeu, un programmeur peut envoyer un signal ou non. La présence de la licence supplémentaire permet d'envoyer un signal indiquant en plus la qualité du programmeur, puisqu'elle varie selon le contexte. Le gain en cas d'échec, pour sa part, est plus élevé si les conditions sont bonnes et qu'il n'y a qu'une seule licence, mais il varie de manière similaire dans les deux modèles. Ceci indique que les gains, lorsqu'il y a deux licences, sont adaptés à chacun. Un programmeur avec un talent faible ne doit pas espérer un gain (en cas de succès) très élevé, mais il est aussi vrai qu'un programmeur de très grand talent n'a pas à s'attendre à un faible gain. En somme, la présence d'une licence supplémentaire, qui permet de tenir compte des motivations des programmeurs, permet une plus grande utilité chez les programmeurs, donc pour la société dans son ensemble. Un aspect est toutefois surprenant : la proportion de projets qui devraient utiliser un licence BSD par rapport à une licence GPL est pratiquement l'inverse de ce que l'on retrouve dans la réalité. Il faut donc en déduire que les licences sont actuellement utilisées d'une manière contraire à ce qu'elles devraient l'être.

Enfin, un dernier commentaire sur la robustesse des résultats. Il a été mentionné à quelques reprises que les résultats obtenus dépendent en grande partie de la forme retenue pour la fonction d'utilité des programmeurs ainsi que de la distribution uniforme du talent parmi les programmeurs. Ceci fait en sorte que les équations obtenues pour les différentes variables d'intérêt (gains, limite entre un talent fort et un talent faible, etc.) sont non linéaires et d'une résolution difficile, mais n'affecte pas les résultats obtenus. Les statiques comparatives présentées dans les précédents chapitres sont aussi, en quelque sorte, une analyse de sensibilité ; l'impact des différents paramètres est évalué tout comme les valeurs maximales et minimales que ceux-ci peuvent prendre. Une forme plus générale de la fonction d'utilité et une distribution plus réaliste du talent dans la société aurait évidemment produit des résultats différents, mais, outre leur valeur, les résultats auraient probablement été similaires. Cette affirmation s'appuie sur deux points : premièrement les résultats obtenus correspondent en tous points à ce qui se retrouve dans la littérature, et deuxièmement, lorsqu'un tel exercice a été effectué sur un modèle similaire (BENABOU et TIROLE, 2003), les résultats obtenus concordaient avec les prévisions tirées du modèle général.

CONCLUSION

Ce mémoire a présenté deux modèles qui tentaient de démontrer que tenir compte des motivations des programmeurs participant à la communauté Open Source est bénéfique pour la société dans son ensemble. Dans un premier temps, il a été rappelé qu'à l'aube de l'informatique, la programmation en général était *de facto* Open Source. Les programmeurs de différents laboratoires s'aidaient, puisque le code écrit dans un laboratoire ne pouvait pas être utilisé ailleurs, les machines ne le permettant pas.

Puis, dans les années soixante-dix, le langage de programmation « C » a été créé, et a servi à faire le premier système d'exploitation dit « portable ». Celui-ci pouvant être utilisé sur tout type d'ordinateurs, des programmeurs ont commencé à s'approprier du code écrit par d'autres en le modifiant légèrement, afin d'en tirer profit. Ce faisant, les auteurs originaux n'avaient plus de droits sur leur propre œuvre.

Afin de s'opposer à ces gestes, la Free Software Foundation, en 1986, puis l'Open Source Initiative, en 1998, ont élaboré différentes définitions de ce que doit être un logiciel libre. En parallèle, des licences libres ont été élaborées afin de définir les « gauches d'auteurs » : il s'agit du pendant libre des droits d'auteurs. Ces licences diffèrent en trois points : un logiciel libre peut être redistribué, modifié et aussi utilisé différemment que ce pour quoi il est initialement conçu. Les licences commerciales donnent seulement le droit d'utiliser un logiciel selon la prescription de l'éditeur.

Environ 82% des logiciels libres, en 1998, utilisaient la licence GPL, alors que la licence BSD était utilisée par environ 7% des projets. Aujourd'hui, les projets employant la licence GPL représentent environ 46% des projets hébergés par sourceforge.net, alors que la BSD représente environ 5% des projets. Ce faible pourcentages place néanmoins la licence BSD au deuxième rang des licences libres en termes de popularité.

Selon la littérature économique traitant des logiciels libres, les programmeurs qui participent à un projet Open Source le font essentiellement pour deux raisons. Ils peuvent être motivés par leur future carrière, s'il n'ont pas de diplômes mais souhaitent tout de même signaler à l'industrie leur talent, ou ils peuvent programmer pour leur plaisir. Autrement dit, les motivations des programmeurs peuvent être extrinsèques ou intrinsèques. La communauté Open Source, actuellemnt, ne considère pas que ces motivations doivent être prises en considération ni pour la définition d'un projet, ni pour le choix d'une licence. Or, des études semblent indiquer le contraire.

Deux licences libres ont été étudiées ici : la GPL, avec son copyleft fort, et la BSD, au copyleft faible. Le copyleft fort implique une meilleure visibilité des programmeurs, puisqu'une telle licence est dite contaminante. La licence BSD, au contraire, n'insiste pas sur la visibilité des programmeurs puisqu'elle peut être enfouie à la fin des contrats de licences d'utilisations d'un logiciel commercial. C'est d'ailleurs un point de litige entre la FSF et l'OSI, cette dernière disant que la GPL est trop « radicale », qu'elle bloque dans une trop grande mesure les applications commerciales des logiciels libres. Le schisme entre ces deux organisations est donc plus au niveau idéologique que pratique.

Les deux licences mentionnées ont été utilisées afin de vérifier s'il est pertinent de tenir compte des motivations des programmeurs dans un modèle de décision d'effort, en considérant que peu importe leur motivation personelles, les programmeurs visent tous à avoir une meilleure réputation au sein de la communauté Open Source. Un continuum de programmeurs, différenciés par leur talent original, doivent choisir le niveau d'effort qu'ils fourniront, étant donnés les gains anticipés en cas de succès ou d'échec. Ces gains sont, eux, déterminés par rapport au talent du programmeur, anticipé par la communauté Open Source. Autrement dit, c'est un équilibre bayesien : l'ensemble de stratègie des programmeurs est constitué de l'effort optimal à fournir, et les paiements sont les gains anticipés en cas de succès ou en cas d'échec.

Il y a deux versions du modèle. La première version n'inclut qu'une seule licence (la GPL), et ne tient pas compte des motivations des programmeurs. La seconde version

inclut deux classes de projets, destinés aux différents types de programmeurs. Des projets avec une licence BSD sont destinés aux programmeurs qui ont un talent plus faible ; il est possible de qualifier ces projets de faciles, ne requérant pas de prouesses pour être menés à termes. Les autres projets, assujettis à une licence GPL, sont destinés aux programmeurs qui ont un talent élevé et qui souhaitent envoyer un signal à l'industrie commerciale. Ces projets peuvent aussi être considérés comme difficiles, ou comme demandant un fort talent pour être menés à bien.

Les deux modèles, une fois résolus, ont un comportement (en statiques comparatives) qui correspond bien à la réalité, et qui colle aux hypothèses de travail, que ce soit en ce qui a trait à l'effort optimal fourni par les programmeurs, des gains anticipés ou, dans le cas du modèle à deux licences, à la limite entre ce qui est considéré comme un talent fort ou un talent faible. Tant les niveaux que les variations suite à un changement de paramètres ont du sens.

En dehors des statiques comparatives, afin de comparer des modèles, trois situations ont été définies : une bonne, une moyenne et une mauvaise. Le but de cette étape est de permettre la variation de l'ensemble des paramètres et, dans tous les cas, la présence de la deuxième licence et des motivations des programmeurs est bénéfique pour la société dans son ensemble. La réaction des deux modèles à un changement de l'ensemble des paramètres est similaire, donc ce n'est pas qu'une question de sensibilité des modèles. Les niveaux atteints sont beaucoup plus élevés dans le second modèle, que ce soit par rapport à l'effort optimal ou à l'utilité générée pour l'ensemble des programmeurs. Cependant, il n'est pas possible d'affirmer hors de tous doute que l'auto-sélection des programmeurs s'effectue, puisque les niveaux d'effort et d'utilité sont les mêmes pour les deux classes de projets dans le modèle à deux licences.

Quant aux programmeurs, ils auraient tout intérêt à réviser l'utilisation qui est faite des différentes licences libres qui existent actuellement. Il a déjà été mentionné que les gestionnaires des différents projets Open Source privilégient grandement la GPL ; or, s'il n'existe que celle-ci, elle procure beaucoup moins d'utilité à la société. Qualifier

les différentes licences pour les différents types de programmeurs permettraient à la communauté de « diriger » chacun où il devrait être, donc d'employer les meilleurs éléments de la meilleure façon possible. À l'opposé, puisqu'il n'y a que les programmeurs eux-mêmes qui connaissent leur talent initial, chacun pourrait choisir de participer à un projet « à la mesure de son talent ».

Ces résultats peuvent même être appliqués aux décideurs publics : les divers paliers de gouvernements auraient apparamment intérêt à tenir compte des préférences et des motivations des divers membres de la population lorsque vient le temps d'élaborer des politiques touchant divers groupes. Plusieurs situations requérant l'action du gouvernement ont été mentionnées dans l'introduction. Dans tous les cas, il est nécéssaire d'au moins évaluer si les agents ont des motivations différentes pour entreprendre une activité et, le cas échéant, s'il est approprié d'en tenir compte.

Une parenthèse doit ici être ouverte. Tant les comportements présentés et analysés que les résultats déterminés auraient pu être obtenus par un modèle théorique plus simple. Par exemple, un modèle dans lequel le talent est binaire (élevé ou bas pour une probabilité donnée), l'effort est binaire (présent ou absent, pour un coût donné) et qui présente une probabilité de succès et de visibilité permettrait de produire des résultats similaires : une visibilité supérieure engendre une participation supérieure et des gains en cas de succès plus faibles (ce qui correspond *grosso modo* à la licence GPL). Un tel modèle permet de dégager les intuitions de bases, mais celles-ci sont aussi dispobibles auprès de programmeurs mêmes ; plusieurs études permettant de percevoir ces intuitions ont étées réalisées, qui ont été résumées par Tirole et Lerner (LERNER et TIROLE, 2002).

Le modèle qui a été developpé ici permet quant à lui non-seulement de dégager ces mêmes intuitions, mais aussi de vérifier et de quantifier leur sensibilité aux différents environnements susceptibles d'affecter tant la participation des programmeurs à un projet Open Source, que l'effort qu'ils fourniront, les résultats anticipés, leur intérêt et celui de leur pairs. De plus, les résultats présentés, en plus d'être plus précis, sont plus

riches : la présence de paramètres supplémentaires permet d'obtenir un plus grand réalisme.

La prochaine étape dans l'évolution du modèle développér ici sera d'affiner la fonction d'utilité afin que les divers types de projets soient plus réalistes. Par la suite, il faudra intégrer au modèle la possibilité pour les programmeurs de ne pas participer à la communauté Open Source, vérifier si les motivations des programmeurs sont aussi importantes une fois intégrées au modèle indépendemment des licences, et vice-versa. Ensuite, il faudra adapter la distribution du talent parmi les programmeurs, la distribution uniforme ayant été utilisée dans un premier temps afin de simplifier les calculs. Une distribution normale du talent pourrait être beaucoup plus réaliste.

APPENDICE A

RÉSOLUTION DU MODÈLE À UNE SEULE LICENCE

La résolution est effectuée telle que décrite à la section ??

A.1 Croyances en cas de succès- $\overline{\theta}_G$

$$p(G) = \int_0^1 s \cdot p(e^*(\theta), \theta) \cdot f(\theta) d\theta$$

où La distribution est uniforme, donc $f(\theta)$=1.

$$= \int_0^1 s \cdot \beta(\theta \cdot e^*(s, \theta)) \cdot d\theta$$

$$= \int_0^1 s \cdot \beta\left(\theta \cdot s\theta\left(\overline{\theta}_G - \overline{\theta}_B\right)\right) \cdot d\theta$$

$$= \int_0^1 \beta s^2 \cdot \theta^2\left(\overline{\theta}_G - \overline{\theta}_B\right) \cdot d\theta$$

$$= \beta s^2 \cdot \frac{\theta^3}{3}\left(\overline{\theta}_G - \overline{\theta}_B\right) \cdot \Big|_0^1$$

$$p(G) = \frac{1}{3}\beta s^2\left(\overline{\theta}_G - \overline{\theta}_B\right)$$

$$f(\theta|G) = \frac{s \cdot \beta p(e^*(\theta), \theta)}{\int_0^1 s \cdot \beta p(e^*(\theta), \theta) d\theta}$$

$$= \frac{s \cdot \beta\left(\theta \cdot s\theta\left(\overline{\theta}_G - \overline{\theta}_B\right)\right)}{\frac{1}{3}\beta s^2\left(\overline{\theta}_G - \overline{\theta}_B\right)}$$

$$= \frac{\beta s^2 \theta^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}{\frac{1}{3}\beta s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}$$
$$= \frac{s^2 \theta^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}{\frac{1}{3} s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}$$
$$f(\theta|G) = 3\theta^2$$

$$\overline{\theta}_G = \int_0^1 \theta \cdot f(\theta|G)\, d\theta$$
$$= \int_0^1 \theta \cdot 3\theta^2 d\theta$$
$$= \int_0^1 3\theta^3 d\theta$$
$$= 3\frac{\theta^4}{4}\Big|_0^1$$
$$\overline{\theta}_G = \frac{3}{4}$$

A.2 Croyances en cas d'échec - $\overline{\theta}_B$

$$p(B) = \int_0^1 \left[1 - s \cdot \beta p\left(e^*(\theta), \theta\right)\right] d\theta$$
$$= \int_0^1 \left[1 - s \cdot \beta\theta \cdot s\theta\left(\overline{\theta}_G - \overline{\theta}_B\right)\right] d\theta$$
$$= \int_0^1 \left[1 - \beta s^2 \theta^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)\right] d\theta$$
$$= \theta - \beta s^2 \frac{\theta^3}{3}\left(\overline{\theta}_G - \overline{\theta}_B\right)\Big|_0^1$$
$$p(B) = 1 - \frac{1}{3}\beta s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)$$

$$f(\theta|B) = \frac{\left[1 - s \cdot \beta p\left(e^*(\theta), \theta\right)\right]}{\int_0^1 \left[1 - s \cdot \beta p\left(e^*(\theta), \theta\right)\right] d\theta}$$

$$= \frac{1 - \beta s^2 \theta^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}{\int_0^1 \left[1 - \beta s^2 \theta^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)\right] d\theta}$$

$$= \frac{1 - \beta s^2 \theta^2 \left(\frac{3}{4} - \theta_B\right)}{1 - \frac{1}{3}\beta s^2 \left(\frac{3}{4} - \theta_B\right)}$$

$$\overline{\theta}_B = \int_0^1 \theta \cdot f(\theta|B) d\theta$$

$$= \int_0^1 \theta \cdot \frac{1 - \beta s^2 \theta^2 \left(\frac{3}{4} - \theta_B\right)}{1 - \frac{1}{3}\beta s^2 \left(\frac{3}{4} - \theta_B\right)}$$

$$= \int_0^1 \frac{\theta - \beta s^2 \theta^3 \left(\frac{3}{4} - \theta_B\right)}{1 - \frac{1}{3}\beta s^2 \left(\frac{3}{4} - \theta_B\right)}$$

$$= \left. \frac{\frac{\theta^2}{2} + \frac{1}{4}\beta s^2 \theta^4 \left(\theta_B - \theta_G\right)}{1 + \frac{1}{3}\beta s^2 \left(\theta_B - \theta_G\right)} \right|_0^1$$

$$\overline{\theta}_B = \frac{\frac{1}{2} + \frac{1}{4}\beta s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}{1 + \frac{1}{3}\beta s^2 \left(\overline{\theta}_G - \overline{\theta}_B\right)}$$

APPENDICE B

RÉSOLUTION DU MODÈLE À DEUX LICENCES

La résolution du modèle s'effectue comme celle du modèle de base.

B.1 Croyances en cas de succès du projet facile – $\overline{\theta}_{G_1}$

$$
\begin{aligned}
p(G_1) &= \cancel{F(\theta)} \int_0^{\widehat{\theta}} p(e_1^*(\theta).\theta) \frac{f(\theta)}{\cancel{F(\theta)}} d\theta \\
&= \int_0^{\widehat{\theta}} \beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) \cdot 1 \\
&= \frac{1}{3}\beta\theta^3 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\Big|_{\theta=0}^{\widehat{\theta}} \\
&= \frac{1}{3}\beta\,\widehat{\theta}^{\,3} k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)
\end{aligned}
$$

$$
\begin{aligned}
f(\theta|G_1) &= \frac{p(e^*(\theta),\theta)}{p(G_1)} \\
&= \frac{\beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)}{\frac{1}{3}\beta\,\widehat{\theta}^{\,3} k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)} \\
&= 3\frac{\theta^2}{\widehat{\theta}^{\,3}}
\end{aligned}
$$

$$
\begin{aligned}
\overline{\theta}_{G_1} &= \int_0^{\widehat{\theta}} \theta \cdot f(\theta|G_1) d\theta \\
&= \int_0^{\widehat{\theta}} \theta \cdot 3\frac{\theta^2}{\widehat{\theta}} d\theta \\
&= \int_0^{\widehat{\theta}} 3\frac{\theta^3}{\widehat{\theta}^3} d\theta \\
&= \frac{3}{4}\frac{\theta^4}{\widehat{\theta}^3}\Big|_{\theta=0}^{\widehat{\theta}} \\
\overline{\theta}_{G_1} &= \frac{3}{4}\,\widehat{\theta}
\end{aligned}
$$

B.2 Croyances en cas de succès du projet difficile – $\overline{\theta}_{G_2}$

$$
\begin{aligned}
p(G_2) &= \cancel{F(\theta)} \int_{\widehat{\theta}}^1 p(e_2^*(\theta), \theta) \frac{f(\theta)}{\cancel{F(\theta)}} d\theta \\
&= \int_{\widehat{\theta}}^1 \beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) \cdot 1 \, d\theta \\
&= \frac{1}{3}\beta\theta^3 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\Big|_{\theta=\widehat{\theta}}^1 \\
&= \left\{\frac{1}{3}\beta k_1 s_1 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\right\} - \left\{\frac{1}{3}\beta\,\widehat{\theta}^3 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right\} \\
&= \frac{1}{3}\beta\left(1 - \widehat{\theta}^3\right) k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)
\end{aligned}
$$

$$
\begin{aligned}
f(\theta|G_2) &= \frac{p(e_2^*(\theta), \theta)}{p(G_2)} \\
&= \frac{\beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)}{\frac{1}{3}\beta\left(1 - \widehat{\theta}^3\right) k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)} \\
&= \frac{3\theta^2}{1 - \widehat{\theta}^3}
\end{aligned}
$$

$$
\begin{aligned}
\overline{\theta}_{G_2} &= \int_{\widehat{\theta}}^{1} \theta \cdot f(\theta|G_2)d\theta \\
&= \int_{\widehat{\theta}}^{1} 3\frac{\theta^3}{1-\widehat{\theta}^3}d\theta \\
&= \frac{3}{4}\frac{\theta^4}{1-\widehat{\theta}^3}\Bigg|_{\theta=\widehat{\theta}}^{1} \\
&= \left\{\frac{3}{4}\frac{1}{1-\widehat{\theta}^3}\right\} - \left\{\frac{3}{4}\frac{\widehat{\theta}^4}{1-\widehat{\theta}^3}\right\} \\
\overline{\theta}_{G_2} &= \frac{3\left(1-\widehat{\theta}^4\right)}{4\left(1-\widehat{\theta}^3\right)}
\end{aligned}
$$

B.3 Croyances en cas d'échec de l'un ou l'autre projet – $\overline{\theta}_B$

)

$$
p(B) = \underbrace{\int_0^{\widehat{\theta}} \left[1 - p(e_1^*(\theta), \theta)\right] f(\theta)d\theta}_{p_1(B)} + \underbrace{\int_{\widehat{\theta}}^{1} \left[1 - p(e_2^*(\theta), \theta)\right] f(\theta)d\theta}_{p_2(B)}
$$

$$
\begin{aligned}
p_1(B) &= \int_0^{\widehat{\theta}} \left[1 - p(e_1^*(\theta), \theta)\right] \cdot 1 \, d\theta \\
&= \int_0^{\widehat{\theta}} 1 - \beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) \, d\theta \\
&= \theta - \frac{1}{3}\beta\theta^3 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\Big|_{\theta=0}^{\widehat{\theta}} \\
p_1(B) &= \widehat{\theta} - \frac{1}{3}\beta\,\widehat{\theta}^3 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)
\end{aligned}
$$

$$
\begin{aligned}
p_2(B) &= \int_0^{\widehat{\theta}} \left[1 - p(e_2^*(\theta), \theta)\right] \cdot 1 \, d\theta \\
&= \int_{\widehat{\theta}}^{1} 1 - \beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) d\theta
\end{aligned}
$$

$$= \theta - \frac{1}{3}\beta\theta^3 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\Big|_{\theta=\widehat{\theta}}^{1}$$

$$= \left\{1 - \frac{1}{3}k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right\} - \left\{\widehat{\theta} - \frac{1}{3}\beta\,\widehat{\theta}^3 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right\}$$

$$p_2(B) = \left(1 - \widehat{\theta}\right) - \frac{1}{3}\beta\left(1 - \widehat{\theta}^3\right)k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)$$

$$p(B) = p_1(B) + p_2(B)$$

$$= \left\{\widehat{\theta} - \widehat{\theta} - \frac{1}{3}\beta\,\widehat{\theta}^3 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\right\} + \left\{\left(1 - \widehat{\theta}\right) - \frac{1}{3}\beta\left(1 - \widehat{\theta}^3\right)k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right\}$$

$$p(B) = 1 - \frac{1}{3}\beta\left[\widehat{\theta}^3 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) + (1 - \widehat{\theta}^3)k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right]$$

$$f(\theta|B) = \begin{cases} f(\theta|B)_{\text{II}}: & \dfrac{1 - p\left(e_1^*(\theta), \theta\right)}{p(B)} \quad \text{si} \quad 0 < \theta < \widehat{\theta} \\ f(\theta|B)_{\text{III}}: & \dfrac{1 - p\left(e_2^*(\theta), \theta\right)}{p(B)} \quad \text{si} \quad \widehat{\theta} \le \theta < 1 \end{cases}$$

$$\overline{\theta}_B = \frac{\int_0^{\widehat{\theta}} \left[\theta \cdot f(\theta|B)_{\text{II}}\right] d\theta}{p(B)} + \frac{\int_{\widehat{\theta}}^{1} \left[\theta \cdot f(\theta|B)_{\text{III}}\right] d\theta}{p(B)}$$

$$\int_0^{\widehat{\theta}} \left[\theta \cdot f(\theta|B)_{\text{II}}\right] f(\theta) d\theta$$

$$= \int_0^{\widehat{\theta}} \theta \cdot \left[1 - \beta\theta^2 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\right] d\theta$$

$$= \int_0^{\widehat{\theta}} \theta - \beta\theta^3 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) d\theta$$

$$= \frac{1}{2}\theta^2 - \frac{1}{4}\beta\theta^4 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\Big|_{\theta=0}^{\widehat{\theta}}$$

$$= \frac{1}{2}\,\widehat{\theta}^2 - \frac{1}{4}\beta\,\widehat{\theta}^4 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)$$

$$\int_{\widehat{\theta}}^{1} \left[\theta \cdot f(\theta|B)_{\text{III}}\right] d\theta$$

$$= \int_{\widehat{\theta}}^{1} \theta \cdot \left[1 - \beta\theta^2 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right] d\theta$$

$$
\begin{aligned}
&= \int_{\widehat{\theta}}^{1} \theta - \beta\theta^3 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) d\theta \\
&= \frac{1}{2}\theta^2 - \frac{1}{4}\beta\theta^4 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) \Big|_{\theta=\widehat{\theta}}^{1} \\
&= \left\{\frac{1}{2} - \frac{1}{4}\beta k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right\} - \left\{\frac{1}{2}\,\widehat{\theta}^2 - \frac{1}{4}\beta\,\widehat{\theta}^4 k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right\} \\
&= \frac{1}{2}\left(1 - \widehat{\theta}^2\right) - \frac{1}{4}\beta\left(1 - \widehat{\theta}^4\right) k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)
\end{aligned}
$$

$$
\begin{aligned}
&\int_0^{\widehat{\theta}} \left[\theta \cdot f(\theta|B)_{\text{II}}\right] d\theta + \int_{\widehat{\theta}}^{1} \left[\theta \cdot f(\theta|B)_{\text{III}}\right] d\theta \\
&= \left\{\frac{1}{2}\,\widehat{\theta}^2 - \frac{1}{4}\beta\,\widehat{\theta}^4 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\right\} + \left\{\frac{1}{2}\left(1 - \widehat{\theta}^2\right) - \frac{1}{4}\beta\left(1 - \widehat{\theta}^4\right) k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right\} \\
&= \frac{1}{2} - \frac{1}{4}\beta\left[\,\widehat{\theta}^4 s_1^2 k_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) + \left(1 - \widehat{\theta}^4\right) s_2^2 k_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right]
\end{aligned}
$$

$$
\overline{\theta}_B = \frac{\frac{1}{2} - \frac{1}{4}\beta\left[\,\widehat{\theta}^4 s_1^2 k_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) + \left(1 - \widehat{\theta}^4\right) s_2^2 k_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right]}{1 - \frac{1}{3}\beta\left[\,\widehat{\theta}^3 k_1^2 s_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) + (1 - \widehat{\theta}^3) k_2^2 s_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right]}
$$

B.4 Détermination de $\widehat{\theta}$ – Fonctions d'utilité *ex post*

$$
\begin{cases}
\overline{U}_1^* = y - \frac{1}{2}\left(e_1^*(\theta)\right)^2 + \beta\left[p\left(e_1^*(\theta), \theta\right)\overline{\theta}_{G_1} + \left(1 - p\left(e_1^*(\theta), \theta\right)\right)\overline{\theta}_B\right] \\
\overline{U}_2^* = y - \frac{1}{2}\left(e_2^*(\theta)\right)^2 + \beta\left[p\left(e_2^*(\theta), \theta\right)\overline{\theta}_{G_1} + \left(1 - p\left(e_2^*(\theta), \theta\right)\right)\overline{\theta}_B\right]
\end{cases}
$$

$$
\begin{cases}
\overline{U}_1^* = y - \frac{1}{2}\left(\beta\theta k_1 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\right)^2 + \beta\theta k_1 \left[\beta\theta k_1 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)\right]\left(\overline{\theta}_{G_1} - \overline{\theta}_B\right) + \beta\overline{\theta}_B \\
\overline{U}_2^* = y - \frac{1}{2}\left(\beta\theta k_2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right)^2 + \beta\theta k_2 \left[\beta\theta k_2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)\right]\left(\overline{\theta}_{G_2} - \overline{\theta}_B\right) + \beta\overline{\theta}_B
\end{cases}
$$

$$
\begin{cases}
\overline{U}_1^* = y + \frac{1}{2}\beta^2\theta^2 k_1^2 \left(\overline{\theta}_{G_1} - \overline{\theta}_B\right)^2 + \beta\overline{\theta}_B \\
\overline{U}_2^* = y + \frac{1}{2}\beta^2\theta^2 k_2^2 \left(\overline{\theta}_{G_2} - \overline{\theta}_B\right)^2 + \beta\overline{\theta}_B
\end{cases}
$$

$$
\left\{\overline{U}_1^* = \overline{U}_2^*\right\} \Rightarrow g\left(\overline{U}^*, \theta\right)
$$

$$
\widehat{\theta} = \theta^*\left(g\left(\overline{U}^*, \theta\right)\right)
$$

APPENDICE C

DÉTERMINATION DE $\widehat{\theta}$ - RACINES RETENUES

$$\widehat{\theta_1} = \frac{-3k_2s_2 + 4\beta k_2s_2 + 3k_1s_1 - 4\beta k_1s_1}{9(k_2s_2 - k_1s_1)} +$$

$$(2^{1/3}\left(-9(k_2s_2 - k_1s_1)(-3k_2s_2 + 4\beta k_2s_2 + 3k_1s_1 - 4\beta k_1s_1) - (-3k_2s_2 + 4\beta k_2s_2 + 3k_1s_1 - 4\beta k_1s_1)^2\right)) /$$

$$\left(9(k_2s_2 - k_1s_1)\left(540k_2^3s_2^3 - 540\beta k_2^3s_2^3 - 144\beta^2k_2^3s_2^3 - 128\beta^3k_2^3s_2^3 - 891k_2^2k_1s_2^2s_1 + \right.\right.$$

$$1620\beta k_2^2k_1s_2^2s_1 + 432\beta^2k_2^2k_1s_2^2s_1 + 384\beta^3k_2^2k_1s_2^2s_1 + 162k_2k_1^2s_2s_1^2 - 1620\beta k_2k_1^2s_2s_1^2 -$$

$$432\beta^2k_2k_1^2s_2s_1^2 - 384\beta^3k_2k_1^2s_2s_1^2 + 189k_1^3s_1^3 + 540\beta k_1^3s_1^3 + 144\beta^2k_1^3s_1^3 + 128\beta^3k_1^3s_1^3 +$$

$$\sqrt{\left(\left(540k_2^3s_2^3 - 540\beta k_2^3s_2^3 - 144\beta^2k_2^3s_2^3 - 128\beta^3k_2^3s_2^3 - 891k_2^2k_1s_2^2s_1 + 1620\beta k_2^2k_1s_2^2s_1 + \right.\right.}$$

$$432\beta^2k_2^2k_1s_2^2s_1 + 384\beta^3k_2^2k_1s_2^2s_1 + 162k_2k_1^2s_2s_1^2 - 1620\beta k_2k_1^2s_2s_1^2 - 432\beta^2k_2k_1^2s_2s_1^2 -$$

$$\left.\left.384\beta^3k_2k_1^2s_2s_1^2 + 189k_1^3s_1^3 + 540\beta k_1^3s_1^3 + 144\beta^2k_1^3s_1^3 + 128\beta^3k_1^3s_1^3\right)^2 + 4\left(-9(k_2s_2 - k_1s_1)\right.\right.$$

$$\left.\left.\left(-3k_2s_2 + 4\beta k_2s_2 + 3k_1s_1 - 4\beta k_1s_1) - (-3k_2s_2 + 4\beta k_2s_2 + 3k_1s_1 - 4\beta k_1s_1)^2\right)^3\right)s_2\right)^{1/3}\Bigg) -$$

$$\frac{1}{92^{1/3}(k_2s_2 - k_1s_1)}\left(540k_2^3s_2^3 - 540\beta k_2^3s_2^3 - 144\beta^2k_2^3s_2^3 - 128\beta^3k_2^3s_2^3 - 891k_2^2k_1s_2^2s_1 + \right.$$

$$1620\beta k_2^2k_1s_2^2s_1 + 432\beta^2k_2^2k_1s_2^2s_1 + 384\beta^3k_2^2k_1s_2^2s_1 + 162k_2k_1^2s_2s_1^2 - 1620\beta k_2k_1^2s_2s_1^2 -$$

$$432\beta^2k_2k_1^2s_2s_1^2 - 384\beta^3k_2k_1^2s_2s_1^2 + 189k_1^3s_1^3 + 540\beta k_1^3s_1^3 + 144\beta^2k_1^3s_1^3 + 128\beta^3k_1^3s_1^3 +$$

$$\sqrt{\left(\left(540k_2^3s_2^3 - 540\beta k_2^3s_2^3 - 144\beta^2k_2^3s_2^3 - 128\beta^3k_2^3s_2^3 - 891k_2^2k_1s_2^2s_1 + 1620\beta k_2^2k_1s_2^2s_1 + \right.\right.}$$

$$432\beta^2k_2^2k_1s_2^2s_1 + 384\beta^3k_2^2k_1s_2^2s_1 + 162k_2k_1^2s_2s_1^2 - 1620\beta k_2k_1^2s_2s_1^2 - 432\beta^2k_2k_1^2s_2s_1^2 -$$

$$\left.\left.384\beta^3k_2k_1^2s_2s_1^2 + 189k_1^3s_1^3 + 540\beta k_1^3s_1^3 + 144\beta^2k_1^3s_1^3 + 128\beta^3k_1^3s_1^3\right)^2 + 4\left(-9(k_2s_2 - k_1s_1)\right.\right.$$

$$(-3k_2s_2+4\beta k_2s_2+3k_1s_1-4\beta k_1s_1)-(-3k_2s_2+4\beta k_2s_2+3k_1s_1-4\beta k_1s_1)^2\Big)^3\Big)s_2\Big)^{1/3}$$

$$\widehat{\theta}_2=\frac{-3k_2s_2+4\beta k_2s_2-3k_1s_1+4\beta k_1s_1}{9(k_2s_2+k_1s_1)}+$$

$$\Big(2^{1/3}\Big(-9(k_2s_2+k_1s_1)(-3k_2s_2+4\beta k_2s_2-3k_1s_1+4\beta k_1s_1)-(-k_2s_2+4\beta k_2s_2-3k_1s_1+4\beta k_1s_1)^2\Big)\Big/$$

$$\Big(9(k_2s_2+k_1s_1)\Big(540k_2^3s_2^3-540\beta k_2^3s_2^3-144\beta^2k_2^3s_2^3-128\beta^3k_2^3s_2^3+891k_2^2k_1s_2^2s_1-$$

$$1620\beta k_2^2k_1s_2^2s_1-432\beta^2k_2^2k_1s_2^2s_1-384\beta^3k_2^2k_1s_2^2s_1+162k_2k_1^2s_2s_1^2-1620\beta k_2k_1^2s_2s_1^2-$$

$$432\beta^2k_2k_1^2s_2s_1^2-384\beta^3k_2k_1^2s_2s_1^2-189k_1^3s_1^3-540\beta k_1^3s_1^3-144\beta^2k_1^3s_1^3-128\beta^3k_1^3s_1^3+$$

$$\sqrt{\Big(\Big(540k_2^3s_2^3-540\beta k_2^3s_2^3-144\beta^2k_2^3s_2^3-128\beta^3k_2^3s_2^3+891k_2^2k_1s_2^2s_1-1620\beta k_2^2k_1s_2^2s_1-}$$

$$432\beta^2k_2^2k_1s_2^2s_1-384\beta^3k_2^2k_1s_2^2s_1+162k_2k_1^2s_2s_1^2-1620\beta k_2k_1^2s_2s_1^2-432\beta^2k_2k_1^2s_2s_1^2-$$

$$384\beta^3k_2k_1^2s_2s_1^2-189k_1^3s_1^3-540\beta k_1^3s_1^3-144\beta^2k_1^3s_1^3-128\beta^3k_1^3s_1^3\Big)^2+4\,(-9(k_2s_2+k_1s_1)$$

$$(-3k_2s_2+4\beta k_2s_2-3k_1s_1+4\beta k_1s_1)-(-3k_2s_2+4\beta k_2s_2-3k_1s_1+4\beta k_1s_1)^2\Big)^3\Big)\Big)^{1/3}\Big)-$$

$$\frac{1}{92^{1/3}(k_2s_2+k_1s_1)}\Big(540k_2^3s_2^3-540\beta k_2^3s_2^3-144\beta^2k_2^3s_2^3-128\beta^3k_2^3s_2^3+891k_2^2k_1s_2^2s_1-$$

$$1620\beta k_2^2k_1s_2^2s_1-432\beta^2k_2^2k_1s_2^2s_1-384\beta^3k_2^2k_1s_2^2s_1+162k_2k_1^2s_2s_1^2-1620\beta k_2k_1^2s_2s_1^2-$$

$$432\beta^2k_2k_1^2s_2s_1^2-384\beta^3k_2k_1^2s_2s_1^2-189k_1^3s_1^3-540\beta k_1^3s_1^3-144\beta^2k_1^3s_1^3-128\beta^3k_1^3s_1^3+$$

$$\sqrt{\Big(\Big(540k_2^3s_2^3-540\beta k_2^3s_2^3-144\beta^2k_2^3s_2^3-128\beta^3k_2^3s_2^3+891k_2^2k_1s_2^2s_1-1620\beta k_2^2k_1s_2^2s_1-}$$

$$432\beta^2k_2^2k_1s_2^2s_1-384\beta^3k_2^2k_1s_2^2s_1+162k_2k_1^2s_2s_1^2-1620\beta k_2k_1^2s_2s_1^2-432\beta^2k_2k_1^2s_2s_1^2-$$

$$384\beta^3k_2k_1^2s_2s_1^2-189k_1^3s_1^3-540\beta k_1^3s_1^3-144\beta^2k_1^3s_1^3-128\beta^3k_1^3s_1^3\Big)^2+4\,(-9(k_2s_2+k_1s_1)$$

$$(-3k_2s_2+4\beta k_2s_2-3k_1s_1+4\beta k_1s_1)-(-3k_2s_2+4\beta k_2s_2-3k_1s_1+4\beta k_1s_1)^2\Big)^3\Big)\Big)^{1/3}$$

BIBLIOGRAPHIE

BENABOU, R. et J. TIROLE. 2003. « Intrinsic and extrinsic motivation », *The Review of Economic Studies*, vol. 70, no. 3, p. 489–520.

FITZGERALD, B. 2005. *Perspectives on Free and Open Source Software*, chapitre « 5 – Has Open Source Software a Future ? » *dans* Perspectives on Free and Open Source Software ». The Massachusetts Institute of Technology.

Johnson, J. P. 2001. Economics of open source software. disponible en ligne : `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.6659&rep=rep1&type=pdf`.

LAKHANI, K. R. et R. G. WOLF. 2005. *Perspectives on Free and Open Source Software*, chapitre « 1 – Why Hackers Do What They Do : Understanding Motivation and Effort in Free/Open Source Software Projects ». The Massachusetts Institute of Technology.

LERNER, J. et J. TIROLE. 2002. « Some simple economics of open source », *The Journal of Industrial Economics*, vol. 50, no. 2, p. 197–234.

LÉVÊQUE, F. e. Y. M. 2003. *Économie de la propriété intellectuelle*, no 375. coll. Repères.

Microsoft_Corporation. *Retail Software License Terms*. `http://www.microsoft.com/about/legal/useterms/`, accédé en ligne le 10 juillet 2008.

Mustonen, M. 2003. « Copyleft–the economics of linux and other open source software », *Information Economics and Policy*, vol. 15, no. 1, p. 99–121.

RAJAN, R. G. et L. ZINGALES. 1998. « Power in a theory of the firm », *The Quarterly Journal of Economics*, vol. 113, no. 2, p. 387–432.

RAYMOND, E. S. 2001a. *The Cathedral and the Bazaar*, chapitre « 1 – The Cathedral and the Bazaar ». O'Reilly Media Inc., revised edition édition.

———. 2001b. *The Cathedral and the Bazaar*, chapitre « 2 – Homesteading the Noosphere ». O'Reilly Media Inc., revised edition édition.

REFSNES DATA. *w3schools - OS Platform Statistics*. "`http://www.w3schools.com/browsers/browsers_os.asp`, accédé en ligne le 10 juillet 2008".

SourceForge Inc. 2010. Sourceforge.net. `http ://sourceforge.net/about`, accédé en ligne le 12 janvier 2010.

The Free Software Foundation. THE FREE SOFTWARE FOUNDATION. *General Public Licence 3.0*. `http://www.gnu.org/licenses/gpl.html`, accédé en ligne le 10 juillet 2008.

The_FreeBSD_Project. *Free BSD, The Power to Serve*. `http://www.freebsd.org/copyright/license.html`, accédé en ligne le 10 juillet 2008.

TIROLE, J. et J. LERNER. 2005. « The scope of open source licensing », *Journal of Law, Economics, and Organizations*, vol. 21, no. 1, p. 20–56.

WANG, L.-C. 1976. « Palo alto tiny basic », *Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, Running Light Without Overbyte*, vol. 1, no. 5, p. 12–25.

ZIMMERMAN, J.-B. et N. JULIEN. 2002. « Le logiciel libre : une nouvelle approche de la propriété intellectuelle », *Revue d'économie industrielle*, vol. 99, no. 1, p. 159–178.