N^o d'ordre :

Thèse

présentée à

l'Université de Bretagne Occidentale

pour obtenir le titre de

Docteur de l'Université de Bretagne Occidentale

Mention : *Sciences Économiques*

par

Nicolas Jullien,

sous la direction du professeur Lise Rochaix.

Impact du logiciel libre sur l'industrie informatique.

---

soutenue le 16 novembre 2001. devant la commission d'examen :

Composition du Jury :

Rapporteurs : Thierry Pénard, Professeur à l'Université de Rennes 1 ;
Jean-Benoit Zimmermann, Directeur de recherches CNRS,
EHESS (GREQAM, Marseille).

Examinateurs : Lise Rochaix, Professeur à l'Université de Marseille,
directrice de recherche ;
Patrick Cohendet, Directeur de recherches CNRS,
Université de Strasbourg ;
Godefroy Dang Nguyen, Professeur de l'ENST Bretagne ;
Jean-Michel Dalle, Maître de conférences associé à l'ENS de Cachan ;
Hervé Thouément, Professeur à l'Université de Bretagne Occidentale.

Travail réalisé au sein du département

Économie et Sciences Humaines de l'École Nationale Supérieure des Télécommunications de Bretagne.

L'Université n'entend ni approuver ni désapprouver les opinions contenues dans cette thèse.

Remerciements

Je voudrais remercier le département économie et sciences humaines de l'ENST Bretagne de m'avoir accueilli et de m'avoir fait confiance. Cela s'adresse particulièrement à Denis Phan, qui m'a proposé ce projet. Lise Rochaix-Ranson a accepté d'être ma directrice de thèse et j'espère qu'elle sait à quel point je lui en sais gré.

Je suis très reconnaissant à Jean-Benoît Zimmermann et à Thierry Pénard d'avoir accepté d'être les rapporteurs de ce travail, ainsi qu'à Patrick Cohendet, Jean-Michel Dalle et Hervé Thouément qui m'ont fait grand plaisir en participant à mon jury.

Si cette thèse ne s'est pas faite toute seule, elle n'a pas été non plus faite par un seul : Godefroy Dang Nguyen, Denis Phan, Thierry, Virginie, Myriam, Jérôme, dans le département, Adélaïde, Aymeric, David, Marie, Jean-Michel, à l'extérieur, m'ont aidé à l'écrire et à la corriger. Une pensée particulière à Nicole L'Hopital, qui, non contente de me relire, s'est aussi occupée des tirages et de la distribution, et à Jean-Michel Rouet, le créateur du style de présentation en Latex que j'ai utilisé.
Les erreurs restantes, et il en reste, sont bien sûr miennes !

Une thèse, c'est aussi des voisins de bureau, des amis, parfois rencontrés grâce à elle d'ailleurs, une famille enfin, qui, par leur soutien, leur ironie tendre, ont toujours été là pendant ces trois ans et demi. Merci à tous !

Et puis bien sûr, Adélaïde !

Table des matières

Liste des tableaux

Liste des figures

Introduction générale

LE logiciel libre, c'est-à-dire un logiciel distribué avec son code source[1] et avec l'autorisation de le modifier et de le redistribuer librement (à condition que cette redistribution soit faite aux mêmes conditions), est en train de devenir un des modèles économiques importants de l'industrie informatique. Il permet à ses utilisateurs de coopérer, essentiellement à travers Internet, en ajoutant des améliorations (la plupart du temps marginales), à un logiciel donné et en le redistribuant une fois modifié. De cette façon, chacun peut bénéficier rapidement des innovations apportées par tous les autres.

Et cette production coopérative de logiciels (Raymond [1998 a,b], Kollock [1999]), dans laquelle des producteurs se regroupent pour développer, ensemble, un projet, sans qu'il y ait de relations de subordination explicites (ou, plus précisément, contractuelles), financières ou non financières, n'est pas seulement un concept séduisant. Certains logiciels libres sont aussi des succès commerciaux : le serveur Web «Apache» domine son marché avec plus de 60 % de part de marché et le système d'exploitation Linux est le système utilisé par plus de 25 % des serveurs Internet. Certains observateurs annoncent même que ce système d'exploitation progresse plus vite que ses concurrents (25 % par an contre 10 % en moyenne). D'ailleurs, si Linux est, aujourd'hui, le logiciel libre le plus connu, c'est qu'il apparaît comme le plus grand challenger des systèmes d'exploitation de Microsoft, comme Windows 2000)[2].

Un tel succès doit interpeller les économistes, au moins sur deux points.

Intérêt du logiciel libre pour la science économique.

D'abord, nous devons mieux comprendre comment fonctionne le logiciel libre, autrement dit l'«économie du logiciel libre». C'est un problème délicat car nous ne sommes pas habitués à étudier des phénomènes qui ne relèvent pas du pur marché. Surtout que le «Libre», que nous définirons comme le système de production et de diffusion des lo-

[1] Un programme est réalisé dans un «langage», compréhensible par l'homme, qui fixe les règles d'écriture des instructions (les mots et leur signification en terme de tâche accomplie, la grammaire, c'est-à-dire la façon dont les mots s'articulent, etc.) Le «texte» ainsi produit est appelé le «code source» du logiciel. Ensuite, pour le faire fonctionner sur une machine, ce «texte» est «traduit» dans un langage accessible à la machine, grâce à des programmes spécifiques, appelés «compilateurs» ou «interpréteurs» (on obtient alors du code «compilé» ou «interprété»). La plupart des programmes sont fournis uniquement sous leur forme compilée, ce qui ne permet pas leur modification.

[2] Nous proposons, dans l'Annexe A, une étude de la diffusion des logiciels libres, de l'évolution de la production de tels logiciels, et notamment de l'implication des entreprises dans cette production.

giciels libres, ne semble pas être un système de production qui s'oppose au marché, à la production et à la distribution marchande de logiciel, mais un système qui réussirait à articuler une production non-marchande avec des activités marchandes complémentaires, comme le laisse supposer l'implication grandissante des producteurs marchands dans ce système (nous renvoyons, à nouveau, à l'Annexe A). Étudier le Libre devrait donc faire progresser notre compréhension des relations, des articulations entre structures de marché et structures non marchandes.

Pour cela, il faut comprendre ce qu'est le logiciel, comment il est produit. Le logiciel, qu'on peut apparenter dans un premier temps à un bien informationnel (au sens d'Arrow [1962]), est une forme d'écriture (Lévy [1992], p. 10), car il nécessite un langage d'écriture et il permet de produire un texte actif (un «texte qui agit», Callon [1991], p. 205). Ce texte actif représente pour Zimmermann [1995] «l'expression, dans un langage formalisé, d'un schéma de calcul constitué de l'articulation d'algorithmes», qui permet de «mettre en oeuvre un processus sans requérir [de la part de l'utilisateur] une maîtrise intellectuelle effective des composantes de ce processus». À cause de cela, on peut dire que programmer, créer du logiciel est une activité de codification de connaissance, à la fois connaissance d'un besoin, d'une tâche à automatiser ou à faciliter, et connaissance de la façon de réaliser cette tâche (voir Dréan [1996], p. 226). L'analyse du Libre (et plus généralement de l'économie du logiciel) doit donc s'appuyer sur les travaux qui ont étudié la façon dont sont produites les connaissances. Donc, et c'est la deuxième raison qu'il y a à s'intéresser à ce phénomène, le Libre renvoie directement aux interrogations des économistes sur la production de la connaissance et son étude devrait faire progresser les théories économiques émergentes. Même si Mowery [1996] note que la production non marchande a toujours représenté une part très importante de la production globale de code source, aujourd'hui la production industrielle, marchande de logiciel est largement dominante et c'est celle qui a été la plus étudiée en économie. Comprendre comment est produit le logiciel, c'est d'abord comprendre comment fonctionne cette industrie.

L'économie du logiciel.

Il n'existe qu'un petit nombre de travaux sur le sujet du logiciel. Mowery ([1996], p. 312), l'un des rares auteurs ayant travaillé sur le sujet, écrit d'ailleurs que «l'industrie du logiciel a peu attiré l'attention de manière surprenante, étant donné sa taille, sa croissance rapide et son importance de plus en plus grande dans les industries de haute technologie[3][4]». Cela tient sans doute d'abord à ses caractéristiques, qui en font un produit particulier. En effet, il se distingue de certains biens intangibles comme la musique ou l'oeuvre littéraire, qui n'ont pas d'objectifs utilitaristes. On pourrait le rapprocher des plans, des modèles, qui sont aussi des prototypes, mais contrairement à eux, le logiciel décrit un processus et permet de l'exécuter (Brousseau [1993], p. 207). Il se distingue aussi des services, puisque c'est un produit qui codifie une connaissance, une capacité technique, qui décrit un processus, sans que le producteur de cette capacité technique ait besoin d'intervenir une fois le transfert effectué. Une fois que cette connaissance est codifiée, on dispose d'un outil, dont il faut apprendre l'utilisation (il faut comprendre quelles tâches sont réalisables avec l'outil et la façon dont elles peuvent être réalisées), mais qui peut être produit et distribué quasiment sans coût car c'est un bien intangible, et qui est de plus en plus facilement reproductible.

Reste que le problème que pose la production du logiciel est celui du financement de la production-codification de ce bien «public», tel qu'à pu les définir Samuelson [1954, 1955][5]. Cette production se fait traditionnellement soit grâce à un financement public, soit par la construction d'un système juridique qui permet de créer un marché en rendant le bien exclusif et donc en permettant de rémunérer les producteurs par la facturation d'un droit d'usage (Coase [1959]).

À cause des spécificités du bien «logiciel», cette industrie possède quatre caractéristiques originales (Richardson [1997]), qui expliquent la constitution de monopoles,

[3]Nous reprenons la traduction de Horn ([2000b], p. 8).

[4]Citons tout de même, outre les travaux de Mowery, ceux de Lucas [1987], Richardson [1997], Zimmermann [1998] et Horn [2000].

[5]Un bien public est un bien non exclusif, c'est-à-dire qu'il n'est a priori pas possible d'empêcher quelqu'un de l'utiliser ce bien s'il a pu se le procurer, et non rival, c'est-à-dire que l'utilisation de ce bien ne le détruit pas. Or, il est facile et peut coûteux de se procurer un logiciel car il est reproductible à faible coût et car il n'y a pas de différence entre l'utilisation de l'original ou de sa copie. Et, bien sûr, l'utilisation d'un logiciel (ou de sa copie) ne le détruit pas.

mais aussi le dynamisme et le renouvellement continuel des entreprises productrices de logiciels. Conséquence directe du fait que c'est un bien «public», le coût de développement d'un logiciel n'est pas affecté par l'ampleur de la population des utilisateurs et le coût d'extension marginal de cette population est nul ou réduit à un montant négligeable vis-à-vis du coût de développement ; le rythme de l'innovation est élevé, ce qui réduit la durée de vie des produits ; ces deux caractéristiques engendrant une concurrence forte, parfois agressive au niveau des prix, pour imposer sa solution et pouvoir jouir d'une rente de monopole, les deux autres spécificités renforçant parfois ce type de concurrence, mais le limitant aussi : il existe des «effets de réseau», dans la mesure où un logiciel «n'est d'aucune utilité en lui-même, mais seulement quand il est mis en oeuvre conjointement avec d'autres produits complémentaires au sein d'un système». Les entreprises qui possèdent un logiciel sont alors tentées de développer les logiciels complémentaires. Mais, en même temps, elles peuvent difficilement couvrir l'ensemble du spectre de la demande et de nouvelles entreprises se créent sans cesse pour répondre à de nouveaux besoins. La conséquence en est que les standards jouent un rôle très important car ils permettent de mettre en oeuvre conjointement des produits complémentaires. Là encore, la définition des standards favorise l'entreprise qui contrôle leur évolution, elle peut plus facilement anticiper sur leur évolution et garantir l'interropérabilité des programmes complémentaires. D'un autre côté, le fait qu'elle ne puisse pas répondre à l'ensemble des demandes l'oblige à rendre publiques leurs caractéristiques.

La production «Libre» s'oppose à cette organisation marchande en ce sens qu'il n'y a pas de rémunération par la facturation d'un droit d'usage. Si elle a pu être comparée à la structure de production universitaire (Lerner & Tirole [2000], Horn [2000], Dang Nguyen & Pénard [2000], Foray & Zimmermann [2001]) parce que les producteurs sont souvent des utilisateurs du logiciel, qui contribuent librement et volontairement à son évolution, c'est une structure qui n'intègre pas de système de contrepartie financière directe (contrairement à la production universitaire, qui récompense par des promotions, des crédits, l'effort de ses membres, voir Dalle & Jullien [2000], [2001]). Pour expliquer ce phénomène particulier, Kollock [1998], Dang Nguyen & Pénard [2000] ont parlé d'un système de coopération, fondé sur le don-contre don : chacun contribue car il sait que les autres vont contribuer

aussi. Cela rapprocherait le problème de la production de logiciel des travaux de Cornes & Sandler [1991], qui ont étudié la notion de bien «club», défini comme étant un bien public accessible qu'à un petit nombre d'utilisateurs car il est partiellement exclusif ou partiellement rival (dans le cas du logiciel, ce sera toujours un bien partiellement rival[6]). Lerner & Tirole [2000], Foray & Zimmermann [2001] expliquent que, dans un tel groupe, les principaux producteurs sont reconnus, acquièrent de la réputation, qu'ils peuvent utiliser, soit pour initier des projets qui les intéressent, soit signaler leur qualité, leur capacité à initier et mener des projets, à de futurs employeurs. Ces auteurs montrent que ce système fonctionne quand il est à l'oeuvre dans un groupe relativement petit, relativement homogène, où les actions sont observables et où les besoins de chacun sont suffisamment proches pour que tous bénéficient de toutes les contributions. La diffusion du Libre hors du cercle d'utilisateurs-développeurs, la taille même de cette population de développeurs, l'investissement croissant des entreprises sont des phénomènes qui semblent en partie en contradiction avec ces explications. Plus précisément, soit ces explications sont justes, et le Libre devrait rapidement atteindre sa diffusion maximum (peu des utilisateurs des logiciels ont les connaissances suffisantes pour contribuer à leur développement), soit il y a d'autres explications à ces implications et il faut chercher ce qui a évolué dans les contraintes techniques, institutionnelles, personnelles, pour que ce phénomène apparaisse aujourd'hui.

Finalement, en essayant de replacer ce phénomène dans le cadre plus général de la production de logiciel, nous n'avons fait qu'illustrer le fait que «c'est en définitive l'optimisation des activités économiques contraintes par les règles de droit qui conduit à la définition des biens échangés et donc à leurs différentes propriétés, destruction ou non par l'usage, exclusion ou non d'usage, obligation ou non d'usage, La définition des biens est endogène et non exogène» (Laffont [1991], p. 42)[7]. Le Libre s'explique-t-il alors par

[6]Le nom de «club» vient du fait que certain utilisateurs peuvent se regrouper pour partager le coût de la production d'un tel bien dont ils se réservent l'usage (l'exemple le plus simple est sans doute celui d'installations sportives dont l'accès serait réservé au membres du club sportif). Comme le fait remarquer Callon [1994], pour que la connaissance soit réellement exploitable, il faut pouvoir la comprendre. À ce moment seulement, elle devient non rivale. Dans ce cas, il s'agirait de produire des logiciels qui ne seraient utiles, ou utilisables, que pour un petit nombre de personnes, qui coopéreraient pour les produire.

[7]En rappelant les règles de droit, Laffont met l'accent sur les autres éléments qui entrent en ligne de compte dans la définition d'une organisation économique et notamment les questions d'éthiques, sans doute encore plus présentes dans des domaines comme les biotechnologies. Nous verrons que les initiateurs du Libre étaient mus par des raisons éthiques autant que par des raisons que nous qualifierons d'«économiques», et que donc elles sont importantes pour expliquer le Libre, même si elles sont sans doute

une évolution de l'environnement technico-économique ? Dans ce cas, il faut expliquer en quoi cet environnement a évolué, pourquoi le Libre propose une réponse à cette évolution et pourquoi elle est plus adaptée que celles des organisations actuelles. C'est le but que nous nous fixons dans cette thèse.

Nous nous proposons de le réaliser de la façon suivante.

Plan de la thèse.

Dans le premier chapitre, nous allons déterminer les causes de l'apparition du Libre et de la diffusion des logiciels libres hors des «clubs» des utilisateurs-producteurs. Il s'agira donc d'économie industrielle et nous étudierons l'évolution de l'organisation industrielle de l'informatique. Nous montrerons deux choses : d'abord qu'il y a une tendance générale à la marchandisation du logiciel, tendance qui laisserait à penser que le Libre est un anachronisme ; mais qu'en même temps l'organisation industrielle a changé parce que les évolutions technologiques font évoluer les caractéristiques de la demande. Or, les progrès technologiques actuels, l'arrivée des réseaux (et notamment d'Internet) transforment la demande et il nous apparaîtra que les logiciels libres apportent des réponses nouvelles, adaptées à ces évolutions. Si l'on peut leur trouver des qualités, alors on doit s'interroger sur la viabilité du système qui les produit.

C'est la question que nous nous poserons dans le chapitre 2. Nous serons proches des interrogations de l'économie publique, puisqu'il s'agira d'étudier si le Libre peut être un système de production d'un bien public, alternatif aux systèmes marchand, «propriétaire», et non marchand, «universitaire», qui existent déjà. Nous montrons que c'est une organisation de production cohérente, pérenne, aussi bien à court qu'à long terme car elle incite les producteurs marchands à contribuer à la production de logiciels libres pour pouvoir répondre aux nouvelles demandes des utilisateurs. À partir de là, il faut se demander si c'est une organisation plus efficace que l'organisation actuelle, suffisamment plus efficace pour la supplanter.

Ce sera l'objet du chapitre 3. Nous retournerons à l'économie industrielle et nous

moins prépondérantes aujourd'hui.

montrerons qu'effectivement le Libre fonctionne mieux que le Propriétaire pour répondre aux besoins actuels et qu'il y répond suffisamment mieux pour que les utilisateurs soient prêts à faire l'effort de l'adopter (c'est-à-dire changer de logiciels quand ils en ont déjà un ou tout simplement de changer la façon dont ils adoptent un logiciel). Nous verrons qu'on peut distinguer deux phases dans la diffusion des logiciels libres et donc du Libre : une première phase qui concerne les entreprises et les logiciels techniques, une seconde qui concerne l'ensemble des utilisateurs et les logiciels qu'ils utilisent (système d'exploitation, logiciels de traitement de l'information personnelle, etc.)

Autant la première phase semble en cours, autant il y a peu de signes que la seconde soit entamée. C'est pourquoi les «outils» économiques utilisés pour tester nos hypothèses seront différents suivant les phases et pourquoi nous séparerons cette étude en deux chapitres.

Nous nous intéresserons dans le chapitre 4 à la première étape. Puisque cette diffusion est en cours, il était naturel d'essayer de valider empiriquement nos hypothèses sur l'implication des entreprises utilisatrices dans cette diffusion, sur la façon dont les fournisseurs construisaient leurs offres de services et sur les caractéristiques des relations utilisateurs producteurs. Nous avons donc mené une enquête auprès des entreprises françaises utilisant le libre pour construire leurs offres commerciales et c'est cette enquête que nous présentons dans ce chapitre. Nous constaterons que, globalement, même si la diffusion ne fait que débuter, elle a les caractéristiques attendues.

Nous nous intéresserons à la deuxième étape de la diffusion du Libre dans le chapitre 5. Pour réaliser cette analyse essentiellement prospective, nous avons modélisé ce que peut être un tel processus de diffusion grâce à un modèle de concurrence technologique. Ce modèle s'inspire de ceux proposés par Dalle [1995] et Dalle & Jullien [2000] et s'appuie sur les travaux de Farrell & Saloner [1985], Katz & Shapiro [1985, 1986], Arthur [1989] sur la concurrence entre technologies. Il va nous permettre de montrer que la deuxième étape de la diffusion du Libre est possible si les logiciels produits sont supportés par un noyau d'utilisateurs développeurs et par des entreprises. Nous utiliserons aussi le modèle pour identifier, suivant les caractéristiques des logiciels, les stratégies les plus efficaces pour que ces logiciels s'imposent et, en nous plaçant dans une perspective encore plus lointaine,

pour essayer de voir quel pourrait être le système de concurrence technologique dans le cas où ils se seraient imposés.

La conclusion que nous tirerons de ce travail est la suivante : dans le cas du logiciel, l'évolution technologique revitalise la production coopérative de connaissances et permet une nouvelle articulation entre cette production non marchande et une valorisation marchande. Plus qu'une collaboration public/privé, qui existait déjà, le Libre engendre un mécanisme, auto-entretenu, de collaboration marchand/non marchand, qui peut se montrer supérieur (au sens de Pareto), aux simples mécanismes de marché. L'exemple du Libre est-il transposable à d'autres secteurs de l'économie ? Nous laisserons ce point en suspens, mais l'exemple du logiciel nous fait penser qu'il y a lieu de l'étudier dans les industries basées sur la connaissance.

Premier Chapitre

Pourquoi du logiciel libre maintenant ? Une explication du renouveau de la production coopérative.

LES quelques chiffres présentés dans l'annexe A montrent que les logiciels libres représentent une part importante des logiciels produits et utilisés dans le monde. Dans ce premier chapitre, nous adoptons un point de vue d'économiste industriel; nous cherchons les causes de l'apparition du Libre et de la diffusion des logiciels libres hors des «clubs» des utilisateurs-producteurs qui les ont développés.

Nous constaterons dans la première partie que l'idée de produire de façon coopérative du logiciel, comme le propose le Libre, n'est pas une idée originale. C'est, en fait, le premier système de production qui ait existé. Ce qui l'est plus, c'est qu'il réapparaisse alors qu'il avait peu à peu disparu au profit du système de production propriétaire. Nous montrerons qu'Internet a joué un grand rôle dans cette résurgence.

Mais cette présentation historique ne nous apprendra pas pourquoi la production coopérative a disparu, ni si cette réapparition, initiée par Internet, va perdurer. Nous nous intéresserons donc, dans la deuxième partie, aux causes des évolutions de la production des logiciels. Ceci nous apprendra deux choses : que la disparition de celle-ci est due au fait qu'elle était de moins en moins capable de répondre aux demandes des utilisateurs, mais aussi qu'un nouveau système de production peut remplacer le système en place, s'il est mieux adapté à l'évolution de la demande.

Nous étudierons donc, dans la troisième partie, les évolutions récentes de la demande et nous nous demanderons en quoi le Libre se trouve en phase avec ces évolutions.

1.1 Le Libre, dernier avatar de la production coopérative.

Nous allons d'abord montrer comment la production coopérative, dominante au début de l'informatique, s'est peu-à-peu marginalisée. Nous nous poserons alors, dans la deuxième section de cette première partie, la question de sa résurgence et du succès de son dernier avatar, le Libre. Nous rappellerons le rôle qu'Internet a joué dans ce renouveau.

1.1.1 La production coopérative : une longue tradition, un déclin continu.

Nous allons voir que la production coopérative de logiciel, qui est la pratique courante dans le monde de la recherche, est devenue de plus en plus marginale, au fur et à mesure du développement de la production industrielle d'ordinateurs, sans jamais disparaître complètement.

La production coopérative, le système originel de production du logiciel.

Au début de l'informatique, dans les années 50, la production informatique était pratiquement exclusivement une production américaine. L'activité informatique était très liée aux projets militaires, aux financements militaires et aux projets de recherche : l'État américain a financé la recherche et le développement de l'informatique à ses débuts et la plupart des machines produites à cette époque l'on été pour ce «client»[1].

Les machines étaient livrées «nues» par le fournisseur, c'est-à-dire sans programmes. Le fournisseur aidait son client à mettre en œuvre ses machines et à développer les programmes qui lui permettaient de les utiliser. Le logiciel était alors considéré comme un outil et un objet de recherche. Sa production reposait sur des relations de collaboration poussées entre centres de recherche publics et industrie (Rosenberg [1992]) et les principaux utilisateurs de logiciel étaient alors les principaux producteurs (les universités, les grandes entreprises, surtout celles liées à l'industrie de défense et, bien sûr, le principal producteur de machines informatiques, IBM).

L'ensemble des logiciels produits était disponible, libre de droits de propriété ; la notion même de protection intellectuelle du logiciel n'existait pas. La seule façon de protéger un logiciel était de le garder secret, ce qui n'était pas la stratégie soutenue par les institutions publiques et par les entreprises informatiques émergentes. En effet, la diffusion de logiciels dans le domaine public était directement bénéfique aux producteurs : plus la quantité de

[1] La part du financement étatique de la recherche des sociétés américaines entre 1949 et 1959 est estimée à 59 %, en moyenne pondérée (Genthon [1995], p. 33). Cette part va de 46 % pour l'entreprise RCA à 90 % pour Raytheon. Genthon [1995] estime que le parc public, y compris le parc militaire, représentait encore, en 1962, 40 % du parc installé en valeur.

logiciels disponible était grande, plus l'étaient les incitations à acheter un ordinateur. De plus, le département américain de la défense (DoD) a encouragé ces pratiques en finançant des programmes de recherche et de développement coopératifs universités-entreprises. Il a aussi favorisé la diffusion des «bonnes pratiques» de programmation[2]. Enfin, le milieu de la recherche est traditionnellement un milieu favorable à la coopération et à la diffusion d'informations sur les connaissances produites (Dasgupta & David [1994]).

On le voit, la production coopérative n'est pas neuve. Elle a même dominé l'informatique pendant longtemps, puisque l'apparition d'un droit de propriété intellectuelle sur le logiciel, qui permet de le vendre et surtout d'empêcher ses utilisateurs de le diffuser, date de 1969. Finalement, si l'on se place du point de vue des premiers utilisateurs des ordinateurs, ce qui étonne est que cette production coopérative se soit marginalisée alors qu'elle semblait satisfaire l'ensemble des acteurs, producteurs et utilisateurs de logiciels. C'est pourtant ce qui s'est passé.

Une production coopérative de plus en plus marginale, même si elle reste importante pour l'innovation en informatique.

Initié dans les années soixante, le marché des logiciels et des services se développe vraiment à partir des années soixante-dix. Le système de production coopératif et hors marchand est alors petit à petit occulté. Cela ne veut pas dire qu'il disparaît : Unix, le système d'exploitation des stations de travail et de nombreux serveurs informatiques, est développé de façon collaborative par des entreprises et des universités[3] au cours des années soixante-dix.

Mais il y a une séparation de plus en plus nette entre le système de production, coopératif, qui peut initier des projets, et le système qui distribue et adapte le projet

[2]«The US armed forces, from the earliest days of their support for the development of computer technology, were surprisingly eager for technical information about this innovation to reach the widest possible audience, [...]» (Langlois & Mowery [1996], p. 58).

[3]Unix est un système d'exploitation. Il est inspiré du projet de recherche «Multix» qui regroupait les Bell Labs d'ATT, le MIT et General Electric, avec des financements fédéraux. Faute de ressources, il est abandonné en 1968. Il est repris par deux informaticiens des Bell Labs, Dennis Ritchie et Ken Thompson, rejoints en 1970 par Brian Kernighan. La première version documentée d'Unix a été terminée en 1971 et diffusée librement (ATT, du fait de son monopole dans les télécommunications, n'avait pas le droit d'avoir des activités commerciales en informatique). Elle s'impose rapidement dans les universités d'informatique, qui contribuent fortement à son évolution (principalement l'université de Berkeley, à partir de 1977, voir McKusick [1999]).

aux besoins des utilisateurs. À un certain moment, ces logiciels sont privatisés par des entreprises qui en assurent la distribution et l'évolution. Dans le cas d'Unix, ATT fut autorisée à entrer sur le marché de l'informatique en 1984 et elle a alors demandé le paiement de ses droits sur le nom Unix et sur le système dont elle est à l'origine ; dès le début des années 80, l'entreprise SUN utilisait, pour faire fonctionner ses stations de travail, un système d'exploitation qu'elle avait adapté d'une version issue de Berkeley, système dont elle se réservait les possibilités d'évolution.

Si le système de production coopératif est toujours à la base du développement de nombreux logiciels, le lien entre la demande (les utilisateurs) et les producteurs n'est plus direct, mais passe par l'intermédiaire des entreprises. Il y a d'un côté la recherche, coopérative, productrice de prototypes et financée par des subventions ou des commandes publiques et de l'autre l'industrie, qui s'approprie ces prototypes et qui les distribue.

Cette tendance s'affirme dans les années quatre-vingt dix : ces années correspondent à un relatif désengagement des institutions fédérales américaines du financement des projets logiciels et donc à une plus grande difficulté pour initier et financer de tels projets. De plus, la recherche prend une orientation plus théorique (algorithmique, tests de stabilité des programmes, etc.) Considérant l'orientation de plus en plus abstraite des projets de recherche et la baisse des commandes publiques, il apparaît normal que les industriels abandonnent la production coopérative, qui n'est plus que l'apanage de la production universitaire. Pourtant, il ne faudrait pas croire que l'appropriation, l'adaptation et la diffusion par les entreprises des créations des centres de recherche aient cessé. Nous en voulons pour preuve les exemples suivants : les centres de recherche ont continué à développer des logiciels, à produire de nouveaux concepts comme le langage objet (qui est à la base de Java), qui ont ensuite été adoptés par les industriels. Les années 90 sont celles de la diffusion dans les entreprises d'Internet, système de réseau développé par l'université. Cette diffusion a été facilitée par une invention issue d'un centre de recherche, le CERN, le langage HTML.

Il semble que le système de production du logiciel se soit transformé pour, finalement, aboutir à un schéma classique : les centres de recherche publics inventent des nouveaux

outils et de nouveaux concepts et les industriels les transforment en objets utilisables, en se les appropriant. Le phénomène Libre ne s'inscrit donc pas dans le sens de l'histoire. Est-il pour autant anachronique, est-ce seulement un épiphénomène ? Pour répondre à cette question, il faut d'abord comprendre pourquoi il est réapparu. Nous allons montrer que c'est essentiellement dû à l'arrivée d'Internet.

1.1.2 Un renouveau de la production coopérative grâce à Internet ; épiphénomène ou tendance de fond ?

Une des dernières évolutions importantes de l'informatique est la diffusion d'Internet, à partir de 1991. Cette diffusion explique en grande partie le phénomène Libre, parce qu'elle a permis la diffusion des logiciels libres, mais aussi parce qu'Internet facilite la production coopérative.

La diffusion des logiciels libres a accompagné celle d'Internet.

La diffusion des logiciels libres a accompagné celle d'Internet pour deux raisons : parce qu'ils répondaient à la demande créée par Internet et parce que les producteurs ont trouvé là un réseau permettant de les faire connaître.

Une diffusion qui répond à la demande créée par Internet.

Les logiciels à la base d'Internet (comme TCP/IP), notamment des systèmes d'exploitation libres comme GNU/Linux, FreeBSD ou OpenBSD ont pour la plupart été développés dans les universités et les centres de recherche publics[4]. Aujourd'hui encore, la grande majorité des logiciels d'Internet sont des logiciels libres[5]. Lorsqu'Internet s'est diffusé, se sont aussi diffusés les logiciels libres qui permettaient de le faire fonctionner.

Le fait que les personnes qui installaient les premiers serveurs Web (fournisseurs d'accès à Internet ou responsables informatiques dans les entreprises) étaient souvent des infor-

[4]Les Unix libres BSD ont été développées à partir de l'Unix de Berkeley (BSD signifie «Berkeley Software Distribution»).
C'est au CERN qu'a été développé le langage HTML et notamment le système des liens hypertextes.

[5]Le site http ://www.netaction.org/articles/freesoft.html rappelle l'importance des logiciels libres dans le fonctionnement d'Internet, en expliquant tous les services qui ne fonctionneraient pas si les logiciels libres n'existaient pas.

maticiens a facilité cette diffusion : ils avaient le bagage technique et l'habitude d'aller trouver des logiciels et de l'aide sur le réseau. Ils avaient souvent des moyens financiers limités et la gratuité des logiciels libres devait également être appréciée.

Les logiciels libres ont commencé à être connus en dehors de cette sphère spécialisée parce que des utilisateurs se sont aperçus qu'ils répondaient à leur demande de nouveaux services, autour d'Internet et des réseaux informatiques (qui sont souvent basés sur les mêmes technologies qu'Internet) : serveurs de fichiers, serveurs d'imprimante (avec les logiciels GNU/Linux et Samba, principalement)[6]. Le succès des logiciels libres repose donc d'abord sur une adéquation historique à de nouveaux marchés. Ils ne se sont pas imposés face à d'autres logiciels pour répondre à de nouveaux besoins mais se sont diffusés en même temps que les besoins auxquels ils répondaient.

Une diffusion qui profite d'un nouveau système d'information, Internet.

Enfin, la diffusion d'Internet est aussi la diffusion d'un réseau d'information. Ce réseau était utilisé par les producteurs de logiciels libres, qui y avaient accès dans les universités ou dans les centres de recherche. Ils ont pu utiliser l'ensemble des nouveaux supports de diffusion que permet Internet (listes de diffusions, news, sites Web) pour faire connaître ces logiciels, pour les rendre plus accessibles à de nouveaux utilisateurs.

Mais la diffusion d'Internet n'a pas seulement permis la diffusion des logiciels existants. Elle a aussi rendu plus facile leur production.

Internet facilite la production coopérative.

L'existence de réseaux physiques comme Internet, plus généralement de lieux d'échanges, est nécessaire pour la coopération : ce sont eux qui permettent l'échange de fichiers, donc des contributions, et la tenue de discussions sur les évolutions des logiciels. Il suffit, pour s'en convaincre, de se rappeler que le réseau ARPAnet, ancêtre d'Internet, a été financé par l'ARPA, une agence du ministère de la défense des États-Unis chargée d'initier et de financer des projets en informatique, pour favoriser les échanges d'idées et

[6]En sus des chiffres que nous avons donnés en introduction, nous renvoyons à l'interview de M. Prial, responsable marketing Linux et Solution chez IBM dans Décision Micro & Réseau, daté du 22 mars 1999.

de logiciels entre universités et industries travaillant sur le logiciel[7].

Ces réseaux existaient avant Internet, ce sont eux qui ont permis d'initier les projets libres et de faire se rencontrer des développeurs qui avaient des besoins proches[8]. Cependant, les débits de ces réseaux étaient faibles et les services assez pauvres. Il est certain que la diffusion hors de la sphère de la recherche d'Internet a permis aux développeurs de disposer d'un outil de coordination plus efficace : c'était le premier système qui leur offrait un accès simple à l'information et aux systèmes de stockage, de n'importe où (et notamment de leur domicile), pour un coût modique[9]. Internet a aussi permis la multiplication de forums spécialisés et a augmenté la probabilité que des utilisateurs-développeurs ayant des besoins spécifiques proches se rencontrent et atteignent la «masse critique» qui permet d'initier un projet[10].

Le renouveau de la production coopérative est donc dû non seulement à l'adoption des logiciels produits par une grande population d'utilisateurs (que nous avons qualifié d'adéquation historique à de nouveaux marchés), mais aussi à un regain certain du volume de logiciels produits de façon coopérative (nous renvoyons aux chiffres de l'introduction). On peut alors se demander si ces deux phénomènes ne sont pas les manifestations complémentaires d'une réalité, celle de la diffusion d'un nouveau modèle de production, le Libre.

La diffusion des produits ne veut pas dire la diffusion de leur système de production.

Il serait hasardeux de conclure que, parce que les logiciels libres se diffusent, le Libre est adopté. En effet, ce n'est pas la première fois que des logiciels développés coopérativement

[7] «The ARPANET was the first transcontinental, high-speed computer network. [...] It enabled researchers everywhere to exchange information with unprecedented speed and flexibility, giving a huge boost to collaborative work and tremendously increasing both the pace and intensity of technological advance». (Raymonds [2000])

[8] «Unix even had its own networking, of sorts, UUCP : low-speed and unreliable, but cheap. [...]In 1980 the first USENET sites began exchanging broadcast news, forming a gigantic distributed bulletin board tha would quickly grow bigger than ARPANET.» (Raymond [2000])

[9] «The early growth of Linux synergized with another phenomenon : the public discovery of the Interet [...] Following the invention of the World-Wide Web, the Internet's already-rapid growth accelerated to a breakneck pace.» (Raymond [2000])

[10] Van Alstyne & Brynjolfsson [1997] en ont proposé une modélisation tout à fait intéressante.

sont adoptés par le monde marchand et privatisé. Après tout, Unix a été développé dans le monde coopératif, par des universités et des centres de recherche. Cela n'a pas empêché SUN, puis IBM, HP, Digital, etc. de l'utiliser et d'en proposer des versions propriétaires. Plus près de nous, des logiciels libres comme FreeBSD (un descendant libre de l'Unix de Berkeley) a été approprié par Apple qui en a fait le système d'exploitation MacOS X.

Le phénomène Unix va-t-il se reproduire? C'est la thèse que défendent Genthon & Phan [1999], c'est ce que craint Horn [2000b]. Ils expliquent que les entreprises risquent de se ré-approprier la production de ces logiciels soit en les fermant, comme le fait Apple, soit en publiant des versions de moins en moins compatibles de certain logiciels phares, comme peuvent le faire RedHat ou SuSE avec GNU/Linux. Si, par exemple, les utilisateurs préfèrent les versions propriétaires que les producteurs développent à partir de ces logiciels, la différence entre le Libre et la sphère de la recherche ne sera pas bien grande. Tous deux seront des systèmes chargés de fournir aux producteurs des idées, des concepts et des prototypes de nouveaux logiciels. Horn [2000b] va même jusqu'à considérer que le Libre et la production de la recherche relèvent de la même logique de production, qu'il appelle «monde de la création». Ce sont, pour lui, des systèmes de production non marchands dont les produits nécessitent une marchandisation pour être diffusée.

Il faut donc affiner l'analyse, en se demandant en quoi le Libre répond à des besoins non satisfaits par le système de production actuel et notamment pourquoi l'organisation coopérative pourrait être à nouveau la plus adaptée pour répondre aux besoins des utilisateurs de l'informatique. Et pour cela, il est utile de comprendre pourquoi et comment a évolué le système de production des logiciels. C'est à quoi nous consacrons la deuxième partie de ce chapitre.

1.2 L'évolution du système de production des logiciels : un phénomène rendu possible par les progrès technologiques, mais choisi par les utilisateurs.

Dans cette partie, nous allons montrer deux choses : d'abord que c'est le progrès technologique qui a rendu possible l'évolution du système de production, d'une part en

faisant évoluer la demande, d'autre part en changeant la façon dont les ordinateurs sont construits ; ensuite que ce sont les utilisateurs qui ont sélectionné les organisations de productions les plus adaptées à leurs besoins et que, comme ces utilisateurs ont évolué dans le temps, l'organisation de production aussi.

1.2.1 Les progrès technologiques en informatique permettent d'élargir la demande et de faire évoluer l'organisation de la production.

Partant de la structure de l'ordinateur, nous allons constater que sa spécificité réside dans le fait qu'il est constitué d'une partie matérielle, d'une part, et de programmes qui utilisent les éléments matériels pour conduire des calculs, d'autre part. Le progrès technologique peut alors se faire à deux niveaux : celui du matériel, des composants (mémoires, disques durs, etc.) et celui de l'«immatériel», c'est-à-dire l'assemblage de ces composants et de leur fonctionnement (l'architecture et le logiciel). Et nous allons montrer que si le progrès technologique sur le matériel a eu comme principale conséquence la démocratisation de la machine informatique, c'est le progrès sur l'immatériel qui explique les évolutions de l'organisation industrielle en informatique[11].

L'ordinateur, une machine à calculer programmable.

L'ordinateur est une machine à calculer et, de ce point de vue, il descend des machines inventées par Pascal ou Leibniz. Mais, contrairement à ces premières calculatrices, où c'est l'homme qui transmet ses ordres et conduit le calcul, l'ordinateur réalise lui-même les travaux demandés, à partir du moment où on lui a indiqué comment les mener[12]. Son principe fondamental repose sur le fait qu'il existe des séquences de calcul, prévisibles, qu'on peut décrire et inscrire dans une machine. Celle-ci sera alors capable de les reproduire seule et de l'appliquer aux données qu'on lui fournira. L'ordinateur possède une structure,

[11] Notre compréhension de l'histoire de l'informatique doit beaucoup aux ouvrages de Breton [1990], Genthon [1995] et de Dréan [1996]. L'analyse de l'organisation de l'industrie informatique s'appuiera sur le travail de Gérard-Varet & Zimmermann [1985], initié dans Delapierre & al. [1980]. Leur point de vue théorique a été précisé par Zimmermann [1989, 1995b] et l'analyse industrielle par Delapierre & Zimmermann [1994].

Enfin, notre analyse de l'économie et de l'industrie du logiciel doit beaucoup à Horn [2000b], qui nous apparaît comme l'ouvrage de référence dans le domaine.

Nous encourageons le lecteur désireux d'approfondir ces sujets à se reporter à ces travaux.

[12] Breton [1990] fait l'analogie avec l'automobile en disant que cela correspondrait à une voiture à qui l'on indiquerait la destination et qui déciderait elle-même de son parcours.

une «unité de commande interne», qui permet de réaliser ces programmes en dirigeant les composants de la machine chargés des calculs. À partir de là, on peut théoriquement réaliser n'importe quelle tâche à base de calculs, et ce, indépendamment de la personne qui en a imaginé la séquence. L'ordinateur est un «automate universel à algorithme enregistré» (Breton [1990], p. 94).

Ce principe pose une distinction entre la machine, qui peut être la même pour tous, et l'utilisation qu'on en fait, qui peut être particulière et qui n'est déterminée et limitée que par la capacité à réaliser des programmes (on reste contraint par les limites de la technologie). On comprend alors l'importance du progrès sur les composants dans l'amélioration des programmes : plus ils sont efficaces pour exécuter les calculs demandés, plus on peut imaginer des programmes sophistiqués. Et c'est effectivement grâce au progrès des composants que les machines se sont démocratisées.

Des progrès technologiques sur le matériels qui ont permis de démocratiser l'outil informatique.

Rappelons d'abord les grandes étapes du progrès technologique dans les composants. La première machine de ce type est basée sur les travaux théoriques de Turing, et est réalisée par une équipe de chercheurs basée aux États-Unis à la Moore School, Eckert, Mauchly et von Neumann en 1945. La plupart des ordinateurs sont toujours un assemblage de circuits logiques, binaires (oui ou non) qui, assemblés, permettent de réaliser les différentes opérations de calcul et ils sont donc très proches de cette machine initiale[13]. Seule la façon de construire ces circuits a évolué : les constructeurs ont successivement utilisé des lampes à vide, puis des transistors, et enfin une technologie propre à l'informatique, le circuit imprimé. Petit à petit, tous les composants ont été réalisés en circuits imprimés. Enfin, grâce à la miniaturisation, ils ont fini par être intégrés dans un même circuit. Ce sont les circuits intégrés[14]. Cette évolution n'a pris que 25 ans et a permis de faire passer le volume des ordinateurs de plusieurs mètres cubes à quelques centimètres cubes et leurs poids de plusieurs tonnes à à peine plus d'un kilogramme pour les plus légers.

[13] On parle de machines du type «von Neumann».

[14] Le micro-processeur est, par exemple, un circuit sur lequel sont intégrées toutes les composantes de l'ordinateur (mémoire, organe de commande et organe de contrôle) permettant de réaliser les calculs.

Si l'on comprend bien qu'il est plus facile d'avoir chez soi une machine de la taille d'une télévision que de celle d'un garage, encore faut-il que le prix ait aussi évolué en conséquence. Non seulement cela a été le cas[15], mais cela s'est accompagné d'une augmentation extrêmement rapide de la puissance (c'est-à-dire, grossièrement, de la vitesse de calcul des machines), comme on peut le constater dans le tableau 1.

Tableau 1.1 — Évolution des coûts de la performance informatique.

$/Mips[a]	**Micros**	**mainframes**	**ratio**
1990	700	100000	> 140
1995	60	55000	>900

$/MBytes[b]	**Micros**	**mainframes**	**ratio**
1990	90	6000	>65
1995	6	1500	>250

source : Dréan˜[1996], p. 50.

[a]Mips : million d'instructions par seconde ; unité de vitesse de calcul.

[b]MBytes : million d'octets ; unité de capacité de mémoire.

Grâce à cela, le nombre d'ordinateurs est passé de quelques-uns à plusieurs centaines de millions (cf. le tableau n°2).

Tableau 1.2 — Diffusion de l'ordinateur au cours du temps.

Parc de machines (en milliers)	**1960**	**1970**	**1980**	**1990**	**2000**
dans le monde	$\simeq 6$[a]	$\simeq 50$			> 200 000
connectées à Internet[b]			0,213	160	> 56 000

[a]Genthon [1995].

[b]http ://www.info.univ-angers.fr/pub/pn/internet/sld004.htm et http ://www.industrie.gouv.fr/observat/innov/so_tbi5.htm.

Nous allons montrer que ce passage n'a été rendu possible que parce que de nouveaux logiciels ont été développés, qui permettaient de remplir de nouvelles tâches, donc d'intéresser de nouveaux utilisateurs. Mais ces logiciels n'ont pu être inventés, installés sur les machines que grâce à l'augmentation de la puissance des machines et surtout à la baisse du prix de cette puissance[16].

[15]Le prix d'un ordinateur dans les années 50 atteignait plusieurs millions de dollars ; aujourd'hui, un ordinateur de bureau, incomparablement plus puissant, vaut aux alentours de 1000 dollars (en dollars courants).

[16]Un exemple, pour illustrer cette affirmation : l'interface graphique et la souris on été inventés dès les

Le logiciel, ou plutôt l'immatériel n'a pourtant pas été exempt de progrès, principalement sur la façon dont les logiciels commandent la machine et lui font réaliser les calculs dont ils ont besoin pour rendre les services pour lesquels ils ont été créés. Cette évolution explique, indirectement, l'évolution de la production des logiciels.

Les progrès de l'immatériel font évoluer le lien entre le logiciel et la machine...

Les ordinateurs, nous l'avons écrit, partagent tous les mêmes principes d'«architecture» : la machine possède une mémoire, qui stocke les informations et les programmes, une unité logique, qui traite ces informations, une unité de contrôle, qui organise le fonctionnement interne de la machine et des organes d'entrée et de sortie (clavier, écran, imprimante, ...)[17]. Mais, bien sûr, la façon dont est effectivement réalisée cette architecture varie suivant les machines et est même une des compétences propres, un des «talents» des fabricants de machines. Il semble naturel de penser que les logiciels, qui utilisent ces supports, doivent s'y adapter pour pouvoir fonctionner. La principale innovation dans l'immatériel informatique a justement été de rendre progressivement ces programmes indépendants de la machine sur laquelle ils sont exécutés.

On a d'abord construit, dans les années 50, des traducteurs qui permettaient de séparer la façon dont les instructions sont codées pour la machine de la façon dont elles sont exprimées par le programmeur. C'est l'invention du compilateur, qui a permis de traduire le langage de programmation utilisé (et compréhensible) par l'humain dans le langage (binaire) utilisé par la machine et qui lui est spécifique[18]. À partir de là, on a pu réaliser un programme pour une machine et le réutiliser sur une autre machine, dès lors qu'il existait un compilateur adapté à cette nouvelle machine.

La deuxième étape a été celle de l'invention du système d'exploitation, c'est-à-dire d'un organe logiciel de contrôle de la machine. Cette réalisation a permis de rendre

années soixante (dans les laboratoires de Rank Xerox). Pourtant, ce n'est que vingt ans plus tard qu'Apple a vendu une machine qui utilisait ces inventions.

[17] Il existe des recherches autour d'autres architectures, vectorielles par exemple, mais elles ne se sont pas encore traduites par des réalisations utilisables industriellement.

[18] Les deux langages compilés les plus connus qui ont été inventés à cette époque sont sans doute le Fortran pour les calculs scientifiques et le Cobol pour les calculs de gestion ; ils sont toujours parmi les langages les plus utilisés dans le monde. Mais le LISP, le C, Pascal, Ada utilisent aussi un compilateurs.

complètement indépendants les programmes de l'architecture, de l'organisation matérielle des composants de l'ordinateur. Les programmes s'adressaient au système d'exploitation qui réalisait pour eux les instructions et leur communiquait les résultats demandés. On a pu ainsi séparer l'architecture de la machine, c'est-à-dire la façon théorique dont sont assemblés les différents éléments de la machine, de la réalisation de cette architecture, ce que Dréan [1996] appelle la «structure»[19]. Cette structure, c'est-à-dire les composants matériels, a alors pu évoluer sans qu'on ait besoin de reprogrammer ou de recompiler les programmes. Cela crée une séparation forte entre le matériel et le logiciel. L'architecture et le système d'exploitation deviennent les deux éléments caractéristiques d'une machine, les deux éléments qui permettent de faire fonctionner ensemble les différents composants matériels de la machine (processeur, mémoire, périphériques) et de garantir qu'un programme informatique fonctionnera sur une machine.

Nous allons montrer que c'est la création de ces deux éléments qui a permis de faire évoluer l'organisation industrielle en informatique. Pour cela, nous allons nous appuyer sur une grille d'analyse proposée par Gérard-Varet & Zimmermann [1985], qui aide à comprendre en quoi les innovations d'architecture et de système d'exploitation transforment la manière de produire un ordinateur[20].

Gérard-Varet & Zimmermann [1985] considèrent que l'on peut envisager un bien industriel (ordinateur, voiture, avion) comme un assemblage de composants, appelés «technologies élémentaires» (T) ; ce sont les processeurs, les mémoires, les écrans, etc., dans le cas des ordinateurs. Cet assemblage permet de construire des «plates-formes», ou encore des «produits» (P), selon leur terminologie. À ce niveau, il faut considérer le produit comme une machine universelle, dont les performances techniques sont évaluables, qui ouvre un espace d'utilisations possibles, sans en réaliser aucune : c'est l'ordinateur sans logiciel, c'est l'A380 avant que l'on décide du nombre de sièges que l'on y mettra. Avant, finale-

[19] «L'architecture est constituée par l'ensemble des règles et des conventions d'utilisation, dont par exemple la représentation des informations, le mode d'adressage, le jeu d'instructions, alors que la structure de chaque machine concerne les caractéristiques d'une réalisation particulière de l'architecture» (Dréan [1996]).

[20] Cette grille, qui a été construite en étudiant le cas de l'industrie informatique, peut s'appliquer à d'autres secteurs industriels, comme l'automobile, par exemple. Zimmermann [1995b] a formalisé cette théorie et nous nous appuierons indifféremment sur ces deux articles pour exposer la grille d'analyse.

ment (et c'est la troisième étape de la construction du bien industriel), qu'on ait décidé à quoi servirait cette machine universelle, à quelles «caractéristiques d'utilisation» (C) elle répondrait.

Le passage d'un espace à un autre représente un acte technologique, souvent industriel. Il s'agit de connaître les composants et les critères de l'espace «support» et de savoir mettre ces critères en correspondance avec l'offre que l'on veut proposer dans l'espace d'arrivée. La difficulté de l'exercice, qui fait aussi sa performance, est qu'il n'y a pas de relations simples (injectives ou surjectives) entre l'espace de départ et celui d'arrivée : il peut y avoir plusieurs technologies disponibles qui permettent d'atteindre le même niveau de performance et plusieurs offres de produits qui peuvent répondre à un même besoin.

Nous nommerons (en suivant Zimmermann [1995]), la correspondance entre l'espace des technologies et l'espace des produits la «*correspondance de production des performances*» . Elle met en jeu des compétences d'assemblage de technologies qui, pour cela, seront désignées sous le terme de «*technologies d'architecturation*». C'est, pour les agents économiques, l'acte de sélection entre toutes les correspondances de production des performances atteignables. La correspondance entre l'espace des produits et l'espace des caractéristiques d'utilisation, nommée «*correspondance de production des caractéristiques d'utilisation*», met en jeu la capacité de combiner des performances pour créer une offre de solutions à un problème posé par/à l'utilisateur. Là encore, il s'agit de compétences «technologiques» d'assemblage que nous désignerons sous le terme de «*technologies d'utilisation*».

Le progrès de l'immatériel, en informatique, a permis de penser l'espace des caractéristiques d'utilisation indépendamment de l'espace des technologies élémentaires, en construisant des machines effectivement universelles. En effet, tant que, pour réaliser un programme (c'est-à-dire répondre à une caractéristique d'utilisation), il fallait connaître la façon dont fonctionnaient les composants, individuellement et ensemble, cela voulait dire qu'un producteur de technologies d'utilisations devait aussi être compétent en technologies d'architecturation. C'est bien l'invention du compilateur et surtout du système d'exploitation qui ont permis de séparer ces compétences et qui de faire évoluer l'organisation de l'industrie informatique.

... et rendent possible l'évolution de l'organisation de l'industrie informatique.

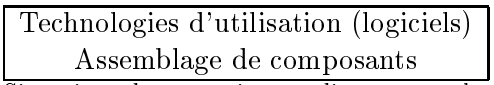
Le compilateur a initié l'idée qu'on pouvait construire des logiciels indépendants des machines. Cela a d'ailleurs eu pour effet d'augmenter les échanges de logiciels, qui devenaient possibles entre utilisateurs de machines différentes. L'invention du système d'exploitation va permettre à IBM de proposer, à partir du milieu des années 1960, non plus une offre d'ordinateurs, mais une gamme de machines, appelée 360. Même si les différentes machines de la gamme sont structurellement (physiquement) construites différemment, toutes reconnaissent le même jeu d'instructions, qui est public, gèrent les périphériques comme les imprimantes de la même façon (gestion des entrée-sorties) et peuvent exécuter les mêmes programmes.

Il devient alors possible d'envisager la construction de produits logiciels indépendants de la machine utilisée (à condition de rester dans la même gamme) et surtout de les faire évoluer, grandir, en fonction des besoins, puisqu'on trouvera toujours un ordinateur plus puissant et compatible pour les faire fonctionner. On peut aussi, et c'est en cela que réside la plus grande évolution, confier la production des technologies d'utilisation à d'autres entreprises que celles qui produisent les technologies d'architecturation. C'est la première possibilité de désintégration verticale de l'industrie. Elle n'aurait pas pu avoir lieu sans le progrès des composants (qui apportent plus de rapidité et qui permet de supporter la perte d'efficacité en vitesse de calcul que suppose l'utilisation d'un système d'exploitation), mais c'est bien le progrès dans l'immatériel qui l'a rendue possible.

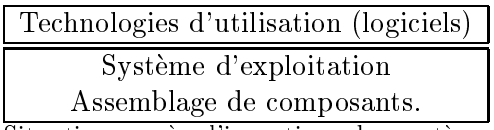
Suite à cela, suite aux procès qui ont cours à l'époque et suite aussi à l'augmentation des coûts de développement, IBM facture, à partir de 1969, ses services de développement de logiciels (autres que le système d'exploitation, qui reste vendu en même temps que la machine), d'assistance technique et de formation. C'est l'«unbundling» et c'est pour Dréan ([1996], p. 38) ce qui a permis la création d'un secteur du service informatique, dynamique et relativement concurrentiel[21]. Grâce à l'utilisation d'innovations technologiques, des entreprises, comme Digital, HP, Cray ou Apple, apparaissent, en se

[21] À cette époque, IBM était aussi sous la menace des premiers procès anti-trust (le premier date de 1968) ; cependant, d'après Dréan, l'augmentation très rapide de coûts de production des logiciels et des services est la principale raison de cette séparation comptable puis commerciale entre la vente des machines et celle des services.

Figure 1.1 — Une première désintégration verticale grâce à l'invention du système d'exploitation : la séparation entre producteurs de technologies d'utilisation et technologies d'architecturation.



Situation des premiers ordinateurs : les logiciels d'utilisation doivent eux-même s'occuper de la façon dont fonctionnent les composants.



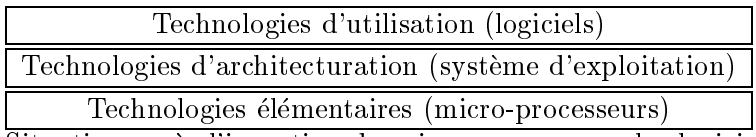
Situation après l'invention du système d'exploitation : les logiciels d'utilisation demandent des services de calcul au système d'exploitation (les services sont standards et n'évoluent pas) et le système d'exploitation se charge de gérer l'assemblage de composants et les évolutions dans la façon dont ils sont assemblés, s'il y en a.

différenciant verticalement par rapport au leader (en créant des machines plus petites, moins puissantes, mais moins chères). Elles obtiennent un certain succès puisque Digital devient, en 1982, la deuxième entreprise informatique derrière IBM et Apple le leader du marché des micro-ordinateurs en 1983. Mais elles reprennent toutes le même système de production et de commercialisation, et ce, jusqu'à l'arrivée du PC.

Une deuxième possibilité de désintégration est en effet ouverte, au début des années 80, grâce à l'arrivée du micro-processeur. Ce nouveau produit de l'évolution technologique du matériel permet de regrouper en une seule puce l'ensemble des éléments qui font les calculs dans un ordinateur. Il devient alors plus facile de séparer la production de ce composant de la production du système d'exploitation car les instructions que peut envoyer le système d'exploitation sont, à leur tour, standardisées. Le constructeur du processeur peut organiser les calculs comme il le veut «à l'intérieur de sa puce», à condition que les instructions que le processeur reçoit et les réponses qu'il donne ne changent pas.

Il y a à nouveau désintégration verticale avec la séparation entre les producteurs de technologies d'architecturation et les producteurs de technologies élémentaires. L'augmentation de rapidité des composants a été, une fois de plus utilisée pour rajouter une «couche», un nouveau système de traduction automatique, entre la façon dont les composants fonctionnent réellement (et les instructions qu'ils comprennent) et la façon dont

Figure 1.2 — Une deuxième désintégration verticale : la séparation entre producteurs de technologies élémentaires et de technologies d'architecturation.



Situation après l'invention du micro-processeur : les logiciels d'utilisation demandent des services de calcul au système d'exploitation (les services sont standards et n'évoluent pas) et le système d'exploitation demande au micro-processeur d'effectuer une série de calcul (grâce à un jeu d'instruction réduit et standardisé). Ce dernier organise lui-même la façon dont il répond aux instructions et dont il mêne les calculs.

les logiciels demandent ces calculs.

C'est une fois de plus IBM, qui, avec le PC, va proposer une machine où la production de ces deux composants n'est pas réalisée par la même entreprise, avec le succès que l'on sait.

Mais il ne faudrait pas croire, pour autant, que cette évolution technologique suffise à expliquer l'évolution de l'organisation industrielle et notamment l'évolution de la manière dont sont produits les logiciels. Après tout, ce sont ceux qui produisaient les logiciels coopérativement qui ont inventé le système d'exploitation ou le micro-processeur. Nous avons même dit qu'a priori, le fait de séparer la production des logiciels des machines rendait plus facile la réutilisation de ces logiciels. Cela aurait dû favoriser l'échange et la coopération. D'autre part, comme le note Genthon [1995], le micro-processeur n'a pas entraîné partout la séparation entre producteurs des technologies d'architecturation et producteurs de technologies élémentaires : cette séparation n'a pas eu lieu au Japon et Apple ou SUN dessinent toujours leurs micro-processeurs.

Pourtant, il faut reconnaître qu'avec l'arrivée de la série 360 s'est développée la production commerciale de logiciels d'utilisation et qu'avec l'arrivée du PC, celle des progiciels, c'est-à-dire de logiciels standards. Nous expliquons cela par le fait que le progrès technologique a fait évoluer la demande et que les offres désintégrées verticalement répondaient mieux à l'évolution de cette demande.

1.2.2 Parce que les demandes ont évolué, certaines organisations de production ont été plébiscitées.

Le fait que l'objet informatique soit un assemblage complexe est source d'inégalité entre les utilisateurs : certains sont capables de comprendre comment il est architecturé quand d'autres n'ont comme connaissance de son fonctionnement que la connaissance de certains logiciels d'utilisation. Nous allons voir que cette compétence a un impact sur la façon dont les utilisateurs envisagent les offres des producteurs et donc sur la façon dont les logiciels doivent être produits pour qu'ils puissent les consommer.

Des utilisateurs aux compétences différentes qui évaluent les offres suivant des critères différents.

On peut distinguer 3 niveaux de compétences chez les utilisateurs, toujours suivant une typologie proposée par Gérard-Varet & Zimmermann ([1985], p. 16 à 19) :

- l'utilisateur naïf, qui ne possède aucune connaissance technique sur le produit.
  Il ne peut donc utiliser que le prix comme information pour faire son choix, information limitée car cela implique qu'il ne puisse comparer que les offres identiques en terme de caractéristiques. Il ne peut pas influencer directement l'offre et il doit payer un service d'aide et d'intervention (recherche d'informations et de mise à disposition de compétences techniques), sans pouvoir évaluer vraiment son efficacité, ou acheter une offre standard de solutions. C'est, par exemple, l'acheteur des offres d'ordinateurs «grand public». Historiquement, c'est le dernier utilisateur qui a eu accès à l'outil informatique grâce à l'arrivée des micro-ordinateurs ;
- l'utilisateur sophistiqué, capable d'exprimer ses besoins en terme de technologies d'utilisation. Se pose, pour lui, le problème de l'efficacité de la réalisation effective de cette demande. Il possède des connaissances suffisantes pour évaluer la qualité des services et des biens d'utilisation et il fait un arbitrage entre son implication personnelle et l'implication d'un tiers. S'il utilise un service d'assistance, il participe à la production de ce service, en définissant ses besoins et en évaluant les propositions qui lui sont faites. C'est, par exemple, l'acheteur de logiciels professionnels spécialisés, capable de comparer les offres de logiciels et les services de développement

et d'adaptation proposés. Ce type d'utilisateur s'est intéressé à l'informatique à partir du moment où cet outil ne nécessitait plus de compétences techniques poussées pour être utilisé, donc à partir du moment où sont apparus les langages de programmation, indépendants de la machine, mais aussi à partir du moment où il a été possible d'utiliser des ordinateurs sans avoir besoin des compétences des ingénieurs informaticiens. Les premiers utilisateurs «sophistiqués» sont donc apparus au moment où l'on a pu utiliser un ordinateur sans se préoccuper du fonctionnement de ses composants, c'est-à-dire à partir du moment où a été inventé le système d'exploitation ;

– l'utilisateur «designer», capable de travailler directement sur les technologies d'architecture et sur les technologies d'utilisation. Son arbitrage entre faire, faire-faire ou acheter s'étend au choix des technologies d'architecturation. Il peut demander des spécificités architecturales ou des modifications de l'architecture des produits qui lui sont vendus. C'est, par exemple, l'ingénieur réseau qui va demander des spécifications matérielles précises, qui va co-construire la partie physique de son réseau informatique et qui va également participer au choix (et à l'adaptation) des logiciels qui vont rendre les «services» demandés. Ce type d'utilisateur est présent depuis les origines de l'informatique. Il a même été, à une époque, le seul capable de faire fonctionner les machines. Mais, plus généralement, c'est le premier adopteur des nouvelles machines (les mini-ordinateurs ont d'abord été utilisés dans les laboratoires d'informatique, dans les universités et les centres de recherche et les premiers micro-ordinateurs étaient destinés à des hobbyistes capables de les monter)[22].

Levons d'abord une ambiguïté : si, grâce à l'apparition de nouvelles machines, des utilisateurs moins «compétents» en informatique ont pu utiliser cet outil, cela ne veut pas dire pour autant qu'à un type de machine correspond un type d'utilisateur. Des utilisateurs «naïfs» peuvent utiliser des mainframes en tant que clients, pour rentrer des données par exemple, une fois que la machine et ses programmes sont installés et il est certain que la

[22]Continuons notre parallèle avec l'automobile, industrie sans doute mieux connue. Au début de l'automobile, les constructeurs étaient avant tout des mécaniciens, capables d'assembler des composants technologiques divers pour construire des prototypes d'automobiles. Notons que le rôle social de ces utilisateurs avancés dans l'adoption de nouvelles technologies a été analysé par von Hippel [1988] mais aussi par Abernathy & Utterback [1987]. Nous y reviendrons dans le chapitre suivant.

plupart des utilisateurs sophistiqués et designers possèdent un PC ou un Machintosh. Mais, on le voit, ces utilisateurs ne vont pas aussi loin dans l'analyse du bien qu'ils achètent.

Or, à cause des phénomènes de standardisation, le choix des utilisateurs a une influence sur l'offre disponible, mais aussi sur la demande des autres utilisateurs. Par exemple, plus il y a de personnes qui choisissent un système d'exploitation, plus l'offre de logiciels pour ce système se développe et plus il devient intéressant de le choisir. L'ordre dans lequel les adopteurs choisissent les technologies est donc important et a une influence sur l'offre finalement disponible. Nous allons montrer que c'est la baisse des connaissances moyennes des utilisateurs qui a entraîné l'évolution de la demande.

La baisse de la connaissance moyenne des utilisateurs d'ordinateurs entraîne une standardisation et une marchandisation du logiciel.

Reprenons l'histoire de l'informatique, en y incluant cette fois les utilisateurs et leur niveau de connaissance de l'objet ordinateur.

Les débuts de l'informatique : recherche et co-production.

Aux débuts de l'informatique, les clients étaient souvent des militaires, qui passaient des contrats proches de contrats de recherche. Les producteurs et les utilisateurs inventaient en même temps les services et l'outil qui permettrait de réaliser ces services. Les utilisateurs étaient tous des utilisateurs «designers».

L'évaluation des offres se faisait au niveau du matériel, des composants technologiques ; la relation client-fournisseur était une relation de service, de co-définition des besoins et de co-construction de la réponse. De plus, parce que c'était une industrie stratégique, soutenue par les politiques industrielles nationales, une collaboration rapprochée s'était mise en place entre des clients (souvent publics) et des fournisseurs pour favoriser la recherche dans ce domaine. Le progrès technologique était alors le principal critère d'évaluation des offres, ce qui se comprend aisément dans un contexte industriel proche de la recherche scientifique. Recherche scientifique, culture du prototype, volonté de la puissance publique de développer les échanges pour accélérer l'innovation et surtout utilisateurs capables de produire eux-mêmes les logiciels dont ils ont besoin ; toutes

ces raisons font que la production de logiciel était «naturellement» une production coopérative.

Mais la diffusion de ces machines dans les entreprises a fait évoluer les contraintes ; l'évaluation des offres a alors intégré des critères économiques et ce sont ces critères qui ont fait le succès d'IBM avec la série 360.

L'arrivée du système d'exploitation et des entreprises entraîne une marchandisation de la production de logiciel.

Les entreprises clientes plébiscitèrent cette offre qui rendait possible l'amortissement de leurs investissements dans les logiciels sur une plus grande période et le développement de ces logiciels à chaque fois que de nouvelles demandes apparaissaient, sans être contraintes par la puissance de la machine. Autrement dit, la stabilité du système d'exploitation permettait de capitaliser les apprentissages faits sur une machine car, même si elle n'est plus assez puissante pour répondre à l'évolution des besoins, celle qui la suivait l'était et elle fonctionnait de la même façon.

Ces clients étaient des clients designers ou sophistiqués, qui savaient produire des logiciels. Se posait pour eux la question de faire ou de faire-faire les logiciels dont ils avaient besoin. C'est à ce moment qu'est apparu le besoin d'avoir une évaluation du coût de développement d'un logiciel. La possibilité d'accumuler des connaissances sur les logiciels, de développer sans cesse de nouvelles fonctionnalités rendait aussi ces logiciels de plus en plus importants pour les entreprises, mais aussi de plus en plus spécifiques. Enfin, comme ces logiciels devenaient aussi de plus en plus gros, de plus en plus complexes, ils requéraient de plus en plus de connaissances de la part des développeurs. Toutes les entreprises n'avaient pas les moyens d'entretenir des armées de développeurs et les années 1970 (et surtout le début des années 1980) furent les années où les grandes entreprises se sont séparées d'une grande partie de leur services informatiques, qui se transformèrent alors en entreprises de service.

Et tous ces éléments : révélation et augmentation des coûts de développement, importance stratégique et spécificité des logiciels développés, complexité, sont des éléments qui

sont allés à l'encontre de la coopération. Et ce d'autant plus que les informaticiens, de plus en plus nombreux, n'étaient plus tous issus des laboratoires de recherche universitaires en informatique. Ils n'avaient donc pas acquis la culture de la coopération qui restait vive dans ces laboratoires. Naturellement, ce fut aussi l'époque où les producteurs de logiciels (menés par IBM) demandèrent à voir ces compétences reconnues et rémunérées, donc à ce qu'on considérât le logiciel comme un bien auquel il fallait accorder des droits de propriété. Cette revendication a abouti dans les années 1970 et a créé une barrière supplémentaire à l'échange puisque les utilisateurs d'un logiciel protégé n'avaient légalement plus le droit de l'échanger.

Parce que ces utilisateurs étaient les plus nombreux, parce que ce sont eux qui créaient le plus d'activité économique, cette façon de produire du logiciel s'est imposée et la production coopérative a été reléguée (ou plutôt est restée confinée) aux endroits où elle avait toujours existé : dans les laboratoires de recherche.

L'arrivée du PC entraîne une marchandisation du logiciel.

La troisième période, celle de la micro-informatique, a débuté au milieu des années 1970, avec l'idée de construire des machines individuelles. On peut dire que cette demande est venue en partie des utilisateurs des machines informatiques qui, pour pouvoir réaliser leurs calculs de gestion ou leurs calculs scientifiques, dépendaient du bon vouloir de la «direction informatique», alors toute puissante dans les organisations. Apple, nouveau venu dans ce secteur, se faisait alors le chantre de la liberté informatique, contre cette direction et contre la toute puissance d'IBM. C'est pourtant IBM, nous l'avons dit, qui va permettre de restructurer l'industrie en abandonnant le contrôle sur le système d'exploitation et sur l'architecture de sa machine[23]. Cela a complètement changé l'organisation industrielle,

[23]On s'est beaucoup interrogé sur les raisons qui ont poussé cette entreprise à agir de la sorte. Il est toujours facile de prévoir ex-post les conséquences d'un choix stratégique. Pourtant, il nous semble que cela peut s'expliquer. D'abord en disant, comme le fait remarquer Genthon [1995] qu'IBM ne pensait pas que le marché de la micro-informatique s'adressait aux entreprises, mais plutôt aux utilisateurs designers (et c'est vrai que ce fut le premier marché). Si elle a choisi de sous-traiter la production des composants, c'est sans doute principalement à cause de cette «mauvaise» analyse du marché et de son retard sur le marché. L'ouverture de l'architecture permet à des concurrents de produire des clones de la machine. Cela multiplie les offres et permet de couvrir plus rapidement le marché. Ensuite, sa taille, sa notoriété devait sans doute lui permettre, dans un deuxième temps, de limiter la part de marché de ses concurrents. Le recours à des sous-traitants s'explique de la même façon. Les tailles de Intel ou de Tandon (fabricant de disques dur), pour ne pas parler de celle de Microsoft, font qu'à cette époque ces sous-traitants sont dépendants d'IBM.

comme on peut le voir dans le tableau 1.3.

Tableau 1.3 — Organisation verticale de l'industrie informatique.

	Nouvelle informatique (PC)	**Ancienne informatique (mainframes)**
distribution	externalisée	internalisée
applications	externalisées	externalisées
systèmes d'exploitation	externalisés	internalisés
systèmes de réseau	externalisés	internalisés
périphériques	externalisés	internalisés/externalisés
réseau	externalisé	internalisé
plate-forme matérielle	internalisée	internalisée
processeur	externalisé	conçu en interne

source : Genthon [1995], p. 104.

Grâce à la notoriété de l'entreprise IBM, le PC est devenu une offre crédible, et ce d'autant plus qu'elle ne semblait pas remettre en cause le pouvoir de la direction informatique (elle s'adressait toujours à la même entreprise), tout en satisfaisant les velléités d'indépendance des utilisateurs. En même temps, l'offre des concurrents augmentait la pression sur les prix et permettait de faire baisser le coût de ces systèmes.

Les mécanismes du succès de cette machine sont similaires à ceux qui ont expliqué le succès de la série 360 d'IBM. Comme pour la série 360, chacun des deux éléments clefs de la machine, le microprocesseur et surtout le système d'exploitation, n'a rapidement été produit que par une entreprise, en situation de quasi monopole. Comme pour la série 360, cette double standardisation était nécessaire pour pouvoir garantir la pérennité des investissements. La spécificité de cette diffusion résidait dans le fait que plusieurs entreprises pouvaient produire les mêmes machines. Cela a augmenté la concurrence et ce d'autant plus que la standardisation sur le système d'exploitation et sur le micro-processeur permettaient de comparer facilement les offres : les machines fonctionnaient toutes de la même façon, on pouvait les comparer facilement selon quelques critères techniques et surtout selon leur prix, dispositif d'information utilisable par tous. Les utilisateurs ne maîtrisant pas les technologies d'utilisation, c'est-à-dire les utilisateurs naïfs, en sont rapidement devenus les principaux utilisateurs (même si les acheteurs restaient les utilisateurs sophistiqués, qui avaient la responsabilité des choix informatiques dans les entreprises).

Même si les PC étaient sans doute inférieurs techniquement au Machintoshs (Apple

propose une interface graphique en 1984, Microsoft, au début des années 90), cette concurrence plus forte a permis une évolution plus rapide du rapport prix/performances[24] et les a rendu de plus en plus attractifs (voir Genthon [1995]). On peut résumer l'efficacité de cette organisation en disant qu'IBM a apporté à ce produit sa marque et les fabricants de clones, qui n'avaient pas à supporter les mêmes coûts de structure mais pouvaient utiliser les mêmes composants que le leader, la compétitivité en prix.

Comme l'arrivée des utilisateurs sophistiqués (qui ne maîtrisaient pas les technologies d'architecturation) avait favorisé la standardisation de l'offre des machines (c'est-à-dire des produits construits grâce à ces technologies d'architecturation), l'arrivée des utilisateurs naïfs a favorisé la standardisation de l'offre de technologies d'architecturation et, par voie de conséquence, la marchandisation et la production propriétaire de logiciels. D'abord parce que le système d'exploitation, le seul logiciel qui n'était pas vendu, est devenu un composant identique aux autres. D'autre part parce que l'utilisation principale de ces machines était le traitement d'informations personnelles (tableurs, traitements de texte essentiellement), tâches qui ne nécessitaient aucune connaissance en informatique. Donc des utilisateurs naïfs se retrouvèrent à utiliser des ordinateurs, seuls, pour réaliser ces tâches. Pour eux, l'objet informatique était une boîte noire, ils n'étaient pas capables d'exprimer des besoins en terme de technologies d'utilisation. Leurs seuls critères d'évaluation étaient le prix, éventuellement la marque du producteur. S'est alors développée une offre de progiciels «sur étagère» (Mowery [1996]), c'est-à-dire des logiciels standardisés, construits et vendus pour réaliser une tâche précise et intégrés dans la boîte noire.

Ces offres se sont rapidement standardisées autour de quelques produits par fonction à cause des caractéristiques particulières du logiciel : il est sans coût de reproduction, donc plus son utilisation est partagée par un grand nombre de personne, plus la part du coût de développement supportée par chaque utilisateur est faible[25].

[24] On peut cependant se demander pourquoi, au départ, l'offre d'Apple n'a pas été plus choisie par les utilisateurs. On peut constater qu'Apple s'est imposé dans une niche de marché, celle des professionnels de l'édition, qui étaient capables d'identifier et de valoriser la meilleure qualité technique des Macintoshs au niveau du traitement des objets graphiques, et chez les professions indépendantes, qui choisissaient elles-mêmes leurs machines professionnelles. Sans doute parce qu'elle n'était pas aussi connue qu'IBM, peut-être parce qu'elle se posait même comme un anti-IBM, cette entreprise a eu plus de mal à convaincre les acheteurs sophistiqués des entreprises.

[25] Pour la simplicité et la cohérence du propos, nous n'irons pas plus loin dans le détail des mécanismes qui expliquent cette standardisation et l'avantage de cette organisation industrielle, préférant nous concentrer sur l'évolution de la production de logiciel. Nous les avons détaillés dans Jullien [1999], en nous appuyant

Tableau 1.4 — Marché européen du logiciel

	1994	**1999**	**Taux de croissance annuel moyen 1994-1999**
Logiciels applicatifs	21,8	33,8	9,2 %
Logiciels systèmes et outils	12,2	14,5	3,5 %
Total	34,0	48,3	7,3 %

source : Horn [2000c], p. 363, d'après Eurostaf [1999], p. 93.

On peut donc dire, en guise de conclusion partielle, que, grâce au progrès technologique, des utilisateurs de plus en plus nombreux ont eu accès à l'objet ordinateur ; que ces utilisateurs, étant de moins en moins compétents en informatique ont eu besoin de tiers pour développer les logiciels qui répondaient à leurs besoins en terme de technologies d'utilisation ; que l'arrivée de nouveaux utilisateurs a sélectionné les organisations industrielles qui rendaient l'achat de l'objet informatique le plus simple possible pour eux. La simplicité a été synonyme de développement et de vente de logiciels standards par des producteurs principalement parce que ces utilisateurs n'avaient pas les compétences pour les développer eux-mêmes. La conséquence de cette marchandisation progressive a été la marginalisation de la production coopérative.

Après cette analyse, le Libre semble de plus en plus improbable. Pourtant, elle a aussi montré qu'avec l'évolution de la demande, de nouvelles organisations de production pouvaient apparaître. D'autre part, nous avons dit que la production coopérative avait toujours existé et qu'elle avait continuellement apporté de nouvelles technologies et de nouveaux usages aux producteurs et aux utilisateurs de l'outil informatique. Enfin, ce n'est pas parce que les PC sont apparus que les utilisateurs sophistiqués et les utilisateurs designers ont disparu. L'organisation industrielle du PC n'a pas non plus fait disparaître les autres produits, comme on peut le constater dans le tableau ci-dessous.

Actuellement, il y a encore une progression en valeur des mainframes et des serveurs Unix et une relative stabilité de la part de marché de ces mainframes. Enfin, les enquêtes montrent un attachement de la part des utilisateurs à ces machines[26].

sur les ouvrages de référence cités au début de cette partie.

[26] Voir, notamment, Charbit & Zimmermann [1998] et de Saloner & Steinmueller [1996].

Tableau 1.5 — Évolution des parts de marchés des sous-systèmes informatiques (en pourcentage et en valeur, pour les unités centrales.)

		1976	**1980**	**1985**	**1989**	**1992**	**1999**[a]
«Ancienne informatique»	**Mainframe**	70	65	44	30	26	14s[b]
	Mini informatique	30	31	29	24	20	
	Machines sous Unix				7	13	17
«Nouvelle informatique»	**Micro**		4	27	39	41	69
	Total	100	100	100	100	100	100

Sources : Genthon [2000] d'après Bipe pour 1976 et 1980 ; Datamation, 15 juin 1986, 1990, 1993 et EITO [1999] pour 1999.

[a] Chiffres de l'Europe occidentale.
[b] Comprend les mainframes et les minis ne fonctionnant pas sous Unix.

L'arrivée d'Internet fait évoluer la demande et rapproche ces différentes organisations en reliant les produits. Elle a aussi permis une certaine diffusion des logiciels libres. Si ceux-ci répondent mieux à l'évolution actuelle de la demande (et surtout à la demande des entreprises), elle va peut-être s'accélérer. S'ils sont mieux adaptés parce qu'ils sont produits différemment, leur organisation de production, le Libre, peut faire évoluer la façon dont sont produits les logiciels. Il faut donc maintenant étudier l'évolution de la demande et se demander si le Libre est adapté à celle-ci.

1.3 Le Libre est-il adapté à la situation actuelle ?

Répondre à la question «le libre est-il adapté à la situation actuelle» nécessite bien sûr de commencer par décrire cette situation. Nous avons laissé l'évolution de l'informatique aux alentours des années 1990, avant l'arrivée d'Internet et des réseaux d'entreprise. Il va falloir étudier ce que l'arrivée de ces réseaux a changé. Il faudra aussi s'intéresser aux progrès conceptuels sur les logiciels, puisque ce sont eux qui permettent l'évolution de l'industrie. À partir de là, nous montrerons en quoi ils ont fait évoluer la demande de logiciels et plus généralement de technologies d'utilisation. Enfin, nous verrons quelles réponses nouvelles le Libre apporte à ces nouveaux besoins.

1.3.1 Le progrès technologique : composants et mise en réseau.

Le progrès technologique majeur des années 1990 est bien sûr la mise en réseau des ordinateurs, à l'intérieur et à l'extérieur de l'entreprise. Du côté des logiciels, les progrès sur les langages permettent plus facilement de réutiliser des morceaux de logiciels, de produire des composants logiciels, comme il existe des composants matériels.

Ces deux évolutions s'inscrivent dans la continuité des évolutions précédentes de l'industrie informatique et on peut se demander si, elles-aussi, elles entraîneront une réorganisation de la façon dont sont produits les logiciels.

Le progrès sur le matériel : réseau et mobilité.

Au cours des années 90, la principale évolution technique en informatique a été la mise en réseau des ordinateurs, grâce à la diffusion industrielle de technologies développées soit dans les universités, soit par des industriels, producteurs d'informatique (IBM) ou utilisateurs (General Motors). La miniaturisation a, une fois de plus, permis l'apparition d'un nouvelle gamme de produits, «nomades» («organisers» (Psion et Palm), téléphones mobiles).

La mise en réseau et la spécialisation des machines s'inscrivent dans la continuité de l'évolution des produits informatiques : on est passé de la machine unique, dédiée à une tâche connue par avance et dévolue à l'ensemble d'une organisation, à des machines multiples, utilisées pour réaliser des tâches variées et variables dans le temps, intégrées dans une organisation et communicantes. L'arrivée d'Internet n'a fait que renforcer ce phénomène en permettant aux machines de communiquer en dehors du cadre de l'organisation, avec (au moins théoriquement) n'importe quelle machine dans le monde. Tout cela a encore augmenté le ratio $\frac{nombre\,de\,machines}{nombre\,d'utilisateurs}$, soit le nombre de machines utilisées par un individu. Sans compter celles auxquelles l'individu peut accéder par l'intermédiaire d'un réseau, le nombre de machines qu'il possède en propre et entre lesquelles il échange de l'information a aussi dépassé l'unité : ordinateur et/ou téléphone portables, PDA ont rejoint l'ordinateur «fixe» traditionnel.

Mise en réseau, échange entre systèmes hétérogènes, le problème de la communication entre ces machines devient crucial, d'autant plus qu'elles se sont multipliées. Pour le résoudre, il faut mettre au point des standards au niveau du réseau d'échange, mais aussi au niveau des données échangées, afin qu'elles soient transmises et qu'elles soient déchiffrables par tous ces systèmes. Ainsi, le progrès d'Internet n'est pas d'avoir proposé un «protocole» pour permettre la transmission simple des données, il en existait déjà, mais d'en avoir proposé un suffisamment simple, suffisamment souple, pour s'être imposé comme un standard d'échange. Ce besoin de souplesse, d'adaptabilité, a fait aussi évoluer la façon dont sont produits les logiciels.

Le développement des composants : flexibilité et adaptabilité.

Horn [2000] (pp. 126-128) s'est intéressé à l'évolution des techniques de programmation. Nous relèverons deux points dans son analyse : d'abord que l'arrivée des langages de programmation objet (Smaltalk, C++, Java et peut-être bientôt C#, proposé par Microsoft) a facilité la réutilisation de composants logiciels déjà développés ; ensuite que cela a débouché sur le concept du «logiciel modulaire» dont l'idée est de développer un ensemble de petits logiciels (des modules ou des composants logiciels), qui rendraient chacun un service précis et qui seraient associables, utilisables sur n'importe quelle machine car leurs interfaces de communication seraient standards.

Zimmermann [1997] souligne aussi cette évolution de l'industrie vers la réutilisation de composants logiciels interchangeables. Cette réutilisation, dont on parle depuis déjà longtemps, est rendue aujourd'hui plus aisée par les derniers progrès du génie logiciel (Beugnard [2001]), qui permet de mieux spécifier le fonctionnement et les protocoles d'échanges entre composants.

Ce qui caractérise l'évolution technologique dans le logiciel est donc l'interdépendance croissante entre les logiciels, en même temps que la spécialisation de plus en plus fine des composants logiciels utilisés. Ce système ne peut fonctionner que si effectivement les composants sont réutilisables, c'est-à-dire si les producteurs s'accordent autour d'un mécanisme qui permette de standardiser les interfaces et de garantir, dans le temps, la

stabilité des standards (sinon, à chaque fois qu'un des composants est modifié, il faudrait modifier l'ensemble des composants avec lesquels il est en relation). Cette évolution n'est pas non plus sans conséquence sur les caractéristiques de la demande.

1.3.2 Le progrès technologique fait à nouveau évoluer la demande et l'offre informatique.

Nous allons d'abord voir que, parce que les utilisateurs sont de plus en plus confrontés à ces problèmes d'interopérabilité, de stabilité des standards, qu'ils ont aussi constaté les nuisances que peut engendrer l'utilisation de progiciels trop rigides, leur exigence de modularité (qui permet d'adapter plus facilement les logiciels) et d'interopérabilité devient aussi de plus en plus grande.

D'autre part, si le progrès technologique fait évoluer la demande vers plus d'adaptabilité et en même temps vers plus de standards dans les formats d'échange, cela aura une conséquence : il faudra que les entreprises adaptent leurs logiciels aux demandes, ce qui veut dire qu'en plus de vendre des logiciels, elles vendront des services d'accompagnement. Nous montrerons que cette évolution ne se fera pas sans poser des problèmes, principalement sur la qualité des services fournis et sur la construction des standards d'interopérabilité. Leur résolution passe sans doute par une évolution de la relation utilisateur-producteur.

L'évolution de la demande : interopérabilité, adaptabilité, stabilité des logiciels.

La mise en réseau des utilisateurs a accentué le besoin que des logiciels, souvent produits par des entreprises différentes, pour des utilisateurs différents, puissent échanger des données. Il faut aussi qu'elles s'assurent de sa disponibilité dans le temps, malgré les changements de version. Il devient donc nécessaire de standardiser les formats d'échange des fichiers. Ce besoin d'interopérabilité est encore accru quand les logiciels développés font appels à des composants, puisqu'il faut s'assurer qu'ils peuvent communiquer entre eux, mais aussi avec les autres programmes de l'entreprise. Il est alors logique que les entreprises recherchent des solutions plus ouvertes, qui leur garantissent de mieux disposer de ce contrôle. La vitesse de diffusion du protocole TCP/IP témoigne de cet intérêt.

Ensuite s'exprime le besoin d'une nécessaire adaptabilité des logiciels. En effet, l'hétérogénéité de la demande interne aux entreprises s'est aussi accrue avec la mise en réseau des différents systèmes et avec la nécessité que des utilisateurs de l'entreprise partagent les mêmes outils. Il faut alors adapter les logiciels (et particulièrement les progiciels) aux besoins et aux connaissances de chacun.

On peut résumer cette évolution en disant que la mise en réseau fait évoluer la demande (surtout de la part des entreprises), vers le développement de technologies d'utilisation qui permettent «une standardisation croissante des produits et une adéquation plus fine de l'offre à des spécifications plus étroites de la demande» (Zimmermann [1989], p. 1284).

Mais cette évolution est rendue difficile par l'insatisfaction des utilisateurs vis-à-vis de la qualité des logiciels produits par l'organisation industrielle actuelle.

Le fait que, essentiellement pour des raisons de coût, des utilisateurs sophistiqués, voir designers, utilisent de plus en plus de progiciels contribue sans doute à élever le niveau d'exigence : ceux-ci sont plus à même d'évaluer la qualité des logiciels et sont plus exigeants sur cette qualité. Mais les utilisateurs naïfs ont aussi pu se rendre compte, à l'utilisation, de l'inadéquation de certains logiciels avec leurs besoins : l'existence d'un nombre important de fonctionnalités inutiles rend, par exemple, le logiciel très gourmand en capacité de calcul, fait qu'il ne fonctionne pas ou mal sur les machines un peu anciennes et complexifie beaucoup son utilisation. Enfin, la fiabilité de ces progiciels peut être faible car les tests de fiabilité sont coûteux, longs et peuvent donc retarder la mise sur le marché. Et quand le logiciel est un standard de fait, les utilisateurs sont quasiment obligés de l'adopter, indépendamment de son adéquation à leurs besoins, ce qui n'incite pas son producteur à faire des efforts (coûteux) pour améliorer sa qualité (Horn [2000b], p. 350). Pourtant, là aussi, l'importance grandissante de l'information dans l'entreprise (et le temps consacré à sa manipulation)[27] rendent de plus en plus

[27] Pour Charbit & Zimmermann ([1997], p.2), «les technologies de l'information chez les utilisateurs se sont stratégiquement rapprochées du cœur de leurs compétences propres et les conséquences de l'introduction de nouveaux concepts de systèmes d'information ont, chez ces utilisateurs, des répercussions importantes quant à leur organisation, voire même quant au contenu de leurs activités».

Ce changement est suffisamment important pour qu'ils considèrent que l'arrivée des réseaux et notamment d'Internet a fait entrer l'informatique dans une nouvelle ère, l'«ère de l'information», qui succéderait

coûteux les pannes, les pertes d'information. Horn ([2000b], p. 317) fait remarquer que comme elles concernent, directement et indirectement, de plus en plus de monde parce que l'environnement économique dépend de plus en plus des systèmes informatisés et parce que les systèmes informatiques sont de plus en plus inter-connectés, ces erreurs sont de plus en plus difficiles à accepter.

Nous allons voir que la conséquence de ces évolutions techniques et de l'évolution de la demande est que les producteurs doivent construire des offres en y intégrant une part de plus en plus importante de service et en même temps travailler sur l'amélioration de la qualité des composants à la base de ces offres.

La conséquence : une nécessaire évolution de l'offre, qui doit être plus ouverte et intégrer plus de service.

On peut penser que la production par composants et le besoin d'adaptabilité et d'interopérabilité devraient conduire à une spécialisation des différents acteurs de la chaîne de production de logiciel, en même temps qu'à la baisse du coût de l'adaptation d'un logiciel à des besoins propres, à l'image de ce qui s'est passé dans la production des composants matériels, des technologies élémentaires[28].

L'évolution des chiffres d'affaires et des stratégies des entreprises de l'informatique semblent confirmer cette idée : la part des services dans le chiffre d'affaires d'IBM est passée de 32 à près de 37 % en deux ans[29] et c'est le seul poste qui progresse significativement. HP a tenté dernièrement de racheter PriceWaterhouseCoopers, spécialiste du conseil aux entreprises. Compaq, depuis rachetée par HP, avait annoncé en juin 2001 qu'elle se donnait 18 mois pour devenir une «*service company*». Même Microsoft, entreprise réputée pour ne pas vendre de services complémentaires à ces logiciels, a annoncé le projet «.net», qui a pour objet de ne plus vendre de logiciels, mais de proposer

à l'«ère des ordinateurs universels» et à «l'ère de l'informatique».

[28] Ainsi, Beugnard ([2000], p. 3) note que «la spécialisation des compétences ne se fait plus uniquement par technologies utilisées, ou par domaines applicatifs visés, mais par étape de développement du logiciel. De nouveaux métiers devraient apparaître : architecte, assembleur, déployeur, administrateur, «composanthécaire» (par analogie avec bibliothécaire).» Zimmermann [1997] écrit que «bien souvent ce seront les utilisateurs eux-mêmes qui intégreront les composants nécessaires à la réalisation de leurs applications».

[29] 1997 ->1999, voir http ://www.ibm.com/annualreport/1999/report/md_04.html.

des services d'utilisation d'outils logiciels adaptés aux besoins des utilisateurs. Les services représentent presque la moitié des dépenses en informatique, avec un taux de croissance bien supérieur à celui du matériel (et aussi à celui du logiciel, en Europe au moins, voir le tableau 2 de l'introduction).

Ces chiffres et le changement de positionnement des acteurs accréditent l'idée que nous serions entrés dans une nouvelle phase de la production de logiciel, que Horn ([2000b], p. 295-297) appelle le «sur-mesure de masse». Elle serait caractérisée par «la fourniture avec un progiciel d'une proportion croissante de services divers et variés ; le développement de logiciels sur mesure en réutilisant des modules déjà développés et testés» (Horn [2000b], p. 295).

Se posent alors deux problèmes : celui de la garantie que les modules testés remplissent bien les fonctions demandées, communiquent entre-eux de la façon indiquée par les producteurs, ce qui renvoie à la façon dont sont produits ces standards d'échange ; celui aussi de la qualité du service fourni par les producteurs de technologies d'utilisation chargés d'assembler les modules. Ce sont ceux que les producteurs et les utilisateurs de logiciels doivent résoudre pour que se développe effectivement la production du «sur-mesure de masse».

Le problème de la production des services.

Dans son ouvrage de 1995 (particulièrement dans le chapitre 4), De Bandt étudie le marché des services. Pour lui, «si, en matière de service, les mécanismes de marché ou de concurrence fonctionnent, ils le font selon des modalités particulières, plus imparfaites qu'ailleurs» (p. 97).

En effet, il n'est pas possible de consommer le produit avant de l'acheter, l'activité de service étant censée répondre à un besoin spécifique. On peut tout au plus avoir des informations sur les «inputs» de production, c'est-à-dire sur les compétences et les objets techniques qui vont être utilisés pour fournir le service.

Pour pouvoir évaluer les offres de service, il faut pouvoir comparer :

- «les niveaux de compétences de l'entreprise prestataire de service en fonction des investissements non-matériels (informations, démarches, procédures, ...) et des expériences et apprentissages accumulés ;
- les niveaux de compétences (capacités, formations, expériences) des personnes affectées à l'opération concernée. Les écarts peuvent être importants en ce qui concerne les définitions des niveaux et contenus de compétences ;
- les temps jugés nécessaires et/ou effectivement consacrés ;
- les diverses dimensions qualitatives de la prestation : démarche, adaptation aux besoins spécifiques du client, ...» (De Bandt [1995], p. 100).

Une fois de plus, l'évaluation correcte de ces compétences dépend des compétences de l'utilisateur, donc, entre autre, de son niveau de maîtrise des technologies mises en œuvre.

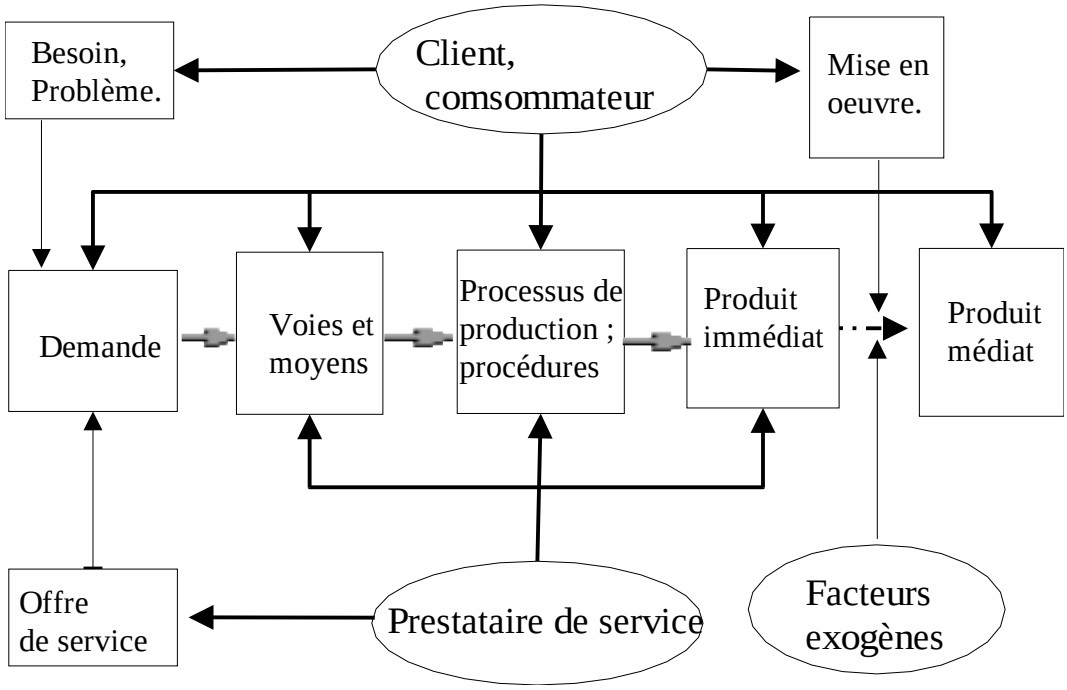
On notera que les obstacles à la réalisation effective du produit peuvent aussi venir du comportement du demandeur, autant que du producteur, s'il n'a pas les capacités nécessaires pour exprimer correctement ses besoins ou s'il choisit un offreur qui n'a pas pas les compétences pour produire le service espéré[30]. On est dans une relation d'échange présentant un fort risque de «sélection adverse», c'est-à-dire où il existe une réelle difficulté à évaluer, classer et donc sélectionner les différentes offres. Il y a un risque important de ne pas choisir la bonne offre, ou même, simplement, une offre adaptée à ses besoins. Une fois les partenaires choisis, il faut s'assurer que leurs efforts correspondent bien à ceux attendus. Il s'agit là de ce qu'on définit en économie sous le nom de «hasard moral». C'est un problème proche de celui qui se pose à une entreprise qui doit recruter un collaborateur (Baudry [1995]). Enfin, cette relation d'échange a une durée bien plus importante que le temps nécessaire au transfert de propriété d'un bien tangible, ce qui fait que les aléas sont bien plus importants. Les différentes étapes de cette relation sont résumées dans la figure 1.1, tirée de De Bandt [1998].

La production de services est donc caractérisée par l'existence d'«asymétries d'information[31]» fortes, généralisées à l'ensemble de la relation d'échange et dont le niveau est

[30] De Bandt ([1998], p. 62) souligne à ce propos que «[d]ans tous les cas, le produit est spécifique, parce qu'il doit correspondre à des besoins (une question, un problème) très particuliers du client/consommateur. Et ce produit est incertain, en ce sens qu'il y a incertitude sur le produit en raison de la double complexité complémentaire à la fois de la question et de la réponse, qui tient à leur multidimentionnalité».

[31] C'est-à-dire par le fait que les partenaires de l'échange ne disposent pas tous deux du même niveau d'information ni sur les capacités, sur la motivation de l'autre partenaire, ni sur la difficulté (technique,

Figure 1.3 — Processus de production de service (co-production bilatérale).



source : De Bandt [1998], p. 68, schéma 1.

fonction du niveau de compétence du client, autant que de celui du partenaire.

Cela a pour conséquence un fonctionnement très imparfait des marchés des services (et notamment des services informatiques). Les études des relations clients/fournisseurs le montrent bien : De Bandt [1995] signale «des degrés très élevés d'insatisfaction réciproque dans les relations entre [clients et prestataires de services]»[32]. Lorsqu'on s'intéresse aux enquêtes de satisfaction sur les produits informatiques[33], on constate une satisfaction vis-à-vis de l'outil informatique, mais une insuffisance dans le service après-vente, surtout dans

...) à remplir effectivement le service attendu.

[32]Il s'agit d'une enquête menée sur des services «informationnels», qui sont définis d'après la terminologie de Baumol et al. [1989], comme les activités de service à niveau de compétence élevé et très élevé.

Ces activités sont typiquement des services d'aide et/ou d'intervention et elles sont caractéristiques des activités que l'ont rencontre dans l'informatique.

Concernant spécifiquement l'informatique, Dréan [1996], ancien dirigeant de sociétés de services en informatique, va jusqu'à écrire, au sujet des prestations d'assistance technique que «à la limite, peu importe que le travail soit inutile, de mauvaise qualité ou médiocrement production tant que le client paie de bonne grâce» (p. 276) et au sujet de l'activité de développement que «[la nécessité] de promettre beaucoup pour le prix le plus faible possible [fait que] d'une certaine façon, les qualités qui sont nécessaires pour réussir dans l'exécution d'un projet (lucidité, prudence, honnêteté, ...) sont autant de raisons potentielles de ne pas emporter les contrats» (p 277).

[33]Nous nous référons ici à l'enquête de satisfaction réalisée chaque année par l'hebdomadaire 01 Informatique auprès des grands comptes français (enquêtes de 1998, numéro 1521, de 1999, numéro 1566 et de 2000, numéro 1612).

le logiciel (1998). La tendance de fond est que le client «recherche un meilleur support avant et après-vente, une aide pour résoudre ses difficultés et satisfaire ses besoins» (1999).

Le deuxième problème que pose l'évolution des besoins et des technologies informatiques est celui de la production de standards d'échanges ouverts.

La difficulté de produire des standards ouverts de qualité avec le système actuel de production des logiciels.

Comme l'explique David ([1987], p. 211) : «*The establishement of standards has greatest significance when economic agents cannot assimilate without substantial costs all the relevant information about the commodities that may be exchanged with other agents, and the processes by means of which those goods and services can be produced*». Nous avons bien vu l'importance de la standardisation du système d'exploitation et de l'architecture dans le développement de l'industrie informatique. La notion de standard représente «des réalités très diverses : des caractéristiques simples qui ont souvent un aspect relativement arbitraire (écartement des voies ferrées), [...], des spécifications plus complexes de dispositifs techniques [...] matériels [...] ou immatériels [...], voire ces dispositifs techniques eux-mêmes (logiciels ou matériels)» (Horn [1999], p. 93). Mais c'est toujours un «langage commun qui permet d'assurer la production, l'échange, mais aussi la compatibilité de la production des biens et des services» (Brousseau [1993], p. 257).

Horn ([1999], p. 94) remarque qu'il existe deux types de standard. Si ses caractéristiques sont définies de façon publique (souvent ex-ante, comme c'est le cas pour un processus de normalisation) chacun peut utiliser le standard librement, sans qu'il y ait rivalité ou exclusion de l'usage ; c'est alors un bien collectif «pur», pour reprendre les termes de Foray [1995]. S'il est possible de protéger le standard (par des systèmes de protection intellectuelle), s'il y a possibilité d'exclure de l'usage, sans qu'il y ait rivalité, c'est un bien public mixte avec externalités (Crozet [1997]). Dans le premier cas, nous parlerons de norme, dans le second, de standard de fait, ou de standard[34]. L'émergence d'un standard (indus-

[34]Nous nous appuyons ici sur le travail de Danièle Benezech, qui signale que «le standard est la référence du marché (ou de l'industrie), la norme est la référence officielle (reconnue juridiquement) ; les deux peuvent correspondre mais fondamentalement cette correspondance n'a rien de systématique» (Benezech [1995], pp. 9-10). Cette définition est compatible avec la classification que font David & Greenstein [1990] des différents types de «standards», à ceci près, comme le souligne Benezech, qu'ils ne s'intéressent qu'aux

triel) peut alors être due à une normalisation, voire à une réglementation (normalisation imposée par la puissance publique), ou être due au marché, comme c'est le cas, la plupart du temps, en informatique.

La normalisation de certains composants logiciels permettrait que se coordonnent les différents offreurs de technologies, mais aussi que les demandes des utilisateurs (et notamment des entreprises) en terme d'interopérabilité et de stabilité des solutions soient prises en compte. Le problème de la standardisation par le marché étant, nous l'avons vu dans l'introduction, que si une entreprise contrôle un standard, elle peut abuser du pouvoir de monopole que cela lui donne en changeant le standard (ce qui oblige les utilisateurs à racheter le produit) ou en profitant du fait qu'elle connaisse le fonctionnement intime de ses composants pour produire des composants complémentaires mieux adaptés et donc plus efficaces que ceux que pourraient proposer ses concurrents.

Il y a bien eu de nombreuses tentatives pour normaliser des interfaces, notamment chez les fabricants de systèmes Unix, sous la pression des clients, des «grands comptes». Des alliances se sont même mises en place pour développer des standards plus proches des normes, surtout dans la galaxie Unix (Open System Fundation ou Unix International), avec un succès mitigé (développement d'un système de gestion graphique, XWindows par le X consortium et d'une interface homme-machine le «Common Desktop Environment»). Les producteurs de matériel de l'industrie du PC sont de grands producteurs de normes (c'est-à-dire, dans ce cas là, de spécifications techniques négociées et publiques) qui se transforment souvent en standards (bus permettant de relier les composants et les périphériques, comme les bus ISA, PCI et maintenant USB ou types de mémoires, comme la mémoire SDRAM).

Mais, globalement, on peut dire qu'en informatique, les processus de normalisation sont généralement lents, hasardeux et pour tout dire le plus souvent inefficaces (Dréan [1996], pp. 55-57), et produisent des normes a minima[35]. Il y a donc matière à des évolutions organisationnelles qui permettraient de produire des logiciels de meilleure qualité, qui

technologies et que nous ne faisons pas vraiment la distinction entre technologie et produit.

[35]Comme la norme POSIX qui garantit qu'un logiciel développé en respectant ces recommandations pourra être compilé et fonctionner avec les différents systèmes d'exploitation qui respectent cette norme.

amélioreraient la production de normes et qui faciliteraient la relation de service entre utilisateur et producteur. Le Libre propose-t-il des pistes d'évolution intéressante de ce point de vue là ? C'est en tout cas l'idée que nous allons maintenant défendre.

1.3.3 Le Libre, une réponse à ces problèmes ?

Nous allons d'abord rappeler qu'aujourd'hui, au delà d'Internet, le succès des logiciels libres est essentiellement dû à leurs qualités techniques, notamment à leur fiabilité, au fait qu'ils respectent les standards et qu'ils sont évolutifs[36]. Nous expliquerons cela par le fait que le Libre, grâce à l'ouverture, grâce à la production coopérative, permet de construire des normes de qualité et d'interopérabilité. Mais d'autres systèmes, plus proches de la production propriétaire, essaient actuellement de s'en inspirer pour améliorer la qualité de cette production. La présentation de ces systèmes nous amènera à définir les conditions d'une diffusion du modèle Libre, conditions que nous étudierons dans le paragraphe suivant.

Un système qui facilite l'interopérabilité et qui permet de construire des normes de qualité.

Le Libre permet de construire des normes de qualité.

Il existe d'abord un certain nombre d'éléments de base que l'on retrouve dans tous les projets de logiciels libres, les «normes de définition» (pour reprendre la terminologie de David [1987]). «[Les producteurs] acceptent [...] volontairement de respecter ces normes puisque c'est par elles que passe l'appartenance à une communauté. Elles correspondent au référentiel sur la base duquel s'élaborent les comparaisons et les jugements de valeur. À partir de cette évaluation, il est possible d'énoncer des seuils minima dont le respect est conforme à l'intérêt général de la collectivité.» (Benezech [1995], p. 465).

Dans les projets libres, il s'agit du respect de certains langages de programmation et d'un certain nombre de règles de programmation. Il suffit de se souvenir que les pre-

[36] À nouveau, nous renvoyons, pour illustration aux articles d'Alain Lefèvre, http ://solutions.journaldunet.com/0101/010111_decrypt_oss.shtml.

Il faut se rappeler que l'idée de logiciel «libre» est lancée par Stallman en 1983 parce qu'il n'était pas satisfait de l'efficacité avec laquelle les logiciels commerciaux répondaient à ses besoins en terme de technologies d'utilisation et notamment à l'évolution de ses besoins.

miers logiciels libres créés et adoptés ont été des outils de développement (et notamment des compilateurs, des interpréteurs et des outils d'aide à la programmation). Ce sont les logiciels qui permettaient de normaliser les langages informatiques, qui sont la base commune minimale nécessaire pour dialoguer. D'autre part, les règles qu'il faut suivre lors du développement d'un projet libre sont souvent très contraignantes[37].

Tout ceci contribue à garantir des seuils minimums de robustesse du logiciel, «des normes de qualité, qui impliquent des procédures de certification et de mesure» (Benezech [1995], p. 465). Cela rend plus lisible le code produit et facilite sa correction. Enfin l'ouverture, la libre disponibilité des sources font que les utilisateurs peuvent effectivement tester les logiciels, étudier leur code et finalement le corriger s'ils trouvent des erreurs. Or, plus il y a de contributeurs, plus la chance qu'un des contributeurs trouve une des erreurs augmente. Augmente aussi la chance qu'il y en ait un qui sache la corriger[38].

Jusqu'à maintenant, cette dynamique de fonctionnement a produit des résultats remarquables au niveau de la qualité des logiciels produits et de la rapidité de leur amélioration[39]. Dans la plupart des tests portant sur la performance des systèmes (stabilité, rapidité) ou leur niveau de sécurité, on retrouve le ou les logiciels libres aux premières places.

Évidemment, ces tests peuvent toujours prêter à contestation ; cependant, tous ne sont pas faits par des entreprises défendant le logiciel libre et la plupart sont en sa faveur[40]. Il est aussi intéressant de noter que quand un test détecte une faiblesse dans le service rendu par un logiciel libre, des développeurs travaillent pour le corriger (Wheeler [2001] illustre ce fait dans une étude de cas).

De même que la qualité des logiciels libres s'explique d'abord par le respect de normes

[37] Voir http ://www.gnu.org/prep/standards.html pour les recommandations techniques à suivre pour programmer des logiciels GNU.

[38] «Give a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone» (Raymond [1998a]).

[39] La communauté des utilisateurs de Linux a reçu deux ans de suite le prix de la meilleure assistance technique (voir http ://www.infoworld.com/cgi-bin/displayArchive.pl ?/98/05/poy6a.dat.htm).

Lakhani & von Hippel [2000] ont montré l'efficacité de ce système dans le cas d'Apache.

[40] En terme de performances, GNU/Linux était le leader en mai 2001 dans le test comparatif de ZDNet portant sur le support de bases de données (source : «Take that ! Linux beats MS in benchmark test», http ://www.zdnet.com/zdnn/stories/news/0,4586,2760874,00.html.

Les résultats de nombreux tests comparatifs sont disponibles sur les sites http ://www.spec.org et http ://www.kegel.com/nt-linux-benchmarks.html (ce dernier s'intéressant d'abord aux comparaisons NT/Linux). Enfin, Wheeler [2001] a recensé nombre de données (test, parts de marché, etc.) sur les logiciels libres et la plupart des chiffres que nous donnons ici ont été trouvés grâce à ce recensement.

Tableau 1.6 — stabilité de différents logiciels.

Étude des principaux logiciels serveurs du Web.

Temps d'indisponibilité	**Apache**	**Microsoft IIS**	**Netscape**	**Andere**
Septembre	5,21	10,41	3,85	8,72
Octobre	2,66	8,39	2,80	12,5
Novembre	1,83	14,28	3,39	6,85
Moyenne	2,23	11,03	3,35	9,21

source : étude de Syscontrol AG sur les 100 premiers sites suisses
http ://www.syscontrol.ch/e/news/Serversoftware.html.

Comparaison de la stabilité de Gnu/Linux et de Windows NT.

Sur un an	**Gnu/Linux**	**Windows NT**
Nombre de crash	1	68
Temps total de réparation (en heures)	4	65
Disponibilité	99,95	99,26

source : test de Bloor Research (http ://gnet.dhs.org/stories/bloor.php3/).

de programmation et par l'existence de beta-testeurs, de même ces éléments favorisent le respect des normes de compatibilité.

Le libre, un outils pour construire des normes de compatibilité.

Le deuxième type de logiciel développé en libre a été les logiciels de réseau, donc des logiciels concernés en premier lieu par les problèmes d'interopérabilité. À côté des organismes classiques de normalisation (ISO, ANSI, W3C) qui proposent des normes sur les langages ou les protocoles de réseau, etc., le Libre apparaît comme un système qui permet de garantir leur mise en application dans des produits et, éventuellement, de développer des normes complémentaires (ce sont des normes car elles sont publiques et librement utilisables). Les développeurs d'Apache, imposent ainsi que les normes publiques (adoptées ou créées par exemple par le W3C) soient respectées par les autres acteurs du marché[41].

Ce qui garantit le mieux l'interopérabilité des logiciels libres est sans doute l'ouverture de leur code source et le fait qu'ils sont produits par plusieurs personnes. Pour que les développeurs puissent travailler en parallèle, il faut que les interfaces de communication soient publiques, normalisées, et la compatibilité du logiciel avec les logiciels complémentaires parfaite. L'ouverture du code permet de le vérifier et, si ce

[41] Lors du passage de la norme http/1.0 à la norme http/1.1., AOL a essayé de freiner cette évolution ; les serveurs qui utilisaient Apache et http/1.1 n'étaient plus accessibles pour les clients d'AOL. Parce que cela représentait une grande partie d'Internet, AOL a finalement dû accepter cette évolution. Voir http ://httpd.apache.org/info/aol-http.html.

n'est pas le cas, de modifier les logiciels pour leur faire respecter la norme. Ceci fait dire à Zimmermann [1999], à propos de la façon dont on peut assurer l'interopérabilité des logiciels, que, si la solution des systèmes ouverts a été mise à mal par un récent retour aux cloisonnements des systèmes propriétaires, celle des logiciels libres pourrait contribuer à un dépassement des limites et contradictions des systèmes de propriété intellectuelle et par là même impulser une profonde reconfiguration de l'industrie du logiciel.

Mais est-ce que cela se fera autour du système Libre ou d'un système qui s'en inspirerait ? Avec la marchandisation de plus en plus grande des logiciels libres, ce système (ou les logiciels qu'il a produit) ne vont-ils pas être absorbés par les producteurs de logiciels propriétaires ? Le Libre semble bien, aujourd'hui, à la croisée des chemins. Et la direction qu'il prendra dépendra essentiellement de sa capacité à intégrer la marchandisation, sans doute nécessaire pour diffuser massivement ses logiciels, et à créer des technologies d'utilisation plus efficacement que les autres systèmes de production.

Le Libre à la croisée des chemins, entre marchandisation et imitation.

Jusqu'à maintenant, le Libre est un système de production créé par des utilisateurs-designers, pour des utilisateurs-designers. Peut-il dépasser ce stade ?

Jusqu'à peu, les logiciels libres étaient auto-financés par leurs utilisateurs qui passaient du temps à développer et à corriger le logiciel. Ils en retiraient un bénéfice personnel, en se formant, en développant une réponse à un besoin technique, par exemple. Tant que cette collaboration était organisée de manière non marchande, les contributeurs estimaient sans doute que le bénéfice collectif dégagé, les logiciels libres ainsi créés, étaient une rémunération suffisante pour leur travail[42].

Avec l'arrivée d'agents du secteur marchand, d'autres motivations apparaissent, souvent en contradiction avec les objectifs du groupe en terme de qualité et de stabilité du

[42]Certains auteurs ont aussi signalé, à juste titre, qu'une partie des investissements est prise en charge par la collectivité, qui paye les chercheurs qui développent des logiciels libres (Notamment Genthon & Phan [1999], Horn [1999], Zimmermann [1999]). Il faut aussi se demander si ce financement, partiellement sur des fonds publics, est efficace et ne représente pas une concurrence déloyale pour les producteurs privés de logiciel. Mais il ne faut pas négliger la contribution des étudiants, des particuliers qui faisaient ce développement sur leur temps libre, ni l'apport de la production publique aux entreprises de logiciel. Après tout, SUN s'est lancé en «récupérant» Unix et des innovations développées à Standford.

logiciel : impératifs de production (respect de délais, utilisation d'outils ou de formats imposés par le client) ou obligations de rentabilité, par exemple. Inversement, certains utilisateurs qui souhaiteraient contribuer ou adapter un logiciel à leurs besoins ne le peuvent pas, par manque de compétence ou par manque de temps. Il faut donc que quelqu'un s'en charge. Seront-ce les producteurs initiaux, pour qui ces développements ont peu d'utilité, peut-on envisager que des entreprises le fassent et, dans ce cas, comment se financent-elles ?

On voit bien que ces deux problèmes sont liés : il s'agit de comprendre comment l'organisation libre peut mettre en place un système d'incitation, c'est-à-dire un système qui permette à tous les utilisateurs de faire prendre en compte leurs besoins dans le respect des règles et de la structure du système de production.

Il nous faut aussi nous interroger sur l'efficacité de cette production.

La production privée, centralisée, permet de mutualiser les coûts de développement de logiciels, même si cela ne va pas sans une certaine inefficacité. Pour que le Libre s'impose, il faut qu'il soit plus efficace que cette organisation. Or, les producteurs de logiciels propriétaires se sont inspirés du Libre pour créer de nouvelles licences qui permettent aux utilisateurs d'avoir accès aux codes sources, même si les modifications apportées restent la propriété du créateur du logiciel. Ce sont les licences de SUN (SUN Comunity License) ou d'Apple. Théoriquement, elles devraient permettre d'améliorer la qualité du logiciel et de garantir l'interopérabilité (puisqu'on peut corriger les erreurs et étudier comment communiquent les logiciels), tout en garantissant une source de revenu au producteur du logiciel (il reste le seul à pouvoir le vendre), donc en préservant son incitation à innover. Pour cette raison, Genthon & Phan [1999] considèrent que ces systèmes hybrides sont une réponse beaucoup plus efficace que le Libre à l'évolution des marchés informatiques. Est-ce vraiment le cas ?

Finalement, il faut non seulement que le Libre soit un système de production cohérent et pérenne, mais aussi qu'il soit plus efficace que les autres systèmes s'il veut s'imposer, se diffuser au-delà du public initial des utilisateurs-développeurs, quitter le statut de système de production marginal et légèrement suranné pour devenir le système de production des

logiciels. Nous allons, dans la suite de ce travail, essayer de montrer que c'est le cas.

1.4 Conclusion

Le Libre, système de production coopératif, est d'abord apparu anachronique au vu de l'évolution historique de la façon dont est produit le logiciel. Mais nous avons ensuite montré que l'industrie du logiciel est une industrie qui a beaucoup évolué, remettant à chaque fois profondément en cause l'organisation de production du logiciel précédente.

Notre étude a permis de nous rendre compte que la séquence de ces évolutions est toujours la même : à cause des progrès technologiques, l'objet informatique permet de répondre à de nouveaux besoins. Ces nouveaux besoins changent la façon dont les utilisateurs considèrent l'objet logiciel (successivement partie inséparable de la machine, résultat de la production d'un service et enfin objet). Ajoutés aux insatisfactions nées de certaines inefficacités de l'ancienne organisation de production, ils poussent les utilisateurs à en sélectionner une nouvelle, qui devient rapidement dominante (et qui, si elle ne fait complètement disparaître la précédente, la marginalise). La question qui s'est alors posée était de savoir si de nouveaux besoins sont apparus, s'ils ont fait évoluer la demande, et si le Libre y répond mieux que les systèmes anciens.

Nous avons répondu positivement à la première partie de cette question : avec l'arrivée d'Internet, avec la multiplication des supports de traitement de l'information (ordinateurs, téléphones, Psion), on est entré dans l'ère de l'échange d'information et de la mobilité. Les logiciels sont aussi de plus en plus produits sous forme de composants, capables d'effectuer chacun une tâche élémentaire et qui sont assemblés pour répondre à un besoin en terme de caractéristiques d'utilisation (ce qui permet de profiter des économies d'échelles dues à la réutilisation des composants tout en rendant plus flexible et plus évolutive la solution). Cela a fait évoluer la demande, vers du «sur-mesure de masse», c'est-à-dire des offres basées sur des logiciels standards adaptés aux besoins de chaque utilisateur. Cela oblige les producteurs à intégrer dans leurs offres une plus grande part de service, mais aussi à garantir que les composants utilisés fonctionnent correctement, qu'ils sont compatibles et que cette compatibilité ne sera pas remise en cause dans le futur.

Nous n'avons fait que répondre partiellement à la deuxième partie de la question : le Libre apporte effectivement une réponse aux problèmes de qualité du logiciel. Comme les logiciels libres sont à la base d'Internet, ils répondent bien aux nouveaux usages qu'entraîne sa diffusion. Comme ils sont ouverts et que le système de production est très rigoureux (notamment sur le respect des normes sur les langages et les protocoles d'échange), ils sont souvent de meilleure qualité que les logiciels propriétaires concurrents et ils sont plus adaptables.

Reste à montrer qu'au delà d'Internet, le Libre peut être un système de production viable économiquement et capable de prendre en compte les besoins de l'ensemble des utilisateurs. C'est ce que nous ferons dans le prochain chapitre. Reste aussi à vérifier que ce système est plus efficace que l'organisation actuelle, et peut la remplacer. Le troisième chapitre sera consacré à cette question.

Deuxième Chapitre

Fonctionnement du Libre.

LE chapitre 1 se terminait sur cette question : le Libre peut-il être un système pérenne pour produire du logiciel, c'est-à-dire un système qui permettrait à tous les utilisateurs de faire prendre en compte leurs besoins ? Nous avons expliqué dans l'introduction que le logiciel était un bien public et que la production de ce bien posait problème car les agents n'étaient pas obligatoirement motivés pour contribuer au financement de sa production (que ce soit en espèce ou en temps de travail). Nous avons aussi vu qu'il fallait alors construire des systèmes d'incitation pour remédier à ce problème.

Dans ce chapitre, nous allons nous demander si le Libre peut être un de ces systèmes. Nous allons donc étudier le système de production Libre pris isolément, en supposant que c'est le seul système de production existant. Nous allons voir si les mécanismes, les règles mis en place par les inventeurs de cette organisation assurent, par leur seule existence, que suffisamment d'agents vont contribuer pour répondre aux besoins de l'ensemble des utilisateurs, aujourd'hui et à long terme. Cela peut sembler un peu spécieux alors que de nombreux développeurs donnent comme raison pour contribuer au Libre la volonté de s'opposer à un logiciel propriétaire existant. C'est pourtant nécessaire si l'on veut prouver que le Libre est capable de remplacer l'organisation propriétaire, ce qui veut dire fonctionner sans l'aiguillon que lui apporte cette concurrence.

Nous allons mener cette analyse en trois étapes.

Nous commencerons par étudier comment est organisée la production de logiciel, quelles sont les règles que les producteurs doivent respecter, comment ils peuvent proposer une contribution, comment cette contribution est évaluée et intégrée dans le logiciel. Nous défendrons le point de vue que le Libre est une organisation, c'est-à-dire un système de production très structuré, et dont la cohérence interne est favorisée par le fait qu'il existe un système de récompenses et de sanctions qui assure sa cohésion.

Ensuite, nous essayerons de comprendre les motivations des agents producteurs pour accepter ce système de récompenses et de peines[1]. Nous verrons que tous les agents,

[1]On pourra nous reprocher de prendre le parti-pris d'interdit aux développeurs de créer avec pour seul but l'intérêt général. On pourra d'abord répondre que le bénévolat, pour qu'il fonctionne, doit être intéressé. Ainsi, Blanchet [1990] écrit que : «[...] la très grande majorité des enquêtes et études consultées sont venues confirmer une impression que je possédais préalablement, à savoir que ce qui constitue la source de motivation la plus fréquente et la plus productive, est une réponse directe et bien orientée à un intérêt personnel du bénévole». En fait, c'est surtout un problème d'interprétation : en affirmant la motivation,

utilisateurs-développeurs, entreprises utilisatrices, producteurs de logiciels, peuvent trouver des raisons non financières pour contribuer, à un moment ou à un autre, à la production d'un logiciel.

Mais nous montrerons aussi que pour assurer la stabilité à long terme de la production et pour adapter les logiciels libres aux besoins des utilisateurs non développeurs, une structure d'incitations est nécessaire, c'est-à-dire un système qui permette de rémunérer des entreprises en échange de ce travail d'adaptation. La troisième partie de ce chapitre étudiera les incitations que le Libre développe.

2.1 Le Libre, une organisation très structurée.

Dans cette partie nous allons analyser le fonctionnement du système de production libre. L'argument défendu sera que chaque projet de développement libre est une organisation, avec ses règles, ses relations d'autorité, ses systèmes d'information et de décision, le tout soutenu par une histoire et par des institutions[2] (comme le droit d'auteur ou la GPL).

Pour le défendre, nous allons décrire le fonctionnement du système de production libre, avant de l'analyser à la lumière de la théorie sur les organisations et les marchés.

2.1.1 Une analyse du fonctionnement de la production libre.

Présentation et critique des analyses classiques sur la production libre.

Les principales analyses existantes sur le système de production libre sont dues à des producteurs de libre, comme Raymond, qui ont essayé de comprendre et de théoriser le fonctionnement de l'organisation à laquelle ils participaient. Pour ce dernier ([1999b]),

nous supposons simplement que le développeur se sent valorisé lorsqu'il défend l'intérêt général, lorsqu'il pense faire ce qui est juste.

[2]Nous entendons le terme «institution» dans le sens que lui ont donné Davis & North ([1971], p. 133), c'est à dire : «a set of fundamental political, social and legal ground rules that govern economic and political activity (rules governing elections, property rights and the rights of contracts are examples of the ground rules)».

Ménard ([1990], pp. 16-17), en propose une traduction : «par institutions, on entendra *un ensemble de règles socio-économiques, mises en place dans des conditions historiques, sur lesquelles les individus ou les groupes d'individus n'ont guère de prise, pour l'essentiel, dans le court et le moyen terme. Du point de vue économique, ces règles visent à définir les conditions dans lesquelles les choix, individuels ou collectifs, d'allocation et d'utilisation des ressources pourront s'effectuer.*» (les italiques sont de l'auteur.)

le processus de production est assez similaire à un marché parfait ou à un processus darwinien, parce que différentes offres sont disponibles, accessibles à tous et que ce sont les meilleures qui sont adoptées par les utilisateurs. C'est ce qu'il appelle la production «bazar» et ce bazar permettrait de s'affranchir des limites imposées aux développements hautement centralisés de la production fermée (mais aussi de la production des premiers logiciels libres produits par la FSF) et ce, avec une meilleure qualité et une plus grande rapidité.

Pourtant, s'il est vrai que les relations entre utilisateurs et producteurs interviennent plus tôt dans le processus de production, que l'on peut appeler «développement coopératif», cela reste un processus, qui intègre, au fur et à mesure que le logiciel se développe, les demandes de nouvelles catégories d'utilisateurs. Il s'agit avant tout de la production de logiciels et, pour analyser cette production, il est utile de se remémorer les résultats des études sur le sujet.

Qu'il s'agisse de production «fermée» ou de production «ouverte», produire du logiciel signifie, aujourd'hui comme hier, faire travailler une équipe de développeurs chargée d'identifier des besoins, de traduire ces besoins en terme de technologies d'utilisation et de réaliser ces technologies d'utilisation.

La production de logiciels a été étudiée et théorisée par Brooks [1974] et ses travaux sont toujours valables. Sa conclusion la plus importante est qu'à partir d'un certain seuil les coûts de coordination deviennent si importants que rajouter des développeurs ne fait que retarder le processus de production du logiciel. Pourtant, il semble que les développements libres échappent à cette limite. Le processus de production traditionnel est centralisé et hiérarchisé en différentes étapes (généralement la demande du marché, l'architecture du système, l'implémentation, l'intégration, les tests sur le terrain et le support). Parfois, des utilisateurs finals sont intégrés dans les deux premières étapes (la définition de la demande et l'architecture) et la dernière partie des tests peut être confiée à un groupe de «beta-testeurs». Le support est traditionnellement une activité de service, souvent externalisée par le vendeur, comme c'est le cas pour les logiciels de Microsoft.

À l'inverse, Raymond compare l'organisation libre à un marché où la coopération aurait

émergé autour des meilleures solutions et des meilleurs programmes. Les explications de Raymond [1998b], centrées sur les individus et proches de théories «hayekiennes» de l'ordre spontané («Hackers have defined their culture by a set of choices about the form which their competition will take») donnent une impression de règles auto-construites au sein de groupes auto-émergents. Cette explication, bien que très séduisante, ne correspond pas à la réalité de l'organisation de la production ; elle n'est pas plus défendable en ce qui concerne la construction des règles qui organisent cette production.

Au niveau théorique, il a été montré qu'un marché spontané est tout sauf spontané et que toutes les organisations collectives ne pouvaient pas être analysées en terme de marché : faisant une analyse critique d'Hayek, Vanberg [1986] a notamment mis en lumière que des structures institutionnelles doivent pré-exister pour permettre la construction d'un marché, qui est alors tout sauf auto-organisé[3] . C'est sans doute même une «institution», dans le sens que lui donnent Davis & North [1971]. Il est d'ailleurs remarquable de constater que Kuwabara [2000], cherchant à établir un cadre théorique à l'analyse de Raymond, met en avant le rôle de certaines organisations clefs. Il apparaît finalement apporter la preuve de l'importance des institutions et des organisations dans la construction de ce phénomène[4]. Enfin, le système de production Libre n'a pas les caractéristiques d'un «marché» : au-delà de l'absence d'unité monétaire, nécessaire à l'existence d'un marché, les utilisateurs ne sont, par exemple, pas libres de proposer leurs contributions à n'importe qui.

Nous allons montrer que la production de logiciel libre n'est pas si éloignée de la production classique de logiciels, telle qu'on peut l'imaginer dans une entreprise. Elle est aussi structurée que cette dernière, mais elle est organisée de telle manière qu'elle favorise sans doute mieux l'innovation incrémentale et les retours entre utilisateurs et producteurs. C'est là sa principale innovation et sa principale qualité. Nous allons montrer que, contrairement à ce qu'explique Raymond, la production libre est très structurée. S'il n'existe pas de règles fixes et rigides, le développement de projets implique l'existence

[3] «A market is always a system of social interaction characterized by a specific institutional framework, that is by a set of rules defining certain restrictions on the behavior of the market participants, whether the rules are informal, enforced by private sanctions, or formal, enforced by a particular agency» (Vanberg [1986], p. 75).

[4] Nous avons développé cette discussion dans Jullien & Phan [1999]. Comme cet argument n'est pas essentiel dans notre démonstration que le Libre n'est pas un système qui fonctionne avec des règles proches de celles du marché, nous en resterons là.

d'une coordination centralisée et de certaines règles dans la construction des logiciels, ce qui nous éloigne du bazar comme de la sélection naturelle.

Le système de coordination des développeurs : un système proche du centralisme démocratique.

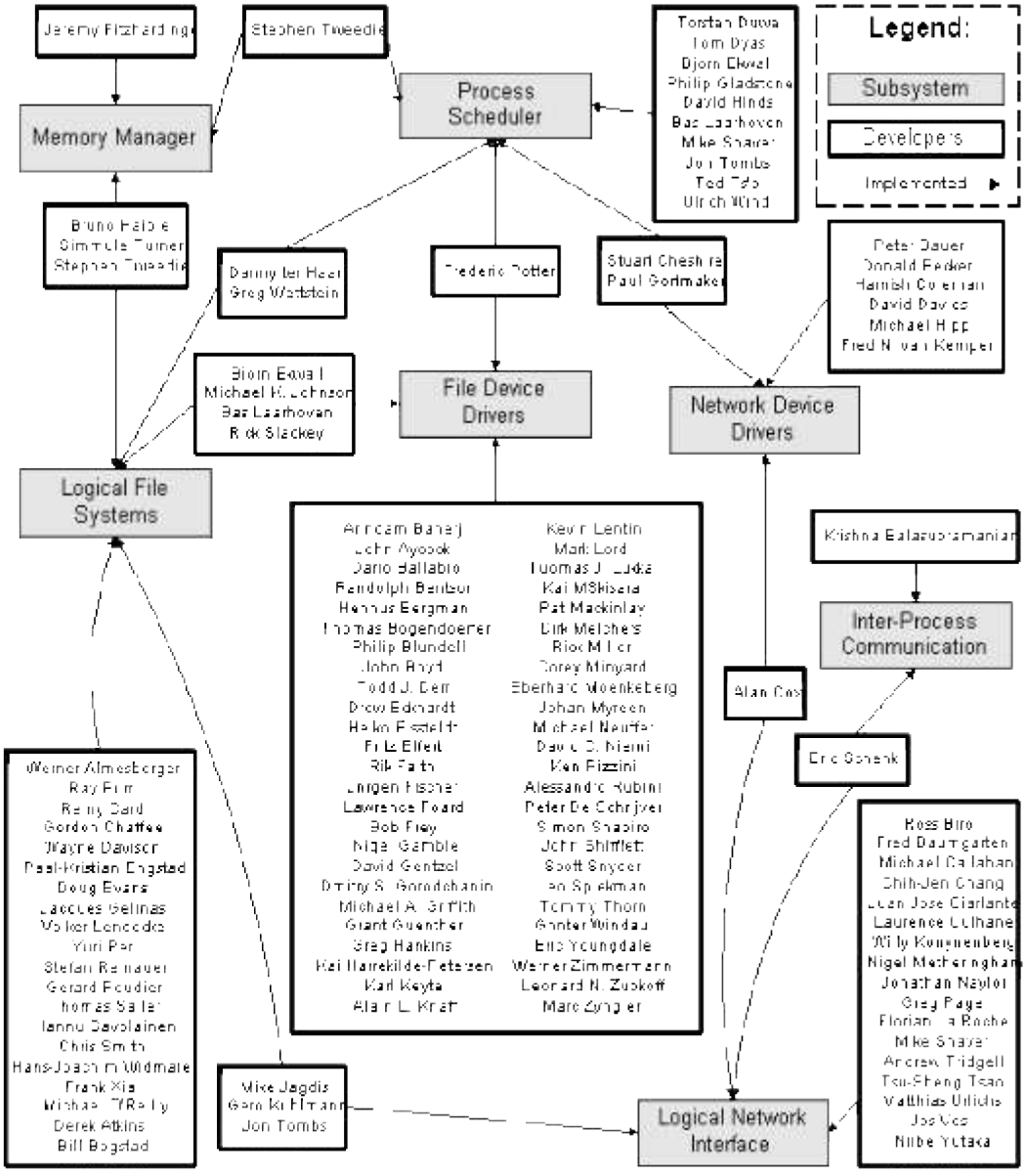
La coordination entre les différents développeurs est souvent assurée par un groupe de personnes, que nous allons appeler les «développeurs clefs». Ces personnes sont parfaitement identifiées et sont chargées d'une sous-partie du programme. L'étude de Mockus & al. [2000] sur le projet Apache souligne cet aspect : «these results indicate that despite broad overall participation in the project, almost all new fonctionality is implemented and maintained by the core group» ([Mockus & al. [2000], p. 268). Les développeurs clefs sont sélectionnés par les autres membres du noyau de développement (c'est le cas d'Apache) ou par le développeur initial du projet (c'est le cas de Linux), mais il y a toujours cooptation. Ce noyau reste ouvert aux nouvelles idées, aux nouveaux talents : on peut confier aux nouveaux venus la responsabilité de parties nouvelles du logiciel, souvent parce que ce sont eux qui ont commencé à les développer, mais la sélection de ces nouveaux membres se fait par le haut. On retrouve les mécanismes de la science, où on est coopté et accepté par ses pairs.

Autour de ce noyau, il existe effectivement de nombreux utilisateurs, développeurs le plus souvent, qui participent au développement, essentiellement en donnant des idées et en proposant des améliorations ou en détectant des erreurs[5]. Ce groupe est important pour le succès et l'évolution du logiciel mais il n'a que peu d'impact direct sur le volume de logiciels développés.

Les logiciels libres présentent une deuxième caractéristique, qui permet d'assurer la cohérence entre le noyau et les contributeurs extérieurs à ce noyau : dès le départ (ou au moins dès que le projet commence à prendre de l'ampleur), ils sont conçus de façon modulaire.

[5]«Of the top 15 problem reporters only three are also core developers. It shows that the significant role of system tester is reserved almost exclusively to the wide community of Apache users» (Mockus & al. [2000], p. 269).

Figure 2.1 — Répartition des responsabilités parmi les développeurs de Linux.



source : Bowman [1998].

Une construction des logiciels spécifique, par modules, qui permet et organise la division du travail.

La modularité, dans le cas du logiciel libre, signifie que les frontières du logiciel de base sont définies, que les interfaces de communication entre ce logiciel et les logiciels satellites, les «composants», sont publiques. On retrouve, une fois de plus, un problème de standardisation : il faut que le cœur du logiciel soit public et évolue de façon prévisible pour que les développeurs des applications périphériques aient une vision de l'évolution de ce standard et puissent y adapter leurs logiciels. La construction modulaire d'un logiciel permet à des groupes de développeurs de travailler relativement indépendamment, ce qui présente deux avantages : d'une part, les équipes de programmation restent petites, ce qui facilite la coordination et la prise de décision[6] et, d'autre part, les innovations et l'extension des fonctionnalités des programmes peuvent se faire plus facilement, par des ajouts successifs de modules.

Prenons quatre exemples, provenant de quatre «types» de logiciels, Gnome (une interface graphique), Linux (un noyau de système d'exploitation), Gnat (un compilateur) et Apache (un logiciel serveur pour le Web). Pour chacun de ces projets, on constate que c'est l'emploi de cette technique qui permet d'augmenter le nombre total des programmeurs sans que la coordination générale du projet en souffre.
Le projet Gnome est divisé en plus de 200 sous-projets, plus ou moins indépendants[7], ce qui fait dire à Zawinski que «most of the larger projects are also fairly modular, meaning that they are really dozens of different, smaller projects. So when you claim that there are ten zillion people working on the Gnome project, you are lumping together a lot of people who never need to talk to each other, and thus, aren't getting in each others' way.»[8]
Nous avons déjà présenté la structure du noyau Linux, qui n'est qu'un composant du système d'exploitation (Gnome en est un autre). La modularité est due à des choix techniques[9], mais c'est pour Torvalds un des facteurs clefs du succès de son logiciel : «the

[6] «In most open source projects» says Zawinski, «there is a small group who do the majority of the work, and the other contributors are definitely at a secondary level, meaning that they don't behave as bottlenecks.» (cité par Jones [2000], Zavinski a travaillé chez Netscape et au projet Mozilla).

[7] Voir http ://www.gnome.org et http ://cvs.gnome.org/bonsai/rview.cgi ?cvsroot=/cvs/gnome .

[8] Toujours extrait de Jones [2000]. La dernière partie de la phrase pose le problème central de la coordination des différents projets libres.

[9] «By allowing for kernel modules, hardware-specific code can often be confined to a module, keeping the core kernel higly portable». (Torvalds [1999], p. 38)

decisions that motivate portability also enable a large group to work simultaneously on parts of Linux without the kernel getting beyond my control» (Torvalds [1999], p. 39).
Bien que contrôlé assez étroitement par une entreprise, le projet Gnat permet aussi le développement de modules indépendants : l'adaptation de l'outil graphique de développement «Gtk» à Ada 95 a été faite par des étudiants; parce qu'il y avait une demande pour ce type de produit, elle a été intégrée dans les produits officiellement supportés par ACT (l'équipe de développement a depuis été embauchée dans cette entreprise).
Enfin, pour ce qui concerne le projet Apache, Behlendorf [1999], un des membres fondateurs, souligne que «in the Apache project, we were fortunate in that early on we developed an internal API to allow us to distinguish between the core server functionality [...] and most other higher-level functionalities [...] Having a really powerful API has also allowed us [...] to separate groups of dedicated developers. This freed the core development group from having to worry about building a 'monster'[...]», et aussi que «we have been also fairly successful (I think)in 'federalizing' the Apache process to sister projects.» (cité par Jones [2000]).

La segmentation des logiciels permet deux choses complémentaires : de développer les fonctionnalités grâce à une division du travail entre programmeurs et de faciliter le contrôle du noyau des développeurs sur le noyau du programme. Cela garantit une certaine stabilité de l'évolution de ce noyau et donc l'interopérabilité entre les différents modules. On constate, une fois de plus, le rôle central de ce noyau dans le développement du logiciel libre. Arrêtons-nous un instant sur les fonctions dévolues à ce noyau.

Le rôle du noyau des développeurs : toujours central, il évolue avec la diffusion du logiciel.

Nous allons montrer que le rôle, la fonction du noyau des développeurs évolue au cours du temps, passant d'un rôle de développeur à un rôle de gestionnaire de projet puis, finalement de garant de la qualité de ce projet[10].

[10]Edwards [2000] illustre ces trois phases en prenant l'exemple du projet Fetchmail (serveur de mail), qui depuis octobre 1997, a officiellement atteint le troisième stade.

Un premier rôle, celui d'initiateur de projet.

Souvent, ce noyau est à l'initiative du logiciel et c'est lui qui produit la première version du logiciel, version qui va servir à intéresser d'autres contributeurs au projet.

Lerner & Tirole [2000] disent à ce propos que le leader doit apporter une vision et que c'est en cela qu'il est la force directrice du projet. Edwards [2000] propose une analyse un peu différente : il signale qu'il n'existe pas de projets libres qui auraient démarré sans que les personnes qui demandent des contributions ne viennent avec «a working demonstration of their idea». Nous adhérons à son point de vue. En effet, tant qu'on reste au niveau de la vision, on reste au niveau de l'expression de besoins ou de problèmes. Mais c'est le développement d'un logiciel qui va fixer les technologies utilisées pour réaliser ces besoins, donc la façon dont ils sont effectivement compris et dont ils vont être effectivement réalisés. La production de code apparaît donc nécessaire pour fixer la direction du projet et aussi, nous le verrons dans l'analyse, pour donner aux développeurs des motivations suffisantes de contribuer à ce projet.

À ce stade, comme le fait remarquer Ousterhout [1999], les adopteurs du logiciel ressemblent au créateur. Ce sont des programmeurs «designers», comme les personnes du noyau et ils utilisent le logiciel car il leur permet de résoudre un problème. Le recrutement se fait donc sur des critères technico-fonctionnels (capacité du logiciel à répondre à un besoin et qualité technique de la base déjà produite). C'est une première étape importante pour le devenir du logiciel. Si celui-ci n'est pas construit, architecturé de façon suffisamment souple, la taille maximale efficace de la population de contributeurs, mais aussi la taille maximale que peut atteindre le logiciel sont rapidement atteintes car les problèmes de coordination, de lisibilité du code produit deviennent rapidement insurmontables[11].

Si cette étape est franchie, il s'agit de gérer le développement de ce logiciel, de coordonner les travaux sur le cœur du logiciel en même temps qu'on encourage le développement des modules périphériques.

[11] Afin de permettre le développement des contributions tout en conservant la cohérence du logiciel et la maîtrise sur son évolution, les initiateurs du projet Apache ont dû ré-écrire le logiciel, essentiellement pour le rendre plus modulaire, pour mieux fixer la frontière entre les tâches qui concernaient le noyau, le «moteur» du logiciel et celles qui devaient être traitées par des modules externes.

Un rôle de coordinateur pendant la période de diffusion.

Il faut alors coordonner les responsabilités et aussi coordonner les projets pour qu'ils soient compatibles et pour qu'ils ne se recouvrent pas, ce que Raymond ([1998 b] p. 15) résume par ces phrases : «who gets to make binding decisions about a project and who gets credit or blame for that» et «how to reduce duplication of effort and prevent rogue versions from complicating bug tracking». Ce qui veut dire gérer les co-développeurs et leurs co-développements ; la fonction du noyau se rapproche de celle d'un chef de projet. C'est aussi le stade où, petit à petit, les principales innovations se déplacent du cœur du logiciel vers les modules externes[12]. L'important, pour franchir cette étape est que les développements du logiciel suivent les évolutions de la demande.

Les utilisateurs «designers» sont rejoints par des utilisateurs «sophistiqués» qui, s'ils déploient le logiciel en entreprise, ont besoin d'outils de déploiement, de maintenance, etc. C'est pourquoi, au cours de cette phase, le rôle de chef de projet, centré sur la coordination des réponses à des besoins[13] évolue vers un rôle de «mainteneur» (Edwards [2000]) où il s'agit de garantir la stabilité de l'offre et sa pérennité dans le temps. À partir de ce moment-là, on peut dire que le logiciel est arrivé à maturité ; les activités de gestion et de maintenance deviennent prépondérantes.

Un rôle de mainteneur pendant la période de maturité.

Il s'agit, dans cette période, d'assurer les services d'«après développement» du logiciel, c'est-à-dire corriger les erreurs qui peuvent apparaître, adapter le logiciel aux nouveaux matériels ou aux nouvelles normes. Le travail peut continuer autour du noyau et le succès du logiciel fait qu'il est adopté par des utilisateurs différents des développeurs initiaux (parfois même par des utilisateurs «naïfs»), ce qui nécessite souvent le développement de nouvelles extensions pour faciliter son utilisation. Si les besoins de coordination et de compatibilité restent importants, le volume de travail diminue, l'importance du noyau aussi et on peut tout à fait imaginer que le nombre de développeurs participant au projet

[12] «A successful project should mature at some point, after which the pace of change within the project typically slows down. [...] But a this point, the most exciteing developments for Linux will happen in user space, not kernel space. The changes in the kernel will seem small compared to what's happening further out in the system.» Torvalds [1999].

[13] «The next best thing to having good ideas is recognizing good ideas from your users» ; «if you treat your beta-testers as if they are your most valuable resource, they will respond by becoming your most valuable resource» Raymond [1998a].

se réduise ; c'est même souhaitable en terme d'efficacité de l'allocation d'une ressource rare, les individus qui ont la capacité de développer et de coordonner des projets.

Cette analyse ne vaut, bien sûr que pour des projets de développement qui nécessitent la contribution d'une équipe importante de développeurs. Si le logiciel peut être programmé par une seule personne, la relation se limite aux échanges entre cette personne et les quelques utilisateurs qui ont pu le récupérer et l'adapter à leurs besoins propres. C'est une des raisons pour laquelle on constate statistiquement que la plupart des logiciels libres n'ont qu'un contributeur et que la plupart des contributeurs ne s'intéressent qu'à un logiciel[14] : c'est souvent celui qu'ils ont fabriqué et on est plus proche ici d'un développement à façon que de la production de progiciels. La différence entre ces projets et les projets réellement coopératifs se fait sur la capacité du logiciel à générer de nouveaux besoins et à intéresser de nouveaux utilisateurs au fur et à mesure de son développement.

De même qu'un projet libre peut rester au stade du développement individuel, à chacun des autres stades de la production il y a un risque que le projet s'arrête, si le noyau n'arrive pas à recruter ces nouveaux utilisateurs qui amènent de nouveaux besoins et donc de nouveaux développements. Si c'est dû à l'absence de besoins pour une telle technologie chez d'autres utilisateurs que ceux de la phase précédente, c'est alors un arrêt naturel. Si c'est à cause de difficultés dans la gestion du projet, dans la coordination ou la motivation des acteurs qui participent au développement du logiciel, c'est qu'il y a une faiblesse dans l'organisation de production. Il s'agit ici d'un problème classique dans la gestion de projet. Nous allons voir que des mécanismes, des règles permettent de faciliter cette coordination.

2.1.2 Les règles qui permettent d'assurer la stabilité de l'organisation.

Nous allons voir que ce système fonctionne grâce à des règles sociales fortes, qui s'appuient sur un certain nombre d'institutions, à tel point que nous nous demanderons si le Libre n'est pas une organisation[15].

[14] Voir les chiffres d'Orbiten [2000].

[15] C'est-à-dire, «*une procédure de coordination spécifique*, [...] qui assure l'allocation de ressources selon des modalités propres et [qui prospère], soit en raison d'insuffisances des mécanismes de marché, soit en raison de leurs inconvénients.» (Ménard [1990], pp. 16-17).

Notons que la deuxième partie de la définition, qui implique de comparer l'organisation Libre à d'autres systèmes de production, sera surtout pertinente dans la suite et précisément dans le chapitre suivant. Retenons simplement l'idée que le Libre permet de faire travailler les personnes et de les coordonner avec

Un système qui fonctionne grâce à des règles sociales fortes.

La duplication des efforts étant en grande partie due à la concurrence entre projets, ces deux choses sont bien sûr liées. On peut poser le problème en ces termes : si l'on suppose (provisoirement) résolue la question des motivations individuelles, l'ouverture favorise la copie des bonnes idées, même entre projets parallèles et concurrents (voir, par exemple la concurrence entre Emacs et XEmacs ou Gnome et Kde) et l'émulation qui naît de la concurrence entre développeurs peut même augmenter la production d'innovations ; mais cette concurrence peut aussi arrêter un projet car la ressource (les capacités de programmation) est rare et, lorsque des projets concurrents émergent, il peut arriver qu'aucun n'atteigne une taille critique pour franchir les différentes étapes du développement d'un logiciel libre. Ainsi la multiplication des projets autour du noyau Unix libre BSD, Open BSD, Free BSD, Net BSD, a ralenti le processus de développement et n'a pas permis la même dynamique que celle de leur concurrent Linux.

Ces problèmes de management sont connus des développeurs. Di Bonna, Ockman et Stone [1999] ont écrit à ce sujet que : «[m]aintaining control of an active Open Source project can be difficult. This fear of losing control prevents some individuals and many companies from active participation [..] Much like a fork in the road, a code base can at time diverge into two separate, incompatible, roads and never the twain shall meet. Look for instance at the multiple forks that the BSD-based OS have taken, leading to NetBSD, OpenBSD, FreeBSD, and many others.»

Les raisons qui poussent à la création de projets concurrents sont variables, de la différence dans les besoins et les objectifs, à des faiblesses dans la gestion du projet, en passant par des querelles technico-idéologiques (comme dans le cas de Gnome, qui a été créé en grande partie parce que le projet Kde utilisait des librairies graphiques non libres). Une fois encore, c'est un problème classique de gestion des groupes, l'originalité reposant sur les solutions que les acteurs ont construites, basées d'abord sur des règles sociales ; Raymond [1998b] met en lumières trois règles qui permettent de prévenir les conflits et aussi de développer des systèmes pour les régler :

– «there is strong social pressure against forking projects. It does not happen except

d'autres mécanismes que ceux du marché.

under plea of dire necessity, with much public self-justification, and with a renaming ;

- distributing changes to a project without the cooperation of the moderators is frowned upon, except in special cases like essentially trivial porting fixes ;
- removing a person's name from a project history, credits or maintainer list is absolutely not done without the person's explicit consent.»

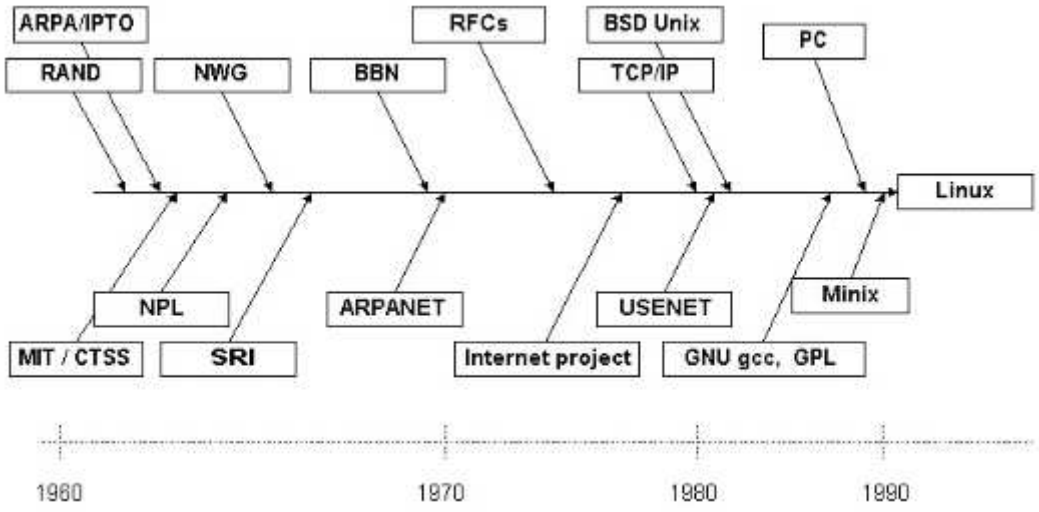
Les actions qui provoquent le conflit sont donc condamnées au niveau social. Il est important de noter que la seule sanction pour ne pas avoir respecté ce code est l'exclusion du groupe. Ce qui veut dire que l'appartenance au groupe est importante pour les développeurs. La structure «sociale», les règles basées sur le maintien de la cohérence du groupe, qui sont intégrées par les programmeurs, reposent sur cette sanction. La stabilité d'une telle organisation nécessite que les personnes qui sont recrutées partagent les mêmes «valeurs». Pour entrer dans le groupe (et notamment dans le noyau), les individus doivent prouver qu'ils partagent ces règles, par leur comportement dans les listes de diffusion et les news et par la qualité de leurs contributions. Ce sont des règles que l'on trouve dans le milieu universitaire, à cette différence, sur laquelle nous revenons une fois encore, qu'il n'existe pas d'organisation comparable à l'université ni de financement publique qui permettraient de rémunérer les efforts des acteurs. Ces règles apparaissent à première vue beaucoup plus informelles, tacites.

L'arrivée de nouveaux acteurs dans la production de logiciel libre, qui n'ont pas obligatoirement la même culture universitaire, pose le problème de la stabilité de ces règles, de la capacité du système à «contraindre» ces nouveaux acteurs à les intégrer. L'existence de nombreuses structures, d'organisations et d'institutions qui ont précédé les organisations de production de logiciel libre et qui l'encadrent aujourd'hui renforcent fortement ces règles.

Une organisation sociale construite et encadrée par de nombreuses institutions.

Nous avons rappelé, dans le chapitre 1, l'histoire de la production coopérative, en soulignant ses origines universitaires, mais aussi que cette culture de coopération avait été encouragée par les institutions américaines, par le financement de projets de recherche collectifs et par la construction de réseaux électroniques permettant l'échange de documents.

Figure 2.2 — Acteurs et logiciels clefs pour Linux.



source : Tuomi [2001].

Tuomi [2001] a étudié l'ensemble des groupes (ou «communautés») et des organisations qui ont précédé l'organisation de production de Linux et l'ensemble des technologies qu'ils ont produites et qui ont servi à construire ce logiciel. Cette étude montre clairement que Linux, comme les autres logiciels libres, est une conséquence de l'existence de ces structures et une évolution des logiciels qu'elles ont créés.

Et l'organisation de production, initiée par Stallman, est aussi une évolution de l'organisation coopérative «universitaire». Tout d'abord, ce mouvement n'a pas été initié par un organisme public, mais par une «association» américaine indépendante, la «Free Software Fundation», créée par Stallman. Cette association s'est donnée pour but de coordonner le développement de tous les logiciels nécessaires pour pouvoir utiliser un ordinateur et notamment le système d'exploitation (aussi de produire une partie de ces logiciels grâce aux dons faits à la fondation et à la vente des logiciels produits). Cela a permis d'activer et de canaliser les énergies des différents groupes de développement qui pouvaient exister.

L'autre institution créée par Stallman est la licence GPL. Cette licence est un contrat, basé sur le droit d'auteur (ou plutôt le copyright aux États-Unis), par lequel le producteur d'un logiciel autorise un utilisateur à l'utiliser, le modifier et le redistribuer à condition qu'il le fasse dans les mêmes termes et conditions[16]. Elle permet d'apporter une base

[16]Pour une analyse juridique de la GPL et notamment pour une étude de son applicabilité en France, on consultera le travail de Clément-Fontaine [1999].

juridique à l'activité coopérative car à partir du moment où un logiciel est distribué sous cette licence, il n'est plus possible de se l'approprier. Les utilisateurs sont sûrs d'avoir toujours accès aux sources du logiciels, donc aux modifications que les autres utilisateurs y apportent s'ils publient ce logiciel. Nous verrons dans la suite son intérêt pour les utilisateurs et les producteurs de logiciels.

Si l'on voit bien que ce système est loin d'être un bazar, on peut se demander en quoi il facilite la production des logiciels et notamment pourquoi il permet de motiver les agents à participer à cette production. Ce que nous allons expliquer, c'est que ce système possède deux caractéristiques économiques : d'une part, le fait que la production soit organisée en trois étapes correspond bien à une caractéristique de la production des biens publics ; d'autre part, le système Libre apparaît être une organisation (au sens économique du terme), ce qui facilite la coordination des producteurs.

2.1.3 La vision économique de ce système de production : une organisation de production d'un bien public au fonctionnement classique.

La production en trois étapes : une caractéristique de la production d'un bien public.

La production en trois étapes qui caractéristique le système Libre ne lui est pas propre. Marwell & Oliver [1993] se sont particulièrement intéressé aux motivations qu'un agent a à participer à la production de tels biens, alors qu'il n'y est pas obligé (C'est ce que Olson [1965] a appelé le problème du passager clandestin). Or, dans leur analyse, ils montrent que la fonction de production d'un bien public a comme propriété d'être du «troisième ordre» (ou, plus simplement, que son sens de variation change trois fois).

Au départ la courbe est croissante, reflétant l'augmentation des gains marginaux due à l'absorption progressive des coûts de production initiaux. Dans la plupart des cas, le retour pour les initiateurs est alors minimal et il y a un problème d'initialisation du processus. Ensuite, la fonction de production devient linéaire, et le gain marginal à contribuer devient proche du coût marginal de cette contribution. Il devient alors rationnel de contribuer si

les autres le font (Oliver & al. [1985], pp. 533-534). Enfin, les retours marginaux diminuent quand on se rapproche des limites des besoins que le bien peut satisfaire. Chaque contribution est de moins en moins valorisée, d'autant plus que le nombre des contributeurs augmente. Il y a alors un risque de passager clandestin, un certain nombre d'agents cessant de participer à la production, ce qui risque de démotiver les autres. Les contributions, toujours nécessaires à l'existence du bien risquent alors de devenir insuffisantes[17]

Dans le cas du logiciel, on peut poser ce problème dans les mêmes termes que le fait Kollock [1999] : «getting individuals to contribute to the provision of a public good despite the temptation to free-ride. The decision not to contribute may spring from at least two sources - the desire to take advantage of someone else's efforts (greed), or an individual may be willing to cooperate but feel that there is not much of a chance that the good will be successfully provided and so does not want to waste his or her efforts (a concern with efficiency)».

La production de logiciel libre passe par ces trois étapes, ce sont les trois étapes que nous avons identifiées dans la première partie. C'est donc un système de production qui semble bien adapté aux spécificités du bien qu'il est sensé produire. Nous verrons dans la deuxième partie qu'effectivement, à chaque étape correspondent des motivations différentes pour les utilisateurs et que le rôle du noyau est à chaque fois de susciter et de canaliser ces motivations. Et justement l'ensemble de règles, de normes que nous avons relevé sont là pour faciliter ce rôle de coordination, à tel point que le Libre apparaît comme une organisation économique.

[17] Heckathorn ([1996], p. 273-274) résume ces trois étapes et leurs caractéristiques :

«Except in the smallest or most highly centralized systems, collective action is organized incrementally. During the *initial phase,* either «zealots» (Coleman 1990, p. 490) or a «critical mass» (Marwell and Oliver 1993) make the initial contributions. During the *intermediate phase*, the ranks of contributing actors continue to grow because of greater marginal returns to contributors, strategic interaction, or the operation of collective sanctions. Finally, during the *mature phase*, the limits of collective good production are approached because of physical constraints on further production of a dwinding pool of additional contributors. [...] The process of collective action also entails changes in the value of what may be termed the «local collective good» that each subgroup potentially produces. This is the portion of the global collective good that can be produced by the subgroup, so it reflects the marginal gains attainable through contributions by the local group's members. During the start-up phase, the value of the «local» collective good is low because initially the slope of the production function is almost flat. During the intermediate phase, marginal gains increase, and so too does the value of the local collective good. Finally, during the mature phase, the slope decreases, reflecting diminishing marginal returns, so the value of the local collective good declines. Thus, organization of collective action is characterized by an increase in the value of the local collective good followed by a decline.»

Le Libre, une organisation de production.

Ménard ([1990], p. 15), reprenant Robbins [1987], appelle organisation «une unité économique de coordination ayant des frontières identifiables et fonctionnant de façon relativement continue, en vue d'atteindre un objectif ou un ensemble d'objectifs partagé(s) par les membres participants.» Pour avoir une organisation, il faut alors, si on le suit, trois choses : un ensemble de participants, une entente (qui peut être implicite) sur des objectifs avec des moyens pour exprimer son accord ou s'en dissocier et une coordination formelle, définissant une structure, «caractérisée par son degré de complexité (la hiérarchie), par des règles et procédures (la formalisation) et par son degré de centralisation (la décision)» (ibid, p. 15).

La structure est là pour faire fonctionner les mécanismes nécessaires à la réalisation de ces objectifs : il faut choisir les objectifs, répartir les tâches à réaliser pour les atteindre, sélectionner les réalisations, récompenser les efforts, etc. Il s'agit de prendre en compte, de coordonner les motivations des agents appartenant à l'organisation, c'est-à-dire «l'ensemble des comportements qui conduisent un groupe d'agents à rapprocher leurs fonctions de préférence, de manière à établir un classement rendant compatibles les valeurs qu'ils attachent aux conséquences des actions à entreprendre» (ibid, p. 74). En même temps qu'un système de production, ou plutôt parce que c'est un système de production, l'organisation doit aussi être un système de transfert d'informations sur les objectifs des agents, sur leurs motivations, sur les informations et les outils dont ils ont besoin pour pouvoir remplir leurs tâches, etc.

Ce qui caractérise l'organisation est autant ses objectifs que les mécanismes qu'elle met en œuvre pour les réaliser c'est-à-dire pour assurer la production des informations sur les objectifs des individus et pour faire converger ces objectifs individuels (Barnard [1938]). Il faut soit que les agents aient tous les mêmes motivations, les mêmes objectifs, soit qu'il existe des incitations, c'est-à-dire un «ensemble [d']événements qui peuvent être manipulés par un décideur, ou une classe de décideurs, de manière à modifier les actes ou les conséquences des actes choisis par les autres agents» (Ménard [1990], p. 67 citant Dufy & Neuberger [1976] et Balassa [1986]).

Nous avons montré que la première caractéristique des projets libres est qu'il existe une structure d'autorité (le noyau de développement) chargée de coordonner les travaux de chacun, principalement en sélectionnant les contributions et en orientant l'évolution du projet (choix technologiques, organisation de la structure interne du logiciel, etc.)

Cette structure d'autorité se double presque toujours d'une structure hiérarchique : c'est le noyau qui sélectionne ses nouveaux membres, c'est le noyau qui accepte les contributions et désigne (au moins reconnaît) les responsables des sous-projets. Pourtant, plutôt que de hiérarchie, nous parlerons de régulation, parce que les mécanismes de sanction sont faibles (l'exclusion du groupe réclame un certain consensus chez les développeurs), les effets de réputation sont construits par les efforts des agents plus que par des choix du noyau. Le rôle d'autorité dévolu au noyau peut être contesté et il est toujours possible de changer cette autorité si une majorité du groupe en ressent le besoin. Le terme de «centralisme démocratique» évoque ainsi le fait que la «hiérarchie» assure son pouvoir de décision, son autorité, plus par ses actes et par la sélection des collaborateurs que par l'exercice d'un pouvoir contraignant sur les producteurs.

Il existe aussi des règles de fonctionnement chargées notamment de garantir l'autorité du noyau et d'assurer la convergence des objectifs (ce sont, par exemple, les normes de qualité et de définition que nous avons présentées au premier chapitre). Ces règles sont défendues par des institutions (la GPL et la FSF) et des autorités morales (comme Stallman ou Raymond) qui organisent le système de production et explicitent les «bons comportements» à avoir si l'on veut l'intégrer. Enfin, des éléments culturels (la culture des «hackers») renforcent le sentiment d'appartenir à un groupe, donc la cohésion du groupe.

On a bien là tous les éléments d'une organisation économique, à part un : nous n'avons pas encore montré que des agents pouvaient accepter de participer à cette organisation. À partir du moment où les agents acceptent les règles de fonctionnement du système, reconnaissent à la hiérarchie son pouvoir de décision, le système fonctionne. C'est finalement assez tautologique. Mais le fait que l'on ait analysé le fonctionnement et les règles de ce système va nous permettre d'identifier plus facilement les récompenses que les agents peuvent attendre de leurs contributions.

Aoki [1984a,b] a constaté, dans un autre type d'organisation, l'entreprise, l'existence d'une structure d'incitation en deux niveaux : le niveau de court terme où la participation prévaut sur les récompenses matérielles, gérée par l'ostracisme, la pression du groupe et le long terme où la motivation est celle de la promotion interne, ce qui implique un effort constant de formation et de coopération, de renouvellement des contributions. On retrouve la même chose dans l'organisation de production Libre. Si les récompenses immédiates et si les motivations stratégiques de court terme doivent assurer la stabilité à court terme de la structure de production et motiver les individus à entrer dans le processus de production, seules les récompenses accumulées garantissent la stabilité à long terme du système et la continuité de la production de chacun.

Nous nous intéresserons donc, dans la deuxième partie de ce chapitre, aux motivations qu'ont les producteurs à participer à chacune des étapes de la production d'un bien public comme le logiciel. Nous montrerons qu'on peut en trouver pour chaque catégorie de producteurs, marchands ou non-marchands, utilisateurs-producteurs ou simples producteurs. Nous verrons aussi que la diffusion des logiciels libres à l'ensemble de la population, et notamment aux utilisateurs naïfs nécessite qu'il existe des producteurs incités à adapter ces logiciels à leurs besoins.

Nous consacrerons la troisième partie de ce chapitre à l'étude de la stabilité dans le long terme de la relation utilisateurs-producteurs dans le cadre de l'organisation libre. Nous montrerons que les utilisateurs trouvent des producteurs qui acceptent de faire les adaptations dont ils ont besoin et que ces producteurs sont incités à les faire efficacement, c'est-à-dire sans diminuer la qualité des logiciels produits. Nous montrerons aussi que le travail des producteurs est suffisamment bien rémunéré pour être attractif et pour leur permettre de développer une activité rentable.

2.2 Une étude des motivations à participer au Libre.

Nous allons, dans cette partie, étudier successivement, dans les trois étapes de la production les motivations qu'ont les acteurs à participer et les conséquences que cela a sur les caractéristiques des logiciels développés.

2.2.1 L'étape 1 : initialisation du logiciel et phase d'accélération.

Cette phase initiale est sans doute la phase la plus difficile pour ces projets (comme pour tout projet de production de bien public, d'après Heckathorn [1996] et Marwell & Oliver [1993]). C'est en effet là que les retours dus à l'utilisation du bien public sont les plus faibles et les investissements à faire dans sa production les plus élevés. Nous allons montrer que les structures institutionnelles ne sont pas suffisantes pour créer un système d'incitation qui permette d'initier ces développements. Par contre, nous pourrons trouver des motivations chez les utilisateurs développeurs, marchands et non-marchands, à participer à ce développement. Ces motivations nous apparaîtront suffisantes pour expliquer la création de structure de production, basée sur des incitations non marchandes.

L'existence de structures institutionnelles : une condition nécessaire mais pas suffisante à l'implication des développeurs.

On constate que les premiers logiciels libres ont été initiés de deux façons : soit un individu avait un besoin non satisfait et était capable de le réaliser, c'est-à-dire qu'il disposait du temps et des compétences techniques pour le faire (Perl, Linux, Samba, Gnome), soit il s'agissait d'une demande d'un acteur public à des organismes de recherche (logiciels d'Internet, Ada).

Le financement public de logiciels libres.

Dans le second cas, un financement public permet de surmonter le défaut d'incitation qui caractérise cette première phase de développement. On est dans le cas classique du financement de la recherche et la question se pose de savoir si ces logiciels sont adaptés à des besoins autres que ceux de la recherche. Il existe des institutions spécifiques au libre, des fondations (l'Apache Fundation, la Free Software Fundation, etc.) qui jouent le double rôle de financier (ou au moins d'organisme de collecte et de redistribution des finances) et de coordinateur/annonceur de projets. On peut dire que ces institutions ont pris, pour une part au moins, le relais des institutions publiques classiques pour initier et coordonner le développement de certains logiciels. Mais elles sont sans doute moins efficaces pour soutenir un programme de recherche/développement similaire à ceux soutenus par les institutions

publiques et pour créer une équipe ex-nihilo qui s'impliquerait dans le projet. Surtout, beaucoup de projets libres, comme Linux ou Apache, n'ont pas été portés à l'origine par ces organisations.

L'initiative individuelle.

Le premier cas, proche du développement à façon, est celui qui initie le plus de projets. Se posent alors les problèmes de l'existence de motivations chez les initiateurs et de l'adhésion de développeurs à ce projet/produit émergent. Von Hippel ([1988], chapitre 4, pp. 85-88), a expliqué, de manière formelle, l'intérêt qu'il peut y avoir, pour un agent qui espère que les autres agents feront de même, à dévoiler une connaissance (c'est ce qu'il appelle le «know-how trading»). Il représente l'échange unitaire entre deux entreprises, ayant chacune une pièce de connaissance que l'autre ne possède pas (nous parlerons, nous, de pièce de logiciel, considérant, comme nous l'avons montré en introduction, que le logiciel est de la connaissance codifiée). Il s'agit, pour lui, d'un cas classique de dilemme du prisonnier, où chaque agent a intérêt individuellement à ne pas coopérer, mais espère que l'autre coopérera, la situation socialement optimale étant que les deux coopèrent.

On peut, à sa suite, esquisser une modélisation de ce problème, modélisation qui va nous servir pour analyser les motivations des différents acteurs qui interviennent dans la production de logiciels libres.

Une modélisation du dilemne publier/garder secret dans le cas du Libre.

Si l'on appelle t le gain de celui qui ne coopère pas quand l'autre coopère, r le gain si les deux coopèrent, p le gain si personne ne coopère et s le gain de celui qui coopère quand l'autre ne coopère pas, on a $t > r > p > s$ et $2r > t + s$. Il a été montré que, quand ce jeu se répète sur un horizon infini, ou quand l'agent joue ce jeu avec beaucoup d'acteurs, il se peut que la coopération s'installe (voir Axelrod [1984], par exemple). Von Hippel étudie alors dans quel cas on se trouve dans un dilemme du prisonnier, faisant l'hypothèse implicite que cette rencontre se fait dans le cadre dynamique d'un horizon infini. Il suppose, pour cela, que chaque unité de savoir/logiciel apporte un gain de R et que le fait de la posséder seule apporte un gain supplémentaire, ΔR. Ce qui nous donne, avec les notations du jeu $t = 2R + \Delta R$, $r = 2R$, $p = R + \Delta R$ et $s = R$.

Tableau 2.1 — Représentation sous forme de jeu du dilemme publier/garder secrète une connaissance.

	Ne pas publier	**Publier**
Ne pas publier	$p = R + \Delta R$ $p = R + \Delta R$	$s = R$ $t = 2R + \Delta R$
Publier	$t = 2R + \Delta R$ $s = R$	$r = 2R$ $r = 2R$

source : von Hippel [1988].

Il vient immédiatement que, quand $R > \Delta R$, il y a bien un dilemme du prisonnier et il est rationnel de coopérer tant que l'autre coopère si l'horizon est infini. Von Hippel souligne que c'est souvent le cas, car le gain à garder une pièce de connaissance (resp. de logiciel) secrète est assez faible. Le concurrent a souvent les moyens de développer lui-même cette pièce (la durée pendant laquelle on est en monopole est donc faible) et que les pièces, prises individuellement, sont rarement vitales pour l'entreprise, elles ne procurent qu'un avantage concurrentiel faible[18].

Développons ce point de vue dans le cas du logiciel libre. Nous remarquerons que, dans ce cas, l'agent ne joue pas contre une entreprise, mais «contre» un groupe, la communauté de développement. La deuxième différence provient du fait que toute production de logiciel libre est disponible. Il n'y a donc pas vraiment d'échange de savoir, ce qui rend plus facile les comportements de passager clandestin (Heckathorn [1996], Kollock [1999]). Pourtant, on peut encore parler de jeu, car, si l'on se place du point de vue de l'agent, on constate qu'il a le choix entre diffuser une pièce de connaissance ou la garder secrète, sachant que dans chaque cas la récompense de la «communauté» sera différente. C'est un jeu à horizon infini car on ne sait pas quand la production de logiciel sera terminée. Comme on se place du côté de l'agent, on peut abandonner les hypothèses restrictives sur le fait que le gain apporté par une unité de logiciel soit constant et égal à R.

Nous appellerons donc R l'utilité idiosyncrasique que l'agent peut attendre de sa contri-

[18]Il est curieux de noter que Johnson [2000] a développé un modèle qui se base sur une hypothèse a priori inverse pour expliquer la production de logiciel libre. Il se place dans le cas où les développements sont librement mis à disposition des autres développeurs. Il estime alors que les développeurs ont à arbitrer entre produire une pièce de logiciel ou attendre qu'elle soit produite par un autre développeur. Il ne la produisent que s'ils considèrent que la probabilité que quelqu'un d'autre le fasse est trop faible par rapport au besoin qu'ils en ont.

Cela lui permet d'étudier l'influence du nombre de développeur sur les motivations qu'a chaque individu à développer du logiciel. Ce modèle repose sur l'hypothèse que les individus n'ont pas le choix entre publier ou ne pas publier, hypothèse qui nous semble très forte et finalement peu réaliste. C'est pourquoi nous privilégierons l'approche de von Hippel.

bution et C le coût de production de cette contribution. $V = R - C$ représente alors la valorisation individuelle de cette contribution (ou plutôt la valorisation qu'il aurait s'il était seul, la valorisation «isolée»). Mais l'agent utilise un logiciel qui peut être utile à d'autres, et qui peuvent aussi le faire évoluer. Il a le choix entre divulguer, pour une valuation, peut-être négative, V_d, ou garder secret, pour une valuation V_m. Ces deux valuations peuvent être négatives, si le gain à divulguer (resp. à garder secret) est inférieur au coût de divulgation (resp. du secret). En effet, si on conçoit aisément que la divulgation, parce qu'elle engendre des coûts et favorise la concurrence, peut être coûteuse, il ne faut pas oublier que cela peut aussi coûter de protéger une invention, mais aussi de la maintenir compatible avec les nouvelles versions des logiciels, par exemple. Il n'est alors pas exclus que V_m et V_d soient négatifs. Une condition nécessaire et suffisante pour que l'agent produise effectivement sa contribution est alors que :

$$V + max(V_m, V_d) \geq 0 \,(1)$$

Une condition nécessaire et suffisante pour que l'agent divulgue sa contribution et qu'il la produise est que :

$$V_d \geq V_m \,(2)$$

.

La spécificité du logiciel fait que l'engagement d'un seul peut suffire pour développer un produit utilisable par d'autres et pour créer une dynamique autour de ce produit ; chaque motivation individuelle peut ainsi être à l'origine du développement d'un nouveau projet et cela explique en grande partie le développement initial de la dynamique libre. Cette argumentation est tout à fait valable dans le cas du libre et ce, d'autant plus que ces logiciels sont modulaires, qu'ils sont construits pour que chaque contribution marginale soit faible[19]. Il n'empêche que l'investissement initial reste coûteux, risqué et qu'on peut se poser la question de la récompense de l'effort que font les agents à publier leur travail pour qu'il soit lisible, efficace et facilement utilisable par d'autres.

Il faut, en effet, transformer ce qui s'apparente à du développement à façon en un projet

[19] On peut trouver le même type d'analyse dans Lakhani & von Hippel [2000].

de logiciel libre, utilisé par différentes personnes dans différentes organisations. Il est alors nécessaire de recruter parmi les utilisateurs designers ou sophistiqués de ce logiciel des producteurs (en nombre suffisant) qui soient capables de développer une première plate-forme utilisable et susceptible de créer de nouvelles demandes par adjonction de nouveaux modules. Le système de publication et de coordination des nouveaux projets est fondamental à ce stade car c'est lui qui va permettre de recruter ce noyau de développeurs et d'éviter le développement d'un trop grand nombre de projets concurrents (la ressource rare étant ici les utilisateurs développeurs capables d'initier un projet). Ces systèmes de publication existent, le plus connu étant sans doute «Source Forge» (http ://sourceforge.net). Et chaque projet dispose d'un système d'information (site Web, listes de diffusion, système de contrôle des versions, etc.) qui permet de coordonner les contributions.

Enfin, répétons ce que nous avons dit dans le premier chapitre : l'existence et la diffusion des réseaux informatiques comme Internet a grandement facilité de développement de ces systèmes de publication et de coordination.

Toutes ces «institutions», même si elles facilitent la coordination des hommes de bonne volonté, seraient inutiles si ces bonnes volontés n'existaient pas. Il faut que tous ces producteurs aient des motivations à contribuer au projet que propose la personne qui publie ses lignes de code sous forme libre.

Les motivations dans la phase initiale.

La principale motivation à publier des utilisateurs non-marchands : la valorisation des contributions que peuvent apporter les autres individus.

Lorsque des utilisateurs-développeurs produisent un logiciel pour leurs besoins propres, il est peu coûteux de le publier en libre et ils n'ont pas vraiment de gain à les conserver secrets (Lakhani & von Hippel [2000] le montrent dans le cas d'Apache) ; de nombreux projets de logiciel libre, à commencer par Linux ont débuté ainsi. La principale raison de la publication de son travail est alors qu'on valorise les retours des autres membres de l'organisation. À cause de cela, ces utilisateurs ont un $V_d \geq V_m$. Car s'ils dévoilent le code, au pire personne ne s'y intéressera et leur coût sera alors identique au fait de le laisser fermé et secret, majoré du coût de divulgation, qui est très faible. Les critiques, les remarques

des contributeurs ou des utilisateurs du produit permettent par contre de le faire évoluer. Il s'agit d'un retour sous forme d'information sur des besoins d'usages complémentaires au produit existant. L'ouverture permet aussi de faire produire certaines fonctionnalités par les utilisateurs qui en ont le plus besoin. Parce que le logiciel est une connaissance codifiée évolutive, l'ouverture permet l'apport par d'autres utilisateurs de nouvelles connaissances, d'idées qui sont utiles à l'initiateur du projet mais auxquelles il n'a pas pensé ou qu'il n'a pas le temps de développer. Laisser à d'autres le soin de développer des applications, c'est finalement leur laisser le coût de maintenance et de développement du logiciel, c'est-à-dire externaliser une grande partie du coût de développement.

Il est aussi (relativement) facile d'échanger des idées-logiciels, donc de co-développer un logiciel, une fois que l'on sait lire le langage dans lequel le logiciel a été programmé. On a vu que c'est la première chose qui a été standardisée en informatique, ce qui montre l'importance des rendements croissants dus aux effets d'apprentissage. La stabilité de la coopération est favorisée par le fait que l'apprentissage de ce langage, c'est-à-dire du système de codification des idées et du schéma d'algorithmes, est coûteux et que cet investissement n'est pas totalement récupérable[20]. Les participants ont donc intérêt au succès du groupe créé autour du langage et du projet de logiciel (Cowan & Foray [1997]). Ensuite, le logiciel produit étant directement utilisable par les membres du groupe, les améliorations profitent directement à l'ensemble des utilisateurs. Le retour sur investissement est plus rapide et plus directement appréciable[21], ce qui a certainement renforcé, dans le cas du logiciel, le sentiment d'appartenance à un groupe, à une «communauté» et donc, comme le montre De Bandt ([1998], p. 83), la propension à coopérer. Enfin, le problème de l'appropriation des fruits du développement ne se pose pas vraiment pour cette population car il n'y a pas à proprement parler, à ce stade, de rivalité entre les utilisateurs producteurs de logiciel. En effet, la programmation de systèmes complexes comme les systèmes d'exploitation, les compilateurs ou les grands programmes de calcul, fait appel à de nombreuses compétences, ce qui crée un besoin de coopération car un développeur seul ne les possède pas toutes.

[20] Même si, comme dans le cas des langues vivantes, il semble que le coût d'apprentissage d'un nouveau langage décroît avec le nombre de langages déjà maîtrisés.

[21] Raymond [1998] est un bon exemple de ce type de discours.

On retrouve les pratiques de l'organisation coopérative classique, proche de l'organisation scientifique. On peut se rappeler, pour s'en convaincre, que le logiciel étant en même temps expression et produit de l'idée d'algorithmique, une des bases de l'enseignement de l'informatique est l'étude des programmes déjà écrits, l'imitation et la réutilisation. Ces habitudes culturelles sont même devenues des techniques que le bon programmeur doit acquérir[22]. C'est ce que Kollock appelle «un effet de bord» de comportements privés. Ce n'est pas une motivation parce que la motivation à produire du logiciel n'est pas liée à l'organisation libre, celle-ci étant simplement utilisée comme support de publication de son travail.

La limite de cette «motivation» est qu'il reste néanmoins un coût à publier ce logiciel, à recruter une population de développeurs. Ce logiciel peut aussi être utile à des concurrents, quand on est impliqué dans une structure marchande. Autant cela ne pose pas de problèmes à des utilisateurs développeurs appartenant à des structures de la science ouverte (universités, centres de recherche), où il est normal de publier ses travaux, autant il peut y avoir un conflit entre les motivations de l'utilisateur-producteur et celles de son organisation.

La contribution des entreprises aux logiciels libres : une action des utilisateurs développeurs des entreprises.

L'utilisateur «en concurrence» peut être soit une entreprise, soit un développeur qui travaille dans une entreprise utilisatrice de logiciel et qui intègre les motivations de son employeur dans son arbitrage entre secret et publication en libre. C'est l'utilisateur que von Hippel [1988] a étudié dans son modèle. Il a un V_m plus élevé que l'utilisateur précédent, à compétence égale, car la divulgation de la contribution profite à ses concurrents (si ce n'est pas le cas, on est ramené au cas précédent). Il s'agit ici de comparer V_d et V_m, ou encore, en écrivant que $V = R + C$, de trouver le signe de $R_d - R_m - (C_d - C_m)$. R_m représente le gain de monopole, ce que von Hippel appelle ΔR.

L'élément le plus important en faveur de la divulgation est la dynamique de production du logiciel : plus le logiciel évolue rapidement, plus on a intérêt à divulguer son travail car

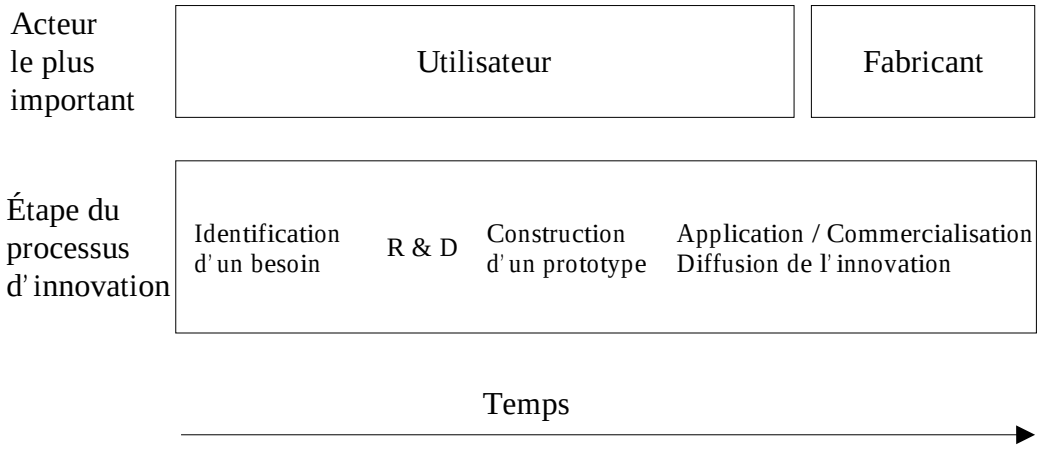
[22] «Good programmers know what to write. Great ones know what to rewrite (and reuse)» Raymonds [1998].

cette dynamique risque de le rendre obsolète (si les mêmes fonctionnalités sont développées par d'autres) ou incompatible avec les nouvelles versions du logiciel, ce qui augmente le coût de maintenance. A contrario, plus le logiciel développé est important dans la stratégie de l'entreprise, plus elle aura tendance à le garder secret. Ces utilisateurs-producteurs vont principalement contribuer au développement des logiciels évoluant vite, assez peu spécifiques à une profession (les contributions n'apportent pas d'informations sur les choix technologiques ou industriels de celui qui les fait), ou très hétérogènes (les contributions seront difficilement utilisables ailleurs que dans l'organisation qui les a proposées). Dans les deux cas, leur contribution est assez neutre du point de vue de l'avantage concurrentiel.

Considérant les caractéristiques du logiciel libre, il est finalement assez naturel qu'il se soit diffusé dans les entreprises. Les premiers logiciels libres à se diffuser, nous l'avons dit, ont été les logiciels supports d'Internet. Ils ont été amenés dans l'entreprise par des utilisateurs développeurs qui avaient suivi des enseignements universitaires[23]. Les premiers développements ont souvent été le fait d'initiatives individuelles, que la hiérarchie a validées ex-post, une fois que l'usage s'est répandu. Il était naturel pour ces personnes d'utiliser ces logiciels car ils étaient gratuits (et comme il s'agissait d'initiatives personnelles, il n'y avait pas obligatoirement de financements). Il était aussi naturel pour eux de contribuer à leur développement : issus pour la plupart de l'université, ils avaient été imprégnés de la culture du développement coopératif. Ces contributions ont continué quand, une fois un logiciel choisi par une entreprise, la maintenance a été confiée en interne à ces utilisateurs-designers. Il y a là une possible source de conflit d'intérêt entre un «agent», employé ou entreprise de service qui serait incité à divulguer pour renforcer sa renommée ou pour assurer une maintenance plus facile du logiciel et un «principal», entreprise, qui peut souhaiter que ces contributions restent internes, secrètes pour des raisons de concurrence. Le caractère modulaire des logiciels libres, parce qu'il diminue le coût de développement, mais aussi la portée des contributions, diminue ce risque de conflit et est certainement un facteur favorable à la production de logiciels libres par des utilisateurs en concurrence.

[23]Ce phénomène a été souligné par Langlois & Mowery [1996], qui notent que les universités ont toujours été, en informatique, un lieu de diffusion de l'innovation à cause de leur «relatively 'open' research and operating environment [and] their production of graduates who seek employment elsewhere.»

Figure 2.3 — Étapes observées dans le développement d'une innovation par l'utilisateur.



source : von Hippel [1988], p. 25.

En revanche, le fait que les développements spécifiques soient favorisés est, à première vue, un obstacle à la dynamique de production du libre qui, rappelons-le, est principalement incrémentale. Le risque est que de nombreux développements spécifiques apparaissent, sans qu'il y ait de cohérence entre ces développements. Ce risque doit cependant être nuancé au regard des travaux de von Hippel [1986] sur les utilisateurs innovants, les utilisateurs «précoces» («lead users»). Il définit ces utilisateurs de la façon suivante :

«- lead users face needs that will be general in a marketplace-but face them months of years before the bulk of that marketplace encounters them, and

- lead users are positioned to benefit significantly by obtaining a solution to those needs.» (von Hippel [1986], p. 796).

Il note, en s'appuyant sur les travaux de Rogers & Shoemaker [1971] que ces utilisateurs précoces existent parce que «important new technologies, products, tastes, and other factors related to new product opportunities typically diffuse through a society, often over many years, rather than impact all member simultaneously» (ibid p. 796). Cela est confirmé empiriquement par Mansfield [1968] qui montre que, sur 12 innovations majeures dans les industries lourdes, pour 3/4 d'entre elles, il a fallu plus de 20 ans pour qu'elles soient diffusées dans toutes les entreprises du secteur.

Les utilisateurs sont souvent à l'origine de l'innovation car ce sont eux qui sont les

mieux à même d'identifier un besoin et de construire un prototype autour de ce besoin. Et ils la diffusent car les coûts de production de cette innovation dépassent l'avantage concurrentiel qu'ils retirent à la garder cachée. Si des utilisateurs développent un logiciel relativement spécifique, ils ont intérêt à le publier pour qu'il soit incorporé à un projet plus vaste, que ses capacités soient étendues en même temps qu'est transféré le coût de sa maintenance (on retrouve une des deux façons dont est initié un projet libre).

Le premier groupe de producteurs : une collaboration basée sur des échanges non-marchands.

Ainsi, l'innovation dans le logiciel libre vient principalement d'utilisateurs qui publient des logiciels en espérant que d'autres utilisateurs auront les mêmes besoins et accepteront de partager les coûts de développement de ces besoins. Ousterhout ([1999], p. 44) résume bien ce processus : «most open-source projects start out with a single programmer solving a personal problem and making the solution available to others. The early adopters of an open-source package tend to be like the creator : sophisticated programmers with a particular problem that the open-source package solve.»

La production de logiciel est d'abord la constitution d'un groupe, d'une organisation, qui doit construire de nouvelles connaissances en même temps qu'organiser la mise en circulation des «meilleures pratiques» entre des agents hétérogènes. La collaboration est assurée par le besoin d'accéder à des savoirs complémentaires, mais aussi par le respect d'une autorité procédurale (collaboration imposée par les institutions), donc par un mélange de recrutement par les pairs (au niveau de chaque institution, université ou entreprises) et de confiance mutuelle (dans les projets collaboratifs entre institutions).

Cette habitude coopérative est efficace dans un milieu regroupant des agents, appartenant ou n'appartenant pas à des organisations marchandes, mais culturellement proches, ayant des compétences à échanger et des besoins communs. Cette organisation de production a naturellement privilégié le développement des logiciels réclamant de telles compétences c'est-à-dire des logiciels reposant sur une forte dynamique de la connaissance et nécessitant une implication des utilisateurs pour (co-)développer les technologies d'utilisation. D'où aussi le fait que sont favorisés les logiciels qui engendrent de fortes externalités

de réseau et qui sont le plus soumis à des interrelations technologiques : ce sont ceux pour lesquels les besoins d'ouverture et de contrôle du code sont les plus forts.

On peut dire, en s'inspirant de la classification de Cohendet & al. [2000], que cette organisation initiale est un mélange de réseau, de communauté de pratiques et de communauté épistémique[24]. Dans ce type de population, relativement petite, la coopération peut fonctionner car les acteurs se connaissent, ont chacun des talents qui les singularisent et les échanges peuvent se faire sur la base du don-contre don (Olson [1965], Kollock [1998], Smith & Kollock [1999], Lakhani & von Hipple [2000], Dang Nguyen & Pénard [2000], [2001]). C'est la «solidarité organique» de Durkheim.

Signalons un fait qui renforce la stabilité de la coopération : Rheingold [1993], Wellman & Gulia [1997] (que Lakhani & von Hippel [2000] illustrent dans le cas d'Apache), ont montré que les individus qui participent, offrent régulièrement des informations, bref qui sont connus pour être de «bons citoyens» dans l'organisation, reçoivent plus rapidement de l'aide. La coopération engendre la coopération, surtout si le groupe est relativement stable (on se rencontre plus souvent). Plus que de créer la motivation initiale à contribuer, cela encourage à renouveler ses contributions. On encourage ainsi une dynamique vertueuse entre contributeurs qui s'auto-alimente et qui peut avoir aussi comme effet d'exclure de nouveaux entrants qui ne verraient pas leurs contributions prises en compte[25].

C'est bien un système où l'innovateur est l'utilisateur, mais contrairement aux industries étudiées par von Hippel, les entreprises de production de logiciel ne prennent pas tout de suite le relais des utilisateurs. Car, à la différence d'autres secteurs industriels, le développement d'un prototype par un utilisateur est aussi le développement du produit ; contrairement à la production de biens tangibles où il faut adapter le prototype à un pro-

[24]Nous avons choisi de traduire le terme «Practice» par «pratique». Du réseau, nous retenons la négociation des spécialisations (médiatisé ici par des institutions) et l'hétérogénéité des agents qui ont un besoin d'accéder à des connaissances complémentaires. De la communauté de pratique, le partage d'un même langage et la construction de ce langage (ainsi que la construction de la «culture» afférente). De la communauté épistémique, le recrutement par les pairs (au moins pour la partie universitaire et à l'intérieur des entreprises).

Le logiciel libre n'«entre pas bien dans les cases» car cette typologie est destinée à étudier les groupes de production de connaissances qui se forment à l'intérieur d'une entreprise. Ici, nous analysons des coopérations entre individus issus de différentes institutions (entreprises, centres de recherche, agences fédérales, etc.)

[25]Ce point est souligné par Van Alstyne & Brynjolfsson [1997] qui remarquent que les «communautés» en ligne pouvaient soit intégrer des individus d'opinions assez différentes, si on valorisait suffisamment la variété, soit, au contraire, se balkaniser autour d'intérêts de plus en plus spécifiques, pour garantir le niveau des échanges entre les membres.

cessus de production industriel, il n'y a pas ici de production. Le logiciel passe alors bien plus facilement du stade de prototype au stade de produit. Le facteur limitant la contribution de cet utilisateur est donc le coût de production et de divulgation de la contribution. Tout ceci a une implication directe sur le type de logiciels développés.

Les conséquences de ces motivations sur l'implication des utilisateurs et sur les caractéristiques des logiciels produits.

C'est parce que des agents ont des connaissances hétérogènes qu'il est moins coûteux pour certains de développer ou d'adapter des logiciels que d'utiliser les logiciels standards existants. Mais c'est aussi à cause de cette hétérogénéité que beaucoup d'utilisateurs sont a priori exclus de l'utilisation du libre. Car seuls les utilisateurs capables de développer des logiciels, utilisateurs designers ou sophistiqués, sont en mesure d'utiliser directement l'organisation Libre pour faire prendre en compte leurs besoins.

Des contributions limitées aux utilisateurs développeurs.

Les contributions seront toujours limitées aux utilisateurs connaissant suffisamment le logiciel pour pouvoir le comprendre, contribuer (c'est-à-dire programmer réellement une extension ou une correction, ce qui nécessite du temps) et faire accepter cette extension ou cette correction. Cela a deux conséquences immédiates, l'une favorable à la stabilité de l'organisation de production, l'autre défavorable à la diffusion de logiciels libres.

Il nous semble que le risque est très faible de voir augmenter fortement la population des contributeurs, en même temps que les logiciels libres deviennent populaires. Les utilisateurs capables de contribuer seront toujours peu nombreux, toujours issus des mêmes environnements culturels pour leur grande majorité (les universités, les centres de recherche en informatique), ce qui facilite la stabilité de la coopération car elle peut toujours être basée sur la réciprocité et la réputation entre des agents culturellement proches[26]. C'est heureux car cette augmentation rendrait difficile la gestion d'une population devenue trop nombreuse et surtout pourrait entraîner une baisse de la qualité moyenne des contributions.

[26] «In addition, each of the feature that encourage reciprocity - ongoing interaction, identity persistence, knowledge of previous interactions, and strong group boundaries - would also work to promote the creation and importance of reputations within an online community.» Kollock [1999].

Est-ce à dire que la production de logiciel risque de rester marginale en volume ? Nous avons vu qu'elle avait déjà atteint un niveau assez important, mais il est certain que, si elle se limite aux contributions d'utilisateurs-développeurs non marchands, le logiciel libre ne répondra qu'aux besoins d'une petite partie de la population. Il y a une nécessité qu'une autre population de développeurs prenne en compte les besoins des utilisateurs naïfs, plus généralement des utilisateurs qui ne peuvent pas mobiliser les compétences suffisantes pour développer les technologies d'utilisation susceptibles de répondre à leurs besoins. Si les nouveaux utilisateurs ne peuvent pas développer ces services parce qu'ils n'en ont pas les capacités, si le noyau de développement n'est pas intéressé par le développement de ces nouvelles fonctionnalités, il y a un réel risque que le public du logiciel soit confiné au noyau de développeurs originel, que le passage de l'étape 1 à l'étape 2 de la production ne se fasse pas. Le logiciel libre sera alors une organisation efficace mais limitée à la production d'objets techniques pour des techniciens.
Il y a nécessité d'une autre source de motivation que celle de l'utilité directe pour que l'utilisateur-développeur prenne en compte les besoins des autres utilisateurs.

L'utilisateur-développeur ne peut pas tout.

Il faut distinguer deux raisons pour lesquelles l'utilisateur-développeur ne peut pas tout : soit le développement d'une fonctionnalité ne l'intéresse pas, ne lui est pas utile, soit il n'a pas les capacités pour le faire. Dans le premier cas, il va falloir l'intéresser, l'encourager, le motiver à le faire, dans le second, il va falloir trouver d'autres développeurs capables de le faire.

On ne peut pas dire que la production de logiciel libre soit limitée à des productions utiles uniquement aux utilisateurs développeurs. Certains utilisateurs «non développeurs» sont prêts à financer le développement de logiciels, en libre, pour leurs besoins propres. La première raison qui vient à l'esprit est que ce sont des institutions publiques, financées par des fonds publics, qui ont pour tradition, quand ce n'est pas pour obligation, de rendre publics les travaux qu'elles financent. Ceci est vrai dans le cas des travaux des universités américaines, donc pour la production des logiciels d'Internet, comme Sendmail, Bind ou Apache. Mais pas dans le cas du langage Ada 95. Dans ce cas, la raison principale était que le client, le ministère de la défense américain, voulait la création d'un standard (il

y avait, à l'époque des centaines de langages différents utilisés pour développer les logiciels de défense), et il voulait être sûr que le logiciel développé ne serait pas approprié par une entreprise qui contrôlerait l'innovation et qui serait en situation de monopole par rapport à ce langage[27]. Cependant ces initiatives restent marginales, et l'exemple d'Ada ne concerne pas une entreprise en situation concurrentielle. On peut considérer que le ministère de la défense américain est un utilisateur-développeur qui a préféré sous-traiter le développement d'un logiciel plutôt que de le produire en interne, pour des raisons d'efficacité. Cet utilisateur avait donc les connaissances nécessaires pour établir des spécifications suffisamment précises de ses besoins et aussi pour sélectionner son fournisseur.

Se pose la question du développement des logiciels qui ne concernent que les utilisateurs naïfs, qui n'ont pas ces capacités de choix et de spécification, ni même, lorsqu'il s'agit d'utilisateurs individuels, les capacités financières pour initier le développement. Ces logiciels risquent de ne pas franchir l'étape 1, la constitution du noyau de développeurs. Mais le problème est le même pour l'ensemble des logiciels qui, bien qu'ayant une base technique, doivent être adaptés pour être utilisables par les utilisateurs sophistiqués ou les utilisateurs naïfs. S'ils franchissent la première étape parce qu'il y a une volonté forte de l'organisation de produire ces logiciels (interfaces graphiques, suite bureautique), ils risquent d'être inadaptés aux besoins de ces utilisateurs naïfs qui ne peuvent pas les exprimer à travers les canaux d'information d'une organisation qu'ils ne maîtrisent pas.

Nous avons dit que les logiciels les plus à même d'être développés en libre par les utilisateurs développeurs étaient les outils techniques évolutifs et les outils soumis à de fortes externalités car c'est pour ces logiciels qu'il y a le plus grand besoin de coordination donc de standardisation. Parmi cette deuxième catégorie, nombreux sont ceux, à commencer par le système d'exploitation et par les compilateurs, qui sont utilisés par des utilisateurs sophistiqués ou naïfs. Pour tous les logiciels libres soumis à de fortes interrelations technologiques (c'est-à-dire qui dépendent étroitement d'autres composants pour fonctionner) et fonctionnant en réseau, la conquête de ces utilisateurs est indispensable pour pouvoir devenir le standard. Il est donc important de comprendre comment un logiciel libre peut dépasser le stade de l'objet technique.

[27] Voir la présentation de Gasperoni [2001], un des créateurs du langage.

Quelles motivations pour dépasser le stade de l'objet technique ? Le passage de l'étape 1 à l'étape 2.

Le passage de la phase 1 à la phase 2 de la production est aussi un moment délicat car il s'agit de transformer un projet spécifique destiné à une population réduite en un projet plus vaste, capable de répondre à des demandes diverses, qu'il faut rendre cohérentes et compatibles. Nous allons d'abord voir que les motivations peuvent être trouvées dans le groupe initial ; il ne s'agit pas de motivations directes pour celui-ci, mais elles peuvent être suffisamment fortes pour lui faire prendre en compte les besoins d'utilisateurs proches par les besoins (interface graphique, suite bureautique, etc.) ou par la fréquence des relations d'échange (s'ils sont dans la même organisation, par exemple). Mais nous allons montrer que les motivations vous surtout se trouver chez les utilisateurs marchands, les entreprises.

Les motivations internes au groupe initial.

Nous allons reprendre l'étude des valorisations possibles des contributions et de la renommée, en regardant en quoi ces valorisations peuvent pousser des développeurs à produire des logiciels qui ne leur seraient pas directement utiles. Évidemment, ces motivations peuvent aussi renforcer leurs motivations à produire des logiciels qui leurs sont directement utiles.

Une raison secondaire importante qui peut pousser un agent à produire du logiciel est que cela renforce la considération qu'il a pour lui-même et notamment l'idée qu'il se fait de son efficacité (Bandura [1995], Constant & al. [1994]) et de son influence sur le monde. D'une certaine manière, on peut dire que l'engagement de Stallman, en 1984, tenait un peu de ce type de motivation, comme celui de Icaza, dans le démarrage du projet Gnome[28]. Dans le même ordre d'idée, on peut produire du logiciel en libre parce que c'est une démarche en accord avec certains de ses idéaux : si Richard Stallman a fondé la FSF, c'est aussi parce qu'il pense que le logiciel est une création intellectuelle et que, comme telle, elle doit être accessible librement. Cette motivation est, finalement, assez proche de ce que Kollock appelle l'attachement à un groupe. Elle représente l'engagement que l'individu est

[28] Voir l'histoire du lancement du projet, http ://primates.helixcode.com/~miguel/gnome-history.html.

prêt à prendre pour ce groupe. Là encore, il s'agit de la construction de l'identité d'une personne, qui est définie par sa relation aux autres (Pizzorno). La contribution, ou plutôt le fait que cette contribution soit acceptée par le groupe est un signe de reconnaissance du groupe envers un de ses membres. Chez Stallman, il s'agissait de défendre une certaine idée de la production de logiciel et de la culture de coopération qui s'était développée dans les centres de recherche en informatique. Si elle est suffisamment forte, cette motivation pousse les développeurs à prendre en considération les besoins des utilisateurs hors du groupe initial, pour assurer la diffusion des logiciels hors de celui-ci.

La réputation fonctionne de la même façon, puisque la personne devient une figure reconnue d'un groupe dans les valeurs duquel elle se reconnaît. La réputation est justement fondée sur la répétition des signes de reconnaissance de la part du groupe. Et cette répétition est fondée sur la répétition des contributions, qui sont autant de signaux sur sa connaissance du programme et sur sa capacité à l'améliorer ou à le mettre en œuvre.

Ces récompenses sont immédiates lorsqu'il s'agit de la satisfaction que l'on retire à être reconnu par le groupe et à voir le projet avancer ; elles sont à plus long terme (ou plutôt elles demandent un effort continu et d'une certaine durée temporelle) lorsqu'il s'agit d'effets de reconnaissance du groupe, base de la construction de la renommée. Dans tous les cas, tant qu'on se situe dans les phases 1 et 2 de la diffusion du logiciel, la motivation à contribuer va augmenter avec la taille du groupe car les nouveaux utilisateurs amènent de nouveaux besoins auxquels il faut répondre pour les convertir. Mais nous allons voir que la réputation permet d'autres valorisations que l'estime de soi.

Le premier intérêt de la renommée, la valorisation la plus immédiatement visible, est qu'elle permet l'accès à des postes de «responsabilité» dans la conduite des projets libres ou en tant que porte-parole de cette communauté. Le meilleur exemple en est sans doute celui de Raymond, qui parce qu'il apparaît comme un gardien des bonnes pratiques du libre a été engagé par la société VA Linux pour faire partie de son conseil de surveillance. C'est ce qui se rapproche le plus du système d'incitation mis en place dans les entreprises, où les agents peuvent progresser dans la hiérarchie. Il est certain que plus le projet est connu, plus le logiciel est utilisé, plus la réputation du chef de projet est grande. Cette

motivation, qui est à rapprocher d'un besoin de reconnaissance individuelle, peut être une motivation secondaire pour diffuser le logiciel hors du public initial d'utilisateurs développeurs. Cependant, la reconnaissance vient surtout des pairs et, comme le montrent Foray & Zimmermann [2001], il est fort probable que cette diffusion n'apporte que peu de reconnaissances supplémentaires. Leur conclusion, que nous ferons nôtre, est que d'autres sources d'incitations sont nécessaires pour assurer l'adaptation des logiciels libres aux besoins des utilisateurs naïfs.

Lerner & Tirole [2000] interprètent ces effets de réputation comme des signaux sur la qualité de l'individu qui acquiert la réputation. Il s'agirait d'un signal en direction de possibles recruteurs sur la qualité de leur travail. Là encore, il est plus facilement valorisable d'avoir contribué à imposer un logiciel comme une référence dans son domaine en ayant su prendre en compte des besoins autres que ceux des utilisateurs développeurs. Il nous semble que cela est surtout vrai si l'on se place dans une perspective industrielle, où le signal est envoyé aux clients qui cherchent à utiliser les connaissances que l'individu ou l'entreprise ont développées sur le logiciel (nous y reviendrons quand nous étudierons l'implication des entreprises productrices de logiciel).

Mais c'est sans doute dans ce passage de l'étape 1 à l'étape 2 que le défaut de motivation des utilisateurs-développeurs est le plus important. C'est dans cette phase que les organisations utilisatrices ont le rôle le plus important, car c'est dans les organisations que peuvent être confrontés les besoins des différents types d'utilisateurs. Ce sont ces organisations qui vont financer les fondations (comme la F.S.F. ou l'Apache Fundation) ou qui vont payer des entreprises qui développent et adaptent les logiciels libres. Elles vont alors créer une nouvelle source de motivations : les développeurs vont être payés pour produire les logiciels qui ne les intéressent pas directement (Dalle & Jullien [2000], [2001], Foray & Zimmermann [2001]).

L'organisation, l'entreprise, l'élément indispensable à la diffusion du logiciel libre.

Pour qu'un logiciel soit adopté par une organisation, il faut justement le diffuser, l'adapter à des besoins qui peuvent être légèrement hétérogènes, faciliter son utilisation,

développer des applications complémentaires, etc. parce que : «as an open-source package becomes more and more popular, a widening gap develops between the needs of the user community and the features provided by the core developers» (Ousterhout [1999], p. 44). Les organisations peuvent répertorier les besoins des personnes travaillant en leur sein et payer des développeurs pour adapter, en interne, les logiciels dont elles ont besoin (elles le font déjà : faut-il rappeler que la plupart des logiciels sont toujours actuellement produits en interne, pour les besoins propres des entreprises). À condition, bien sûr, que la mise en libre soit avantageuse pour ces entreprises.

On a vu dans la première partie que la clef de ce passage, de la «dé-balkanisation» du projet, pour reprendre la terminologie de Van Alstyne & Brynjolfsson [1997], résidait dans la structure organisationnelle de production. Il faut que le projet repose sur une base standard, ouverte et stable, support pour les modules qui permettent de prendre en compte des demandes plus spécifiques et plus hétérogènes. Si cette structure technique existe, si le «management» de la structure d'autorité permet facilement ces contributions aux marges, elles se font sur des petites parties du logiciel, à un coût marginal faible, ce qui est favorable à sa mise en libre par les entreprises. Ceci est d'autant plus vrai que ce sont des logiciels de réseaux ou des logiciels à la base des systèmes d'information puisque ce sont eux qui assurent l'interopérabilité, c'était la conclusion du chapitre 1. On comprend que ces utilisateurs soient prêts à soutenir une solution ouverte, si le déploiement de cette solution est financièrement acceptable (et que notamment les coûts de changement ne sont pas trop importants). Au-delà des aspects techniques, les logiciels libres à la base d'Internet, parce qu'ils étaient la solution technique la plus éprouvée dans ce marché, ont permis aux entreprises de découvrir les logiciels libres, de vérifier leur efficacité. Aujourd'hui, elles peuvent étendre leur utilisation des logiciels libres à des fonctionnalités proches de ce positionnement original (serveur Web puis serveur de fichier, puis serveur de données et poste client de consultation de ces données avant, peut-être, le poste client).

Le risque est qu'elles utilisent les logiciels libres pour développer des solutions internes spécifiques, perdant ainsi le bénéfice des économies d'échelle en production (mutualisation des efforts, des connaissances) qui est le grand intérêt des logiciels «sur étagère», des progiciels. C'est encore un argument pour la mise en libre des contributions de ces grandes

entreprises qui s'assurent ainsi de l'interopérabilité de leurs systèmes dans le temps. Une dynamique d'innovation importante renforce aussi l'intérêt pour les entreprises de publier leurs développements, pour maintenir la compatibilité de ces développements avec le noyau du programme et externaliser cette maintenance. On retrouve les mêmes caractéristiques que dans la phase 1 : les logiciels libres susceptibles d'attirer le plus de contributions sont les logiciels qui sont les plus diffusés, qui sont des clefs pour assurer l'interopérabilité et dont les usages évoluent vite.

Les entreprises sont aussi le lieu de rencontre entre des utilisateurs de niveaux différents mais aussi des organisations qui peuvent coordonner les efforts et investir dans le développement de logiciel. C'est d'elles que peut venir la contribution supplémentaire qui fait passer les logiciels libres (au moins ceux sur lesquels les rendements croissants sont forts) du statut d'objet technique au statut de logiciel potentiellement utilisable par tous.

À partir du moment où ce passage au niveau 2 a eu lieu, comme c'est le cas pour Apache ou Linux, on est dans la phase où il devient plus intéressant de contribuer à ces logiciels que de rester à l'écart. C'est à ce niveau qu'interviennent les entreprises commerciales, qui vont avoir comme fonction d'agréger les demandes individuelles et d'y répondre en agrégeant les contributions financières de chacun de ces utilisateurs, en plus d'adapter les logiciels pour les grands comptes qui préfèrent faire-faire plutôt que faire. Car on retrouve dans les contributions des entreprises clientes les mêmes faiblesses dans la prise en compte des besoins des utilisateurs naïfs isolés : la participation active au développement, voire même la spécification des contributions ne concernent que les utilisateurs designers et, dans une moindre mesure, les utilisateurs sophistiqués. Les utilisateurs qui ne possèdent pas les compétences techniques suffisantes ne sont pris en compte qu'indirectement, s'ils sont dans des entreprises qui payent pour réaliser les contributions qui répondent à leurs besoins[29]. Mais ces entreprises vivent de la production et de la vente des logiciels ; il semble paradoxal qu'elles puissent produire du logiciel libre. Il nous semble pourtant que, pour certains logiciels, cette production se justifie.

[29] «As open-source package becomes more and more popular, a widening gap develops between the needs of the user community and the features provided by the core developers. Eventually, commercial ventures will form to fill this gap». (Outerhout [1999], p. 44.)

2.2.2 L'étape 2 de la production.

Si l'on suit Genthon & Phan ([1999], p. 178), on peut distinguer trois types d'intervenants dans la spécification et la production de logiciels : les producteurs, c'est-à-dire les agents dont une des activités principales est la «spécification, la conception, le codage, l'intégration et la maintenance», les utilisateurs, qui produisent ou font produire par des sociétés de service du logiciel, avant tout pour leurs besoins propres, les «diffuseurs» du produit, les éditeurs, qui ont une activité de «packaging» (duplication sur supports divers, édition de manuels, etc.) et les distributeurs (mise à disposition du produit pour les utilisateurs)[30].

Lorsque l'étape 2 est atteinte dans la production du bien public, les producteurs ont davantage intérêt à participer à la production qu'à rester à l'écart (Marwell & Oliver [1993]). Nous allons voir que c'est effectivement le cas dans la production du logiciel libre et que c'est à ce stade que l'on trouve actuellement le plus de contributions de la part des producteurs. Comme pour les utilisateurs, les motivations des producteurs dépendent, au moins en partie, de leurs caractéristiques et notamment de la façon dont le logiciel est utilisé dans leur offre commerciale. Nous tenterons d'évaluer, pour chacun, l'utilité de la divulgation, vis-à-vis de l'utilité de la non divulgation d'un logiciel. Puisqu'il s'agit d'étudier le fonctionnement de l'organisation économique Libre, nous nous placerons dans le cas où les producteurs et vendeurs de logiciels doivent publier le code qu'ils produisent sous une licence libre (typiquement la GPL), c'est-à-dire qu'ils renoncent à la propriété exclusive du logiciel, en autorisant leurs clients à le redistribuer librement.

Le métier de producteurs de logiciel.

Une entreprise peut espérer différentes formes de valorisations financières de son implication dans la production de logiciel libre, formes que nous avons regroupées dans le tableau 2.2.

Nous parlons en termes uniquement monétaires car nous considérons que si l'entreprise

[30] La distribution des logiciels dépend de la façon dont ils sont vendus : s'ils sont vendus par des éditeurs, alors ils pourront être distribués en tant que tels dans des rayons spécialisés ; s'ils sont intégrés dans des machines (comme c'est le cas pour le système d'exploitation), ils seront distribués par les distributeurs des machines.

Tableau 2.2 — Valorisations espérées par un producteur de technologies d'utilisation de contributions à des logiciels libres.

Localisation dans la fonction de profit	**Type d'impact**	**Explication du mécanisme**
Impact sur les coûts de production	- maintenance & amélioration de la contribution prise en charge par d'autres.	
	- possibilité d'améliorer la contribution en tenant compte des commentaires des utilisateurs.	
	- signal en direction de recrutables :	fait connaître le niveau de l'entreprise, la qualité technique des groupes de programmeurs.
	- plus de facilité pour recruter une communauté d'utilisateurs développeurs :	conséquence de la renommée. N'a un impact que si l'entreprise cherche à développer d'autres projets de libre.
Impact sur les recettes	- signal pour les clients	baisse des asymétries d'information, construction d'une marque.
	- extension de capacités du programme :	permet d'élargir la taille potentielle du marché en intéressant de nouveaux utilisateurs au logiciel.
	- orientation du logiciel vers ses besoins :	permet de mieux répondre aux demandes.
Impact sur les recettes et les coûts de production	- augmentation de l'expertise technique générale et de la connaissance du logiciel en particulier :	effet de réputation (asymétrie d'information) et meilleure efficacité pour répondre à des problèmes posés par des clients ou pour proposer de nouvelles extensions.
	- création d'un bien public industriel :	création d'un concurrent au logiciel établi ou création d'un standard à la profession.

ne peut pas attendre de retour financier de cette activité, coûteuse, elle n'a aucune raison de le faire. Une entreprise est une organisation dont le but est de dégager du profit et même si ce retour est difficilement évaluable, il faut qu'il soit suffisamment attractif pour motiver l'organisation à produire du logiciel libre.

On ne doit plus comparer V_d et V_m car V_m n'existe pas. Il s'agit ici d'étudier quand il est économiquement intéressant pour une entreprise de produire du logiciel libre, sachant

qu'elle ne pourra pas se faire rémunérer pour la production de ce logiciel. La proposition (1) devient alors $V + V_d \geq 0$, avec $V = -C$, le coût de production. On doit étudier les situations pour lesquelles les retombées financières de long terme espérées peuvent couvrir le coût de développement d'une contribution à un logiciel. Nous pouvons aller plus loin : comme nous sommes en présence de logiciels libres, on peut même dire que ces producteurs renoncent aux revenus qu'ils pourraient tirer de la vente de leur production, en même temps qu'ils renoncent à certaines prérogatives de leur droit de propriété. Cela veut dire que la contribution au logiciel ne remet pas en cause ces sources de revenu. Soit elles proviennent d'un autre logiciel si on s'intéresse à un producteur «pur» de logiciel, soit elles proviennent de services pour une entreprise dont l'activité de production de logiciel est une activité annexe.

Pour comprendre la stratégie de ces entreprises, il faut l'analyser dans le contexte de la production et de la vente des technologies d'utilisation, en tenant compte des liens qui existent entre les différents produits qui composent ces technologies d'utilisation. Le métier d'offreur de logiciel nécessite la maîtrise de trois types de compétences : il faut connaître les besoins de ses clients/utilisateurs, il faut être capable de traduire ces besoins en terme de technologies d'utilisation, c'est-à-dire maîtriser les technologies propres à ces utilisateurs et enfin savoir (avoir les compétences informatiques pour) programmer le logiciel.

Nous allons commencer par les producteurs avant d'étudier les diffuseurs.

L'implication des producteurs : des stratégies opportunistes.

La plupart des producteurs qui contribuent à la production de logiciel ne sont pas à l'origine du logiciel libre. C'est le cas de Corel avec Wine, de SUN avec Apache, d'IBM avec Linux. Leurs contributions s'expliquent par le fait que le logiciel libre existe, qu'il est devenu un produit utilisé par les clients de ces entreprises, quand ce n'est pas le standard : il existe alors des raisons stratégiques qui font qu'il devient indispensable de contribuer.

Renforcer un standard dans une concurrence entre standards.

Le premier objectif stratégique poursuivi par les entreprises productrices de logiciel peut être de renforcer un standard dans une concurrence entre standard. Gérard Dréan ([1996],

p. 230) note que les deux moteurs concurrentiels du secteur des progiciels sont, «comme pour les composants, la rivalité entre le leader de chaque sous-secteur et ses challengers» et «l'élargissement continu du champ d'application de l'informatique, qui fait apparaître de nouveaux domaines [...] où tout nouvel entrant peut espérer avoir sa chance de devenir leader à condition d'être le premier à y entrer». Les challengers peuvent avoir intérêt à subventionner un standard concurrent au standard dominant. Surtout, les entreprises qui dépendent d'un standard pour réaliser leur activité (comme SUN avec Apache) ont intérêt à ce que ce standard soit le plus ouvert possible, pour ne pas dépendre de la stratégie de l'entreprise qui fournit le standard.

Là encore, il faut souligner que les logiciels libres susceptibles de recevoir des contributions visant à créer une concurrence entre standards et à favoriser des standards ouverts sont les logiciels avec d'importants effets de rendements croissants. On remarquera que le système de concurrence est alors proche de celui qui s'est établi dans la partie matérielle de la production de PC : un nouveau standard est accepté si tous les concurrents peuvent en disposer, donc s'il est rendu public. Si ce n'est pas le cas, si l'innovation technologique est utilisée pour créer un avantage concurrentiel monopolistique, les concurrents soutiendront un autre standard. Dans ce cas d'une concurrence entre standards, certains auteurs (Genthon & Phan [1999], Lerner & Tirole [2000]) s'inquiètent du fait que la participation des entreprises soit limitée au temps où elles trouveront un intérêt stratégique à affaiblir le monopole concurrent. Si on suppose qu'elles peuvent réussir à affaiblir ce monopole, l'expérience de l'industrie des composants nous amène à penser que tant que la dynamique d'innovation est forte, ce risque est faible. Car même quand il existe un standard, il faut continuer à contribuer pour orienter le standard vers ses besoins.

Orienter le standard vers ses besoins.

Le meilleur exemple de ce type de comportement est sans doute le travail que SUN fournit pour développer le logiciel libre Apache. Cette entreprise s'occupe de toute la partie qui concerne l'adaptation de Java à Apache. En rendant compatible le langage qu'elle a développé avec le standard du marché, cette entreprise peut espérer vendre les outils de développement Java qu'elle produit (ainsi que son expertise dans le domaine). Un autre exemple est le soutien que Corel apporte au projet Wine, qui consiste à développer

un logiciel permettant de faire fonctionner des applications prévues pour Windows sous Linux. L'objectif de Corel est de pouvoir répondre à la demande du marché Linux sans redévelopper ses logiciels pour Linux. Dans ces deux cas, on voit que des entreprises sont prêtes à subventionner une plate-forme pour favoriser des logiciels fonctionnant grâce à elles. On peut estimer qu'il s'agit là de subventions croisées, où l'on subventionne un outil standard ouvert et complémentaire à son offre logiciel. Il s'agit là d'un comportement classique en informatique : Genthon [2000] souligne que, pour favoriser son standard (ou son produit), il faut, entre autre, «établir des alliances avec des co-producteurs» et «sponsoriser les premiers utilisateurs et les premiers produits complémentaires».
En terme de technologies d'utilisation, les produits les plus susceptibles de recevoir ce type de contribution sont les plates-formes, c'est-à-dire les produits supports d'interrelations technologiques fortes : ce sont elles que les producteurs doivent adapter pour faire fonctionner leurs offres logicielles. Plus la demande sera évolutive, plus la dynamique d'innovation sera forte, plus les entreprises seront obligées de contribuer au développement pour transférer une partie de la maintenance de leurs contributions à d'autres producteurs et surtout pour garantir que la plate-forme reste compatible avec leur logiciel (donc plus elles renforceront cette dynamique d'innovation).

On est bien entré dans la phase 2 de la production d'un bien public, phase où il est plus intéressant d'accompagner le mouvement dominant que de rester à l'écart. Il faut noter qu'il y a une grande convergence d'intérêts entre les stratégies des producteurs de logiciel et les stratégies des clients : tous sont prêts à supporter les logiciels «plates-formes», c'est-à-dire les logiciels à fortes interrelations technologiques et les logiciels supportant d'importants effets de réseau. Cela d'autant plus que ces logiciels deviennent des standards[31].

Passons maintenant aux éditeurs et distributeurs de logiciels. Ils sont importants pour les utilisateurs naïfs car ils facilitent l'accès aux solutions, tant au niveau géographique qu'au niveau de la facilité d'utilisation.

[31] Lerner & Tirole [2000] pensent aussi que la réputation peut également servir de signal en direction des recrutables sur l'intérêt et la qualité des projets développés dans l'entreprise. Nous discuterons ce point, fondamental dans la stabilité à long terme du Libre, dans la troisième partie.

Les diffuseurs de logiciel.

Ce métier d'éditeur et de distributeur de logiciel est le premier métier qui a donné lieu à une activité économique dans le logiciel libre : Richard Stallman distribuait des versions d'Emacs[32] sur bandes pour financer sa fondation dès 1985. Moins anecdotique, RedHat, Suse, Caldera sont des entreprises dont le premier métier a été de regrouper un certain nombre de logiciels libres (système d'exploitation et logiciels d'utilisation) sur un support facilement accessible (CD-ROM), dans des «distributions», et d'organiser la diffusion de ces distributions dans les réseaux de vente spécialisés dans les produits informatiques.

Deux choses font cependant douter de la rentabilité de ces entreprises : l'arrivée d'Internet et la liberté de copier les logiciels et les innovations qu'on y apporte. La diffusion d'Internet diminue le coût d'accès aux connaissances codifiées et notamment aux logiciels. Le métier de distributeur de logiciel semble donc peut rentable en lui-même, les prix risquant de se rapprocher du coût marginal de distribution, donc du coût du support, le CD-ROM. En plus, la copie est libre. Donc toute innovation, toute amélioration, dans le logiciel édité peut être copiée par les concurrents. Deux exemples soulignent la difficulté pour un éditeur/distributeur de rentabiliser ses investissements dans le secteur du logiciel libre. Mandrake a créé sa distribution de Linux en utilisant la distribution de RedHat (et notamment les innovations produites par cette entreprise, comme le système d'installation des logiciels, nommé RPM), en y ajoutant une interface graphique, Kde, au moment où RedHat attendait que l'autre interface graphique GNOME soit utilisable. Grâce à une alliance avec le distributeur Macmillan Software, Linux Mandrake est devenue la distribution la plus vendue aux États-Unis. Cette première place n'assure pas la rentabilité de l'entreprise et diminue d'autant les revenus de RedHat, dans un domaine à fort rendement d'échelle. D'autre part, le retrait de Corel montre que c'est un métier coûteux mais peu rentable[33].

Bien sûr, on peut arguer du fait que c'est un métier où les économies d'échelle sont importantes et que les éditeurs qui subsisteront pourraient devenir rentables. Mais le fait que la copie soit libre pousse à distribuer ces logiciels à un prix proche du coût marginal.

[32] Un des premiers éditeurs de texte et à l'époque le plus perfectionné.

[33] Corel a créé une distribution de Linux orientée vers l'utilisateur naïf en y ajoutant ses produits. Cette distribution était basée sur la distribution Debian, qui est un projet libre.

Il faut que les entreprises trouvent des sources de revenu complémentaires à la vente des distributions. Tous ces éditeurs proposent, en même temps que leurs distributions, des offres de service (service après-vente, adaptation de la distribution aux besoins), à tel point qu'on peut se demander si l'édition n'est pas uniquement un moyen de créer une marque pour vendre ce qui constitue le cœur de l'offre, le service[34]. Mais les éditeurs sont alors en concurrence avec les entreprises de service «pures» et on peut se demander quel avantage concurrentiel apportent la maîtrise et l'édition d'une distribution. Cela ne peut se comprendre que si l'on étudie de plus près la relation de service et les incitations qui sont créées par cette relation de service. Mais avant, il nous faut dire quelques mots sur la dernière étape de la diffusion d'un logiciel, la période où il est devenu un standard, mais où il devient difficile d'apporter des innovations.

2.2.3 L'étape 3 de la diffusion : gérer l'après-développement.

Foray & Zimmermann [2001] présentent le problème que pose cette étape de la façon suivante : à partir du moment où les plus grosses difficultés techniques ont été levées et où les quelques nouveaux utilisateurs, naïfs, ont des besoins qui ne sont pas ceux des utilisateurs-développeurs, n'y a-t-il pas un risque que ces derniers soient démotivés et abandonnent, les-uns après les autres la production de ce logiciel ?

On peut d'abord répondre que la construction modulaire permet que le noyau du logiciel, mature, n'évolue plus, pendant que les fonctionnalités continuent à se développer : c'est ce qui est en train de se produire pour Apache et va bientôt se produire pour Linux (d'après son créateur, Linus Torvalds) : donc ce n'est pas parce que le noyau ne se développe plus qu'il n'y a plus d'évolution du logiciel. De plus, il faut souligner que la maturité des logiciels vient avec la diminution de ces besoins. Donc qu'il est plutôt efficace, d'un point de vue économique, que les contributeurs soient moins nombreux, qu'ils s'intéressent à d'autres projets libres. Enfin, les contributeurs marchands seront tentés de continuer à faire des propositions, si ces propositions permettent de toucher de nouveaux utilisateurs avec les logiciels qu'ils distribuent.

Cependant, il est certain que la diminution du contrôle sur la distribution officielle est

[34] Depuis le premier trimestre 2001, les revenus de RedHat générés par les services dépassent ceux générés par la vente des distributions.

un risque autrement plus réel : un logiciel mature peut nécessiter des changements mineurs, et il est important que le noyau de développeurs puisse valider ces changements, donc qu'il continue à exister. Comment évaluer ce risque ? À nouveau, plus les logiciels seront utilisés moins ils seront menacés car, la population d'utilisateurs étant plus importante, il y a statistiquement plus de chance de trouver les erreurs contenus dans les logiciels, mais aussi un utilisateur capable de les réparer. On pourrait aussi se poser la question de savoir si ce risque est plus important pour le Libre que pour d'autres systèmes de production, mais cela sortirait du cadre de ce chapitre et nous reporterons la fin de cette discussion au chapitre suivant. Car, pour que ce problème se pose, il faut que des logiciels se soient diffusés chez tous les utilisateurs, ce qui nécessite que l'organisation libre assure la stabilité de la relation utilisateurs-producteurs.

2.2.4 Conclusion : des contributeurs aux motivations variées mais d'accord sur les objectifs de l'organisation libre.

Finalement, l'organisation de production de logiciels libres, plus qu'un «forum» de rencontre entre employés et employeurs, nous paraît être une organisation destinée à créer une structure d'information permettant aux entreprises de signaler la qualité de leur travail en direction de leurs clients et plus généralement de leurs partenaires.

Après l'étude que nous venons de faire, nous constatons l'importance de la licence GPL : c'est elle qui permet de garantir la qualité des logiciels, en empêchant le contrôle monopolistique sur un logiciel, qui est à la base des motivations des utilisateurs, et qui est de plus en plus important pour les entreprises. D'un autre côté, ces entreprises ont besoin de fournisseurs pour sous-traiter la production de logiciel parce qu'à cause de l'étendue de l'offre de logiciel, elles ne peuvent pas disposer en interne de toutes les compétences nécessaires.

Pourtant, cette GPL pose un problème : parce que les fournisseurs ne peuvent pas vendre le logiciel, il y a un risque qu'ils ne puissent pas se rémunérer et que donc la relation utilisateurs-producteurs ne soit pas viable. En introduisant les services, nous allons voir que le Libre assure en même temps l'implication à long terme des producteurs marchands de logiciels (notamment des producteurs de service) et leur rentabilité. Tout cela car le

Libre permet de résoudre en partie les problèmes de sélection adverse et de hasard moral inhérents à la relation de service.

2.3 Le Libre, une organisation de production d'un bien public pérenne grâce aux relations de service.

Nous allons étudier, dans cette section, la stabilité de long terme des relations utilisateurs-producteurs. Cette stabilité passe par l'implication des producteurs dans le développement de logiciels libres (et pas seulement par des contributions à l'étape 2 de la production), en même temps que par le développement d'activités économiques pérennes, nous l'avons dit.

Nous avons vu que la satisfaction des clients vis à vis des services informatiques est très faible. En nous appuyant sur les travaux de Karpik [1996], nous montrerons qu'il existe, en théorie, des dispositifs susceptibles d'améliorer cette relation de service. Nous défendons alors l'idée que l'organisation libre permet de mettre en place effectivement ces dispositifs dans l'industrie des technologies d'information.

2.3.1 Une classification des systèmes permettant d'abaisser les asymétries d'information.

Nous distinguerons avec Karpik [1996] deux niveaux dans ces dispositifs permettant d'abaisser les asymétries d'information[35] : d'une part, ceux qui permettent de sélectionner le partenaire, en faisant baisser les risques dus à la sélection adverse et, d'autre part, ceux qui, une fois le partenaire sélectionné, garantissent que la relation d'échange sera efficace.

Les dispositifs de sélection du partenaire.

Les dispositifs individuels.

Le premier dispositif est fondé sur l'information personnelle, c'est-à-dire sur des infor-

[35]Karpik parle de dispositifs de confiance. Nous avons choisi de ne pas employer ce terme, qui renvoie à des débats théoriques complexes sur ce qu'est la confiance, débat que nous ne pouvons traiter ici.
Sans employer le terme de confiance, mais en parlant de dispositifs permettant de faire baisser les asymétries d'information et de faire converger les plans des acteurs (donc de dispositifs d'incitation), l'analyse et le classement qu'il fait restent tout à fait valables.

mations en provenance de personnes dont on estime l'avis valable : «le «marché-réseau» se distingue du marché conventionnel fondé sur une relation bilatérale autour d'une information publique par une relation triangulaire dans laquelle une tierce partie, à laquelle on fait confiance pour son expérience et son désintéressement, devient la source et le garant d'une information qui crée les conditions nécessaires aux engagements contractuels» (ibid, p 532). On pense d'abord aux avis recherchés par l'utilisateur auprès de tiers. Nous rajouterons aussi, car ils ont une importance en informatique, les références de projets réalisés que les entreprises produisent. Il s'agit, là aussi, de fournir au «client» des sources d'informations sur la qualité du prestataire, la seule qualité qui compte à ce stade de l'échange. Certains auteurs y voient la principale source de création de confiance (donc, pour nous, de façon équivalente ici, de la baisse des incertitudes informationnelles)[36]. C'est d'autant plus vrai que les utilisateurs peuvent s'adresser à un tiers ayant plus de connaissances du domaine qu'eux.

On a donc là un dispositif qui améliore l'information «privée» moyenne détenue par les acteurs, mais de façon très inégale, et qui souligne encore plus l'importance de l'hétérogénéité des connaissances. Avec ce système, seuls les acteurs ayant le plus de compétences peuvent prendre le risque d'essayer une nouvelle offre et on peut penser que c'est à partir d'eux que l'information (et donc l'adoption) va se diffuser[37]. Lucien Karpik souligne bien les autres inconvénients de ce dispositif : «la circulation de l'information est lente, restreinte et les «producteurs» ne disposent que de moyens limités pour jouer activement de leurs qualités afin d'assurer leur développement économique» (ibid, p 533). On peut alors se demander s'il est possible de construire des dispositifs permettant de mieux sélectionner les acteurs et de diffuser plus rapidement cette information. Ce sont des dispositifs permettant de construire une information collective, ce que Karpik appelle la confiance «impersonnelle». Ils sont plus difficiles à envisager, d'une part parce qu'il ont été moins étudiés que les réseaux, d'autre part parce que leur construction est complexe et qu'ils sont assez hétérogènes.

[36]«I have argued that social relations, rather than institutional arrangements or generalized morality, are mainly responsable for the production of trust in economic life.» (Granovetter [1985], p. 491).

[37]Le modèle de Steyer & Zimmermann [1996] illustre très clairement l'importance de ces premiers adopteurs et de leur localisation dans la diffusion d'une information. Voir aussi les travaux d'Orléan sur les marchés financiers et la construction d'une croyance collective à partir d'informations, de croyances individuelles.

Les dispositifs d'information collectifs.

On peut, toujours selon Karpik, classer ces dispositifs en trois catégories : les classements, les appellations et les guides.

Le premier dispositif regroupe les indicateurs qui «comme le diplôme, le palmarès et la réputation, sans modifier les biens et les services, rendent plus ou moins fidèlement visibles leurs singularités par le moyen de hiérarchies publiques. [...] Cette pratique trouve sa limite dans son degré de précision : elle exclut l'individualisation et n'autorise qu'une canalisation grossière des préférences entre les classes de service» (ibid, p. 533). La limite de ces dispositifs tient aussi à leur construction, discrétionnaire et fondée sur des critères non explicites donc à la construction de la confiance en le dispositif lui-même.

Le deuxième dispositif est l'appellation, c'est-à-dire un mécanisme qui, «tels les labels, les certifications, les appellations d'origine contrôlée, les marques (au moins partiellement) associent des noms à des constructions délibérées de la singularité» (ibid, p. 535). Il s'agit ici, de s'engager sur un certain nombre de critères et de convaincre les acheteurs que ces critères ont un lien avec la qualité attendue du produit. On peut dire, grossièrement, que le jugement se fait avant la construction du produit, alors que dans le cas précédent, il la suit. Deux questions se posent, celle de la légitimité de ces critères dans la définition de la qualité et celle de la légitimité (qui va de pair avec l'efficacité) de l'organisme chargé de vérifier que ces critères sont respectés. Finalement, là encore, il y a un problème de construction des dispositifs.

Le troisième dispositif, enfin, est le guide, ou encore, «l'autorité douce dont les préférences, lorsqu'elles rencontrent la docilité volontaire, permettent de dénouer les affres de l'incertitude sur la qualité» (ibid, p. 536). Il s'agit donc de l'avis d'un agent, qui le rend public et qui, parce qu'il est reconnu par les utilisateurs, produit une hiérarchisation, en même temps qu'un système d'analyse, qui lui est propre. Lucien Karpik explique, au sujet de ce dispositif, que, «[c]omme les autres dispositifs de jugement, les guides tirent leur autorité de la confiance qui leur est conférée mais ils s'en distinguent par leur objet : l'évaluation porte directement sur la qualité. [...] [c]e sont eux qui, avec le réseau, forment la figure exemplaire de l'économie de la qualité» (ibid, p 537).

Il existe là encore des inconvénients à y faire appel : «la circulation de l'information

est lente, restreinte et les «producteurs» ne disposent que de moyens limités pour jouer activement de leurs qualités afin d'assurer leur développement économique» (ibid, p 533). Sachant cela, on peut se demander s'il est possible de construire des dispositifs permettant de sélectionner mieux les acteurs et de diffuser cette information plus rapidement.

Discussion.

Les deux premiers dispositifs sont des dispositifs qui garantissent un certain niveau d'effort, une obligation minimale de moyens, alors que le troisième juge du résultat, en proposant, soit des critères propres, soit, simplement, une évaluation privée de la performance (nous pensons ici aux guides gastronomiques qui attribuent une note à des restaurants sans que cette notation soit reliée à des critères explicites, mais principalement au niveau d'excitation des papilles gustatives des examinateurs). Si l'on «note» un résultat dans ce cas-là, on le fait dans une relation particulière supposée représentative des résultats possibles. Plus les besoins et les situations sont hétérogènes, plus ce dispositif risque d'être pris en défaut. On voit que les différents systèmes sont complémentaires, puisqu'ils ne jugent pas les mêmes éléments de la qualité. Dans les trois cas, il y a construction et publication par un tiers d'un système d'évaluation.

Finalement, le système d'évaluation fonctionne de la même façon que dans le cas du «marché-réseau» : on demande à un tiers de garantir les qualités de l'offreur de service. La différence, ici, est que le tiers est un agent public ou, pour reprendre le vocabulaire de l'économie des réseaux, un agent qui intervient au niveau global contrairement au cas précédent, où il s'agissait de relations purement locales. Le problème de sélection adverse s'est déplacé, mais il existe toujours. La construction de la réputation de l'agent dépend de sa qualité (État, organisme de normalisation ou de certification, regroupement professionnel, éditeur privé, etc.), elle peut être plus ou moins longue et passer par des réseaux locaux ou globaux. Il y a toujours un «détour de production» pour faire baisser les asymétries d'information, puisque c'est un agent tiers qui engage sa réputation sur la qualité d'une offre. La réputation est constatée à un moment donné, il existe des agents qui sont reconnus comme des références capables de garantir la qualité d'une relation d'échange incertaine. Ces engagements sur des relations et la qualité effective des relations vont renforcer ou détériorer le crédit des garants et donc faire évoluer les sentiments des

utilisateurs sur ce système de certification[38].

Cette construction temporelle et le problème de la pérennité des relations d'échange dans le temps vont être à la base de la résolution du deuxième problème qui menace l'efficacité de la relation, c'est-à-dire le problème du hasard moral, de la qualité de l'effort pendant la relation elle-même.

La consolidation de la relation face au hasard moral.

C'est la tâche des «dispositifs de promesse», comme les appelle Karpik. Il distingue deux types de dispositifs, l'un fondé sur la dispersion des risques par la dispersion des commandes, l'autre sur la production de dispositifs permettant de faire baisser les asymétries d'informations, donc les risques de hasard moral.

Dans une relation d'échange de service, il faut s'attendre à avoir à négocier en permanence le contenu du service et de sa rémunération et à ce que les acteurs soient sans arrêt menacés par le comportement opportuniste de la partie adverse. L'objectif de ces dispositifs est de garantir que «dans des situations imprévues, les engagements seront guidés par l'esprit du contrat ou par un principe d'équité» (ibid, p. 540). C'est-à-dire qu'on suppose que le partenaire va avoir un comportement prévisible bien que non-écrit et susceptible de respecter les intérêts des deux parties, intérêts non spécifiés, sinon par un «principe d'équité». Le problème du hasard moral est aussi celui de l'effort engagé et du degré de professionnalisme de la personne avec qui on est en relation (c'est le même type de discussion que l'on retrouve à propos de l'incomplétude du contrat de travail). C'est aussi «la question de l'assurance que, dans la logique même de la relation de service, le prestataire ou partenaire prenne effectivement en compte toute la spécificité de la question : peut-on éviter la tendance presque spontanée à vouloir ramener ou réduire la question ou le problème à des cas communs et de procéder, pour la solution ou réponse, par voie de réplication[39] ?» (De Bandt [1998]).

[38] On a bien là toutes les caractéristiques de la construction d'une convention, au sens de Gomez [1994], qui se développe et évolue localement et globalement (cette construction est proche de la théorie structurelle développée par Burt [1982], complétée par une modélisation de son évolution temporelle par Leydesdorff [1991]).

[39] De Bandt note que «le client/consommateur fait appel à un prestataire ou partenaire, parce qu'en raison de la complexité de son problème, il a besoin de compétences qu'il n'a pas, mais de ce fait aussi il n'a pas la possibilité de vérifier que la solution se situe au niveau souhaitable et possible de complexité.

Évidemment, la construction de ces garanties ne peut être comprise que dans une perspective temporelle, soit que la relation actuelle soit due à une succession de relations (qui sont autant de mises à l'épreuve), soit qu'elle soit garantie par des dispositifs institutionnels qui, eux aussi, ont dû être acceptés par des acteurs comme garants de la relation. Si l'on utilisait le langage de la théorie des jeux, on pourrait dire que ces dispositifs sont basés sur la construction d'un jeu répété (et à horizon infini) soit entre les partenaires de l'échange, soit entre un représentant d'un des partenaires, et l'autre partenaire (ou la population dont est issu l'autre partenaire). On retrouve l'idée de la règle construite dans l'organisation au cours du temps, règle qui évolue en permanence, en même temps que les institutions ou l'organisation qui la défendent.

La norme coproduite.

C'est une construction entre deux partenaires dépendant chacun de l'engagement réciproque de l'autre sur le long terme pour pouvoir tirer bénéfice de l'échange. L'exemple type de ce genre de relations est le partenariat entre, par exemple, un fournisseur et une entreprise, lorsque l'entreprise dépend de l'efficacité du fournisseur pour réaliser ses objectifs en terme de qualité et de délais et lorsque le fournisseur dépend de l'entreprise pour ses débouchés. On retrouve ici les caractéristiques des relations qui se sont développées dans l'industrie automobile avec le «juste à temps» (Altersohn [1992], Baudry [1995], par exemple) . Les deux partenaires ont intérêt à la prolongation de la transaction et vont chercher à la protéger. «[Cette] protection [...] dépend d'un ensemble commun de principes d'orientation de l'action (réciprocité, équité, loyauté, mobilisation) façonnés par les partenaires contractuels et convertis, au cours d'un processus historique contingent, en «obligation morale» associée à des sanctions diffuses» (Sako [1992], p 10)[40]. Cette construction,

Il peut donc se retrouver avec une solution ou réponse trop simple, sans grande valeur, et cela sans avoir rien à redire».
Ceci étant dit, il faut aussi remarquer que l'efficacité d'une relation de service peut résider dans la répétition d'une solution éprouvée.
Si cette répétition, au moins partielle, n'était pas possible, cela voudrait dire qu'il n'y a aucun apprentissage possible dans le métier de fournisseur de services. L'efficacité réside pourtant dans l'identification de ce qui est commun, pour pouvoir dégager des ressources sur le traitement du spécifique.

Nous avons vu que, dans la production de logiciel, les gains de productivité sont dus, pour une bonne partie au développement de progiciels, qui sont une réponse standard à des besoins différents. Si, actuellement, les offres s'orientent vers l'individualisation des solutions, au moins pour une catégorie d'utilisateurs, cette individualisation se fait sur une base commune, standard.

[40]Il y a construction de relations entre les entreprises, mais aussi entre les personnes chargées de mettre en place cette relation. «En fait, loin de pouvoir rabattre le partenariat sur une relation globale - le contrat

qui s'inscrit dans la durée, ne peut exister qu'entre entités capables de négocier des dispositifs pour orienter l'action. Il faut qu'il y ait une certaine égalité dans les connaissances des partenaires pour qu'ils puissent apprendre à évaluer les efforts de l'autre grâce à la durée de l'échange.

Mais ce n'est pas toujours le cas des relations de service, qui peuvent être aussi des relations uniques, ou dans lesquelles les participants auront des niveaux de connaissance très différents. Comment construire un système qui va rendre la «défection» non rentable pour les partenaires, ce qui veut dire abaisser les asymétries d'information et surtout faire en sorte que le résultat de la relation ponctuelle ait un impact sur le futur du producteur pour les obliger à s'impliquer dans cette relation. Ce type de dispositifs est appelé «norme unilatérale» par Karpik.

La norme unilatérale.

«Comment garantir la confiance dans une relation d'échange structurellement asymétrique ?» (ibid, p. 542). Il s'agit en fait de substituer à cette relation intemporelle et unilatérale une relation de long terme, avec un partenaire stable. Pour cela, Karpik identifie deux moyens.

Tout d'abord, la réduction des risques au niveau étatique (par l'instauration de diplômes, condition sine qua non pour pratiquer, ou par l'instauration de certification) permet de réduire le problème de sélection adverse.

Mais pour réduire le problème du hasard moral, on peut avoir recours, et c'est le cas pour les avocats ou les médecins, à «la construction volontaire d'une morale professionnelle fondée sur l'auto-discipline collective», en même temps que la démonstration publique de l'attachement à cette auto-discipline comme preuve de l'implication du gardien de la norme. Il s'agit de substituer à une relation interpersonnelle une relation entre un individu et une organisation, identifiée et stable, voir même entre une opinion publique et une organisation professionnelle. La relation inter-individus n'est alors qu'une «rencontre», qu'un «jeu» dans le jeu répété à horizon infini entre les utilisateurs et le gardien de

entre les deux entreprises - il faut à l'inverse reconnaître que le terme désigne une multiplicité de systèmes d'actions, une multiplicité de dispositifs normatifs et une multiplicité de relations de confiance» (Karpik [1996], pp. 544-545).

la norme, pour filer la métaphore de la théorie des jeux. Tant que les individus dont les organisations professionnelles garantissent l'effort ne sont pas trop nombreux à faire défection (c'est-à-dire tant que le système de règles et de sanctions fonctionne), ce dispositif peut fonctionner. Tant, aussi, que les règles, les obligations professionnelles que s'impose la corporation sont en phase avec les demandes de la collectivité.

Il y a donc, dans la construction de cette règle, de cette convention, constante renégociation des besoins et des règles. On peut dire, comme Karpik, que «la confiance portée à la norme unilatérale, et qui est elle-même unilatérale, ne fait que mesurer le succès d'une construction sans cesse menacée par la méfiance généralisée». Nous allons voir que le Libre permet de construire de tels dispositifs.

2.3.2 Le libre, un outil pour améliorer la relation de service en informatique.

Nous avons déjà vu, au premier chapitre, que le Libre permettait de construire des normes. Nous allons maintenant voir que ces normes peuvent être utilisées par les producteurs de service car elle facilite leur travail. Mais que, plus intéressant, cela leur permet aussi de signaler leur qualité propre et qu'ainsi il y a construction de dispositifs d'information collective. Enfin, la GPL est un puissant outil pour luter contre le hasard moral et nous l'analyserons comme une norme unilatérale construire par la profession des entreprises de services. Au fil de l'analyse, il apparaîtra que Libre et le dispositif d'information qu'il met en place permettent alors aux producteurs de valoriser leur compétences, donc de s'assurer de la pérennité de leur activité.

Le Libre permet de construire des normes.

Ouvrir pour normaliser.

Avec le logiciel libre, les utilisateurs-développeurs ont donc construit un système d'assurance-qualité sur les produits utilisés, système indépendant des distributeurs de logiciels. Et, de la même façon que dans le cas de l'industrie automobile, cette assurance de qualité sur les produits utilisés facilite les relations entre producteurs de technologies d'utilisation et utilisateurs de ces mêmes technologies : les producteurs peuvent garantir

plus facilement la fiabilité des logiciels (libres) qu'ils utilisent, parce qu'ils sont capables d'évaluer cette qualité au travers des normes mises en place dans le développement, parce qu'ils disposent d'un réseau d'assistance et parce qu'il sont capables d'intervenir eux-mêmes sur les logiciels[41].

Le développement par des entreprises de services des logiciels libres peut être vu comme la création d'une norme unilatérale, puisqu'il s'agit, collectivement, de se coordonner pour produire des composants, des briques logiciels fiables. Encore faut-il que, individuellement, ces entreprises soient incitées à respecter la norme. L'évolution des marchés informatiques et l'évolution des technologies de développement expliquent qu'elles le soient.

Les incitations des entreprises à respecter les normes.

On a souvent opposé la production de logiciel à la production de service[42], mais, on a vu dans le chapitre 1 que la production de logiciel s'oriente aujourd'hui vers la production de briques élémentaires et l'assemblage de ces briques pour développer des solutions sur-mesure à partir de composants communs. Le coût de développement d'un logiciel est donc de plus en plus réparti dans le temps, se rapprochant plus d'une structure de production de service où l'on ne développerait la fonctionnalité manquante qu'au moment où elle s'avèrerait nécessaire. Les contributions des entreprises de services ne concernent pas la production complète d'un logiciel mais la production de ces composants pour des clients qui préfèrent que ces logiciels soient libres pour ne pas dépendre de leur fournisseur.

On tient là un nouveau rôle pour des entreprises de service, complémentaires au rôle de certificateur de qualité : celui de mutualisateur des coûts de développement. En effet, un composant développé pour un client peut être réutilisé pour répondre au besoin d'un autre et un «trou de sécurité» détecté chez un client peut être corrigé chez l'ensemble des clients de l'entreprise (c'est aussi vrai pour les clients des entreprises de services concur-

[41] Il est remarquable de noter à ce propos que, pour éviter la reproduction des «splits» successifs des versions d'Unix, les entreprises informatiques ont mis en place des organismes chargés de garantir la compatibilité entre les différentes versions et distributions de Linux et chargés de publier des recommandations techniques sur la façon de programmer les applications pour qu'elles fonctionnent avec ce système, dans l'esprit de la norme POSIX.
C'est le Free Standard Group (http ://www.freestandards.org/), qui regroupe le Linux Standard Base (LSB, www.linuxbase.org) et la Linux Internationalization Initiative (LI18NUX, www.li18nux.org). Parmi les membres de ces comités, on trouve, entre autre, Corel, RedHat, Mandrake, SuSe, VA Linux Systems, Turbo Linux, mais aussi IBM, SUN, SAP, SCO, SGI, ou l'Open Group (chargé de la standardisation sous Unix).

[42] Genthon [2001] explique bien en quoi ces métiers différaient.

rents, d'ailleurs). Ces services sont complémentaires car il s'agit, dans les deux cas, de mutualiser les «risques» de développement, de panne, bref les efforts à fournir pour garantir l'efficacité des solutions basées sur du logiciel libre. On constate, en étudiant les offres des entreprises de services comme Alcôve, Atrid, ACT, RedHat ou Mandrake, que ces entreprises proposent effectivement des services de certification des solutions et des services de développement, le plus souvent sur la base de forfaits qui dépendent du nombre de postes de travail. C'est un indice qu'il y a effectivement, pour elles, des effets de rendements croissants (sinon elles factureraient les clients à l'intervention, qui permet d'être plus proche des coûts réels dépensés pour chaque client).

Il est plus efficace aussi pour le client d'avoir recours à ces entreprises que de développer des compétences en interne car le suivi de l'évolution des logiciels libres est coûteux, le développement des compléments demande de bien connaître les logiciels, donc de posséder des spécialistes de ces logiciels. Ces spécialistes sont rares et les entreprises en ont rarement l'utilité à temps plein. On peut dire des entreprises de services qui basent leurs offres sur des logiciels libres qu'elles proposent gratuitement les connaissances codifiées que sont ces logiciels pour vendre les connaissances tacites qu'elles possèdent : la connaissance du fonctionnement intime des logiciels, la capacité des développeurs à produire des contributions qui fonctionnent, la capacité à faire accepter ces contributions par le noyau qui contrôle l'évolution des logiciels, etc. Ces entreprises sont les mieux placées pour accaparer les rendements croissants d'information et d'apprentissage générés par le développement, l'amélioration de logiciels[43]. C'est ce qu'elles vendent.

On va voir que l'organisation libre permet aussi de résoudre en partie le problème de la sélection adverse.

[43]Sébastien Namesh, un des fondateurs de la société «Virtual-net» (http ://www.virtual-net.fr/) spécialisée dans les solutions à base de logiciel libre, estimait, qu'en moyenne, le temps passé à s'informer sur les logiciels et leur évolution est de un jour par personne et par semaine dans son entreprise. (Cette information est tirée de sa présentation au colloque «Autour du Libre 2000», ENST Bretagne, http ://libre.enst-bretagne.fr/).
Le fait que, pour pouvoir profiter des innovations, il faille innover soi-même est maintenant bien connu de la théorie économique. Cohen & Levinthal ([1989], p. 569) ont montré que si «R&D obviously generates innovations, it also develops the firm's ability to identify, assimilate, and exploit knowledge from the environment». Ce qu'ils appellent «a firm's 'learning' or 'absorptive' capacity».

L'organisation «libre», un système pour signaler sa qualité (sélection adverse).

Dans un contexte où les marges sont faibles à cause de la difficulté à discriminer les qualités des différents offreurs (voir Dréan [1996], p. 84 et pp. 257-284), il est important de trouver des moyens de signaler sa qualité, soit en construisant une marque, soit en facilitant l'évaluation ex-ante de son travail par les clients. L'organisation «Libre» permet de construire un dispositif d'information collectif sur le travail des entreprises de services.

Contribuer pour signaler sa qualité.

Si le métier de ces entreprises est d'adapter des logiciels aux besoins d'une entreprise et de garantir le bon fonctionnement de ces logiciels, les contributions que l'on peut présenter sont autant de garanties de sa connaissance du logiciel[44]. Les entreprises peuvent, bien sûr, présenter les contributions qu'elles ont réalisées pour des clients, références classiques que toute entreprise de service affiche. Parce que, dans le cas du libre, les logiciels sont publics et les contributions signées, le client, à condition bien sûr de disposer des compétences nécessaires, peut effectivement évaluer le travail réalisé.

De plus, parce que les besoins des clients sont différents, il devient important pour ces entreprises de service de maîtriser un porte-feuille de logiciels et de développer les logiciels standards utilisés dans la plupart des offres pour pouvoir toujours présenter aux clients des réalisations en rapport avec leurs problèmes (et pour conserver des compétences sur ces logiciels). Cela explique que ces entreprises consacrent une partie de leurs investissements à des développements de logiciels libres, sans que ces développements soient directement utiles, affectables à un contrat. C'est une activité de recherche et c'est une attitude qui différencie ces entreprises des entreprises de service «classiques» qui consacrent souvent moins de 1 % de leur chiffre d'affaires à la recherche (voir Genthon[2001]). Dans un marché du service fondé sur la valorisation de l'expertise technique, cette activité de contribution renforce l'image de l'entreprise, sa notoriété et, par là même, sa marque. Pour la même raison, ces entreprises sont incitées à produire des corrections lorsqu'un bogue est décelé

[44]Mathias Herberts, dans le cadre d'«Autour du Libre 2000» nous a expliqué qu'il était parvenu à installer des solutions à base de logiciel libre dans une banque parce qu'il avait pu montrer qu'il était l'auteur d'une partie du code de ces logiciels, ce qui avait rassuré le client sur son expertise du produit.

Cette attitude a été confirmée par les autres membres de la table ronde (Table ronde «Une démarche commerciale basée sur du logiciel libre : quels risques et quelles garanties?», «Autour du Libre 2000», http ://libre.enst-bretagne.fr/AL2000/synt_at5.html).

ou à répondre à des questions techniques sur des forums. Toutes ces activités renforcent le signal sur leur capacité d'expertise et de réactivité, les deux qualités qui fondent leur spécificité et qui permettent de singulariser leur offre, donc d'augmenter leur marge.

Si ce système fonctionne avec des utilisateurs avertis, il y a un problème d'information vers les non spécialistes, les utilisateurs naïfs par exemple.

Informer les non-spécialistes.

Quelle organisation peut leur garantir la qualité des logiciels libres et évaluer, pour eux, la qualité des entreprises de services «libres»? On est là au cœur du problème posé par Lucien Karpik : quels dispositifs collectifs d'information sont accessibles aux utilisateurs naïfs dans le cas des logiciels libres?

L'organisation Libre n'offre un surcroît d'information qu'aux utilisateurs designers ou sophistiqués. Les autres utilisateurs ont besoin d'intermédiaires et il est logique qu'ils s'adressent aux intermédiaires qu'ils connaissent déjà. On assiste en ce moment à la construction des guides qui permettront à ces utilisateurs d'adopter des solutions libres. Les journaux «grands publics» (Info PC, ...), comme les journaux plus professionnels (Le Monde Informatique, 01 Informatique, etc.) commencent à s'y intéresser et intègrent des solutions libres quand ils proposent des évaluations. De même, des entreprises spécialisées dans les logiciels libres commencent à être intégrées dans les enquêtes de satisfaction et dans les enquêtes sur les offres de service. Enfin, des journaux spécialisés (Linux Hebdo) apparaissent. L'implication grandissante des entreprises, des marques traditionnelles de l'informatique (IBM, HP, SGI), dans la production et la distribution des logiciels libres, marques qui jouent traditionnellement le rôle de guide pour une partie des utilisateurs, est aussi un facteur décisif pour la construction de ces réseaux d'information collectifs. On se souviendra que c'est l'implication d'IBM sur le marché de la micro-informatique qui, en apportant sa caution de sérieux à ce produit, a initié sa diffusion dans les entreprises.

Cependant, si le Libre peut permettre de mieux discriminer les offres des entreprises de service, il faut aussi garantir que celles-ci travailleront mieux (problème du hasard moral), et que ces entreprises puissent développer un modèle économique suffisamment rentable pour assurer leur pérennité et donc la pérennité de leur service.

Un système d'incitations permettant de garantir la qualité et la pérennité du service.

S'il n'est pas possible de construire une offre commerciale rentable, il existe un risque que les entreprises soient toutes défaillantes et qu'on ne puisse pas trouver de partenaires pour assurer la continuité des services de support ou d'ingénierie que demande l'utilisation de ces logiciels. La relation de service, plus généralement la relation de client à fournisseur est proche d'une relation d'emploi et proche d'une relation de principal à agent. L'objectif pour le principal est de s'assurer que l'agent mettra tout en œuvre pour lui apporter le meilleur travail possible. L'objectif pour l'agent est de s'assurer une rémunération suffisante, à court et à long terme. L'ouverture des logiciels permet aux utilisateurs de contrôler leurs producteurs, mais aussi aux producteurs de signaler leur qualité.

L'ouverture donne un pouvoir de contrôle au client...

L'ouverture du logiciel, de la connaissance codifiée permet de faciliter la concurrence sur les services d'utilisation de cette connaissance (ingénierie, adaptation, développement à façon) et aussi de diminuer les coûts de sortie d'une relation de service : les logiciels développés dans le cadre de la relation de service peuvent, eux aussi, être publiés en libre et incorporés dans les versions officielles de ces logiciels. En même temps, les besoins des utilisateurs se déplacent vers des solutions où l'interopérabilité avec les éléments déjà présents dans l'entreprise (des logiciels, qui sont connus, mais aussi des besoins propres, qu'il faut définir) et avec les outils utilisés à l'extérieur de l'organisation (donc inconnus) est garantie. Donc vers des solutions ouvertes et adaptées, du «sur-mesure» de masse. Là encore, le logiciel libre est un outil adapté parce que l'organisation de production garantit l'ouverture, la normalisation des interfaces des composants et que cette construction par composants facilite l'adaptation des logiciels aux différents besoins.

... mais crée aussi des incitations pour les producteurs.

Tous ces éléments qui favorisent le client sont autant d'incitations pour l'agent, c'est-à-dire le producteur, à réaliser le meilleur service : s'il est maintenant jugé sur des critères de disponibilité, d'adaptabilité et d'évolutivité de la solution, il n'est plus jugé sur des

objectifs de production en terme de quantité de logiciel produit ou en terme de temps passé. Il s'agit de trouver, d'assembler des composants qui permettent de répondre à des tâches précises et d'assurer la maintenance de ces composants. Il y a une évolution vers les services de substitution, services de mise à disposition de capacités techniques entretenues. Or ces services sont caractérisés par des relations de long terme, souvent basées sur des rémunérations fixes, renégociables périodiquement.

Ici se trouve l'explication du développement des relations de coopération[45] entre les fournisseurs des différents composants logiciels que nous avons constaté.

Le développement de relations de coopération entre clients et fournisseurs.

Pour les mêmes raisons, se développent des relations de coopération entre des clients qui ont des besoins en terme de technologies d'utilisation et des fournisseurs qui sont capables de garantir les outils qui réalisent ces besoins plus efficacement que leurs clients. En effet, ils peuvent intégrer des rendements croissants d'apprentissage dus à l'entretien et au développement de ces outils.

Comme dans toute relation client-fournisseur basée sur la coopération, les producteurs sont incités à faire de la recherche et du développement, c'est-à-dire à participer au développement de projets libres, pour maintenir leur avance en terme d'expertise mais aussi pour signaler à leurs clients qu'ils continuent à développer cette expertise. Elle sera un des atouts majeurs dont disposera le fournisseur de service lors de la renégociation du contrat ou lors de la négociation de nouveaux contrats de service. D'autre part, cette implication dans les projets libres est nécessaire pour pouvoir intégrer les nouveaux développements des logiciels ou pour pouvoir faire accepter facilement ses contributions. En effet, c'est cela qui va faire diminuer les coûts de production des services d'assemblage et de certification et donc à rendre concurrentielle, dans le temps, l'offre de service.

La dernière condition pour avoir une relation inter-entreprise coopérative est que la concurrence entre fournisseurs ne soit pas trop exacerbée, que le client restreigne sa relation de service à quelques fournisseurs privilégiés, mais qu'il y ait pluralité des offres pour garantir l'effort du fournisseur. Cette condition s'impose d'elle-même dans le cas du Libre :

[45] Pour une étude de ces relations, on consultera Baudry [1995]. Nous reprenons dans la suite les différents critères qu'il estime nécessaires pour avoir une relation coopérative, qu'il nomme relation «primaire».

il n'y a pas beaucoup de développeurs pour chaque logiciel libre, ce qui veut dire que le nombre d'entreprises qui peuvent développer un pôle de compétences pointues autour de chaque logiciel est forcément petit. Il faut que ces entreprises aient suffisamment de demandes techniques pour amortir le coût d'acquisition de ces compétences.

Donc, à cause de ces effets d'apprentissage, à cause de la difficulté à diffuser les connaissances tacites qu'il faut maîtriser pour suivre et influer l'évolution d'un logiciel libre, ce rôle sera forcément réduit à quelques entreprises. Elles regrouperont en leur sein les spécialistes des logiciels et les mettront à disposition des entreprises clientes. Elles auront construit des marques fortes, reconnues par les utilisateurs développeurs de logiciels et connues des clients (designers, bien sûr, mais sans doutes aussi sophistiqués voire naïfs).

Le Libre construit un système d'incitation entre fournisseurs de services et utilisateurs, où la concurrence permet de garantir l'effort du fournisseur tout en lui assurant un contrôle sur des actifs spécifiques, des connaissances accumulées, qui lui assurent son revenu et garantissent sa pérennité. On comprend alors pourquoi les entreprises Libre semblent «valoriser» la concurrence[46] : son existence, favorisée par la libre disponibilité des sources, garantit au client que son fournisseur ne sera jamais en position de monopole, donc qu'il ne sera jamais complètement captif de son offre. Cet engagement est d'autant plus crédible que le logiciel est un standard ; l'investissement dans la recherche de la part du fournisseur sera d'autant plus important que la dynamique d'évolution est forte (il faut que la solution reste compatible avec le logiciel) ou que le logiciel est contrôlé par une seule entreprise (car alors le risque de se voir «enfermé» dans un standard propriétaire est plus fort)[47].

2.4 Conclusion.

Nous avons montré dans ce chapitre que l'organisation de production et de diffusion d'informations «libres» permettait de construire un système de relation clients-

[46] On lira, à ce propos, les déclarations des producteurs tels que ACT Europe, Mandrake ou Alcôve au workshop «NEL» (http ://parmentille/~njullien/rntl/workshop1/)

[47] L'exemple du pneu «Radial», inventé par Michelin, est une bonne illustration de ce phénomène : ce pneu, qui se dégonfle très lentement après une crevaison n'a intéressé les constructeurs automobiles qu'à partir du moment où Michelin a accordé une licence de production à Goodyear, son principal concurrent, cédant par là même son monopole sur l'exploitation de cette innovation technologique.

fournisseurs efficace quand les clients sont capables d'évaluer les compétences techniques de leurs fournisseurs.

Parce que ces utilisateurs ont besoin de garantir la pérennité de leurs investissements et la compatibilité des logiciels, parce qu'ils sont souvent dans un rapport de dépendance par rapport à des producteurs de logiciels en situation dominante, ils sont intéressés par l'utilisation du libre. Comme le montre von Hippel [1988], dans de nombreuses situations, il est aussi plus intéressant pour eux de contribuer au développement de ces logiciels en publiant les modifications qu'ils y apportent, plutôt que de les garder secrètes car cela leur permet d'externaliser la maintenance de ces solutions. La construction modulaire des logiciels libres, parce qu'elle diminue le coût de production d'une nouvelle fonctionnalité, est très favorable à ce type de collaboration. La licence GPL est aussi un élément important car c'est elle qui garantit à ces utilisateurs-producteurs marchands que leur travail ne sera pas approprié et qu'ils auront toujours un contrôle sur l'évolution de ce travail (ou au moins que le contrôle ne retombera pas dans les mains d'un fournisseur, recréant une situation de dépendance).

Du côté des producteurs, il existe d'abord des motivations indirectes à contribuer au développement de certains logiciels libres. Les challengers d'une entreprise qui contrôle un standard peuvent avoir intérêt à subventionner un standard concurrent pour l'affaiblir. Surtout, les entreprises qui dépendent d'un standard pour réaliser leur activité (comme SUN avec Apache) ont intérêt à ce que ce standard soit le plus ouvert possible, pour ne pas dépendre de la stratégie de l'entreprise qui le fournit. Les motivations ne sont alors pas très différentes de celles des utilisateurs marchands. Mais l'évolution de la relation utilisateur-producteur en informatique crée aussi des incitations de long terme pour ces entreprises. La production de logiciel s'oriente aujourd'hui vers la production de briques élémentaires et l'assemblage de ces briques pour développer des solutions sur-mesure à partir de composants communs (Horn [2000b], Beugnard [2001]). Le coût de développement d'un logiciel est donc de plus en plus réparti dans le temps, se rapprochant plus d'une structure de production de service où l'on ne développerait la fonctionnalité manquante qu'au moment où elle s'avérerait nécessaire. Or l'industrie du service est caractérisée par de fortes asymétries d'information (sélection adverse et hasard moral) qui nuisent à son efficacité.

Avec le Libre, les utilisateurs-développeurs ont construit un système d'assurance-qualité sur les produits utilisés, indépendant des distributeurs des logiciels. Cette assurance de qualité sur les produits utilisés facilite les relations entre producteurs des technologies d'utilisation et utilisateurs de ces technologies : les producteurs peuvent garantir plus facilement la fiabilité des logiciels (libres) qu'ils utilisent, parce qu'ils sont capables d'évaluer cette qualité au travers des normes mises en place dans le développement, parce qu'ils disposent d'un réseau d'assistance et parce qu'il sont capables d'intervenir eux-mêmes sur les logiciels. Le développement par des entreprises de service des logiciels libres peut être vu comme la création d'un engagement collectif de la part des producteurs de logiciel pour mieux se coordonner pour produire des composants, des briques logiciels fiables. Ces entreprises de services basés sur des produits libres développent des offres de certification des solutions et de développement de programmes, de mutualisation des «risques» de ces développements, de panne, bref des efforts à fournir pour garantir l'efficacité des solutions basées sur du logiciel libre. On peut alors dire que ces entreprises de service proposent gratuitement les connaissances codifiées que sont ces logiciels pour vendre les connaissances tacites qu'elles possèdent : la connaissance du fonctionnement intime des logiciels, la capacité des développeurs à produire des contributions qui fonctionnent, la capacité à faire accepter ces contributions par le noyau qui contrôle l'évolution des logiciels, etc. Elles sont les mieux placées pour accaparer les rendements croissants d'information et d'apprentissage générés par le développement et l'amélioration de logiciels.

La licence GPL est alors un élément fondamental pour garantir leur engagement : elle assure, parce que ces producteurs de logiciels ne sont plus en situation de monopole sur la production du logiciel, qu'ils porteront leurs efforts sur la relation de service car c'est là dessus que va porter le jugement des utilisateurs. L'organisation Libre, fondée sur la GPL est un système qui permet de résoudre une partie du hasard moral qui caractérise l'industrie du service et particulièrement du service informatique (voir De Bandt [1998]). Et parce qu'elles doivent signaler leurs capacités (nous reprenons là l'argument de Lerner & Tirole [2000], mais dans le cas des entreprises), que ce label de qualité est délivré par l'organisation libre au vu des contributions, elles doivent se conformer aux normes de qualité qui caractérisent l'organisation libre lorsqu'elles contribuent au développement

de logiciels libres (résolution d'une partie de la sélection adverse). Mieux, comme elles doivent entretenir leurs connaissances tacites sur l'évolution, les caractéristiques des logiciels, elles sont incitées à produire des logiciels libres, en dehors des logiciels développés pour leurs clients.

On a alors plusieurs systèmes de production de logiciels, qui reposent sur des formes d'incitations différentes et qui organisent différemment l'échange de logiciel.

Tableau 2.3 — Les différents systèmes économiques de production dans le logiciel.

		Brevets / Droits d'auteur	**Science ouverte**	**Logiciel libre**
Création et divulgation	*Incitations*	Sur-profits et protection	Réputation par reconnaissance par les pairs et ressources financières en résultant.	Réputation par reconnaissance par les pairs et acceptation de la contribution. Profit.
	Système de contrôle	Tribunaux grâce à des procès.	Conventions, auto-régulation de la communauté scientifique.	Licence GPL.
Distribution	*Outils pour l'échange et la publication d'information*	Bases de données de brevets maintenues par des agences publiques.	Journaux scientifiques, conférences, courrier électronique.	Internet (courrier électronique, listes de diffusion, sites Web, groupes de news).

source : Dalle & Jullien [2000].

Nous avons montré que le modèle Libre était un modèle d'organisation économique, comparable à l'organisation «propriétaire» et non pas seulement une forme un peu particulière de l'organisation de production coopérative qui peut exister dans le monde de la recherche. En ce sens, nous nous opposons d'ailleurs à Horn [2000b], qui, distinguant plusieurs systèmes ou «mondes de production[48]», place le Libre dans ce qu'il appelle le «monde de la création», catégorie des organisations de production non-marchandes (c'est-à-dire où les incitations sont principalement non-financières, donc non marchandes), représentée typiquement par l'organisation de la recherche. Nous avons justement montré

[48]Il se réfère à la théorie des «mondes de production» développée par Salais et Storper [1993].

dans ce chapitre que le Libre était une organisation économique qui «fonctionnait» parce qu'elle permettait de créer des incitations financières qui poussaient des producteurs marchands à contribuer à la production de tels logiciels.

Cependant, le fait que cela soit une organisation économique «marchande» ne signifie pas qu'elle va s'imposer pour devenir l'organisation économique dominante, encore moins que cela soit souhaitable, car ce n'est peut-être pas la plus efficace économiquement. Au niveau de l'innovation, par exemple, on a souvent dit que l'organisation Libre n'était bonne qu'à produire des logiciels «génériques», au sens des médicaments génériques, pour reprendre l'expression de Kott [2001] : ce serait une «voiture balais» capable de reproduire des innovations quand elles sont devenues des standards mais pas capable d'en produire. Après avoir montré qu'on avait bien affaire à une organisation économique, il nous faut étudier son efficacité, en la comparant à l'étalon qu'est l'organisation dominante, l'organisation propriétaire. Ce sera l'objet du chapitre 3.

Troisième Chapitre

Efficacités comparées du Libre et du Propriétaire.

AU terme du premier chapitre, il est apparu que l'évolution du système de production de logiciel était possible, lorsque de nouveaux besoins apparaissaient, besoins auxquels l'ancien système ne répondait que difficilement. Nous avons aussi montré qu'actuellement, avec l'arrivée des réseaux, avec la place de plus en plus importante prise par les logiciels dans le fonctionnement des objets économiques, de nouveaux besoins de développaient, que nous avons qualifiés de «sur-mesure de masse». Les logiciels libres, parce qu'ils sont ouverts, parce qu'ils sont produits avec un certain niveau d'exigence technique, répondent mieux que les logiciels fermés à ces nouvelles exigences. Reste à savoir si cela est dû au système de production (le Libre), ou à certaines caractéristiques, imitables par l'organisation actuelle, comme l'ouverture du code.

Pour répondre à cette question, il a d'abord fallu s'assurer que le Libre était bien une organisation de production originale et non pas une simple évolution du système de la recherche (qui est aussi une organisation de production coopérative, mais qui ne concurrence pas le système propriétaire dans la production et la distribution des logiciels). C'est ce que nous avons montré dans le chapitre 2, en soulignant l'importance des organisations de production commerciales et des relations de service dans la stabilité de cette organisation.

Nous pouvons alors, et c'est ce que nous ferons dans ce chapitre, étudier les avantages de l'organisation Libre en tant que système de production et l'impact de son émergence sur l'organisation de l'industrie informatique. Dans la première partie de ce chapitre, nous montrerons que, tant au niveau de la capacité à générer des innovations et à en faire profiter le plus grand nombre, que surtout au niveau de la production de standards et de services (les éléments fondamentaux du sur-mesure de masse), le Libre présente des avantages indéniables sur les autres systèmes de production. Mais constater ces avantages ne suffit pas pour prouver le succès de ce système de production : il faut qu'il soit suffisamment plus efficace pour que les utilisateurs acceptent de changer les logiciels qu'ils possèdent déjà, parfois de s'adresser à d'autres distributeurs, ou au moins de voir les relations avec ceux-ci évoluer. Nous étudierons la diffusion du Libre dans la deuxième partie ; nous montrerons qu'elle se fera probablement par étapes, qui dépendront des caractéristiques des logiciels et des compétences des utilisateurs.

3.1 Efficacité comparée des organisations Libre et Propriétaire.

Il s'agit, dans cette partie, d'étudier l'efficacité comparée des organisations Libre et Propriétaire à répondre aux besoins des utilisateurs du «sur-mesure de masse». Fondamentalement, cette efficacité dépend du problème que rencontre un utilisateur : s'il existe un logiciel parfaitement adapté à ses besoins, qu'il soit libre ou propriétaire, il l'adoptera, s'il n'existe qu'une réponse, libre ou propriétaire, la question ne se pose même pas. Mais il ne s'agit pas ici de proposer une analyse besoins-solutions, analyse qui relève d'un travail de consultant. Il s'agit d'étudier les qualités économiques de ces systèmes de production et d'abord de définir les questions économiques qui vont nous permettre de les évaluer.

Il nous semble que les trois principales questions sont celle de l'incitation à développer des logiciels, celle de la «qualité» des standards produits (interopérabilité et évolutivité, concernant notamment leur adaptation aux besoins des différents utilisateurs) et enfin celle de la qualité des services fournis grâce à ces logiciels, services qui représentent une partie toujours plus importante des technologies d'utilisation en informatique. C'est à ces trois questions que nous allons maintenant nous intéresser.

3.1.1 Les incitations à produire du logiciel.

Nous avons vu dans le chapitre 2 que la production d'un logiciel passait par trois étapes : l'initiation d'un projet, la diffusion et l'amélioration du logiciel et enfin sa phase de maturité où les évolutions et les adopteurs se font de plus en plus rares. À chaque phase, on peut montrer qu'il y a des producteurs incités à développer des logiciels libres. Mais est-ce que ces incitations sont aussi importantes que celles de l'organisation propriétaire ? Quelle est l'organisation la plus efficace pour produire des innovations, c'est-à-dire de nouvelles technologies d'utilisation, pour les améliorer et les rendre accessibles à tous ? C'est ce que nous allons étudier ici, en reprenant successivement les trois étapes de la production d'un logiciel.

L'étape 1 de la production : le problème des incitations à innover.

Horn [2000b] identifie trois sources d'innovations : il peut s'agir d'une technologie initialement développée pour un besoin propre et qui s'avère utile à d'autres (ce sont les utilisateurs-innovateurs de von Hippel) ; il peut s'agir de produits issus de la recherche et qui trouvent des applications en tant que technologies d'utilisation ; il peut s'agir, enfin, de produits créés par un producteur, qui a identifié un besoin nouveau. Nous avons vu dans le chapitre précédent que la phase de l'innovation était une phase délicate dans la production d'un bien public et il est souvent dit que le Libre est moins efficace dans cette phase que l'organisation Propriétaire. Pourtant les produits développés en libre ne semblent pas plus mauvais, au niveau des choix techniques que les produits propriétaires : on critique souvent Linux qui repose sur des concepts développés pour Unix au début des années 70. Mais que dire alors de MacOS X, lui aussi développé sur une base Unix (BSD) et de Windows NT, qui reprend des concepts de VMS (système d'exploitation de Digital dans les années 70) et d'Unix.

En fait, parce que le but des producteurs de logiciels libres est proche de celui des producteurs de logiciels propriétaires : répondre à des besoins en terme de technologies d'utilisation[1], les deux systèmes, Propriétaire et Libre, sont assez proches en terme d'incitations à l'innovation, avec sans doute même un avantage pour le Libre. Nous allons le montrer en étudiant comment ces deux organisations puisent à chaque source de l'innovation.

La valorisation des produits de la recherche.

Il existe deux arguments qui plaident pour une plus grande efficacité de l'organisation libre dans sa capacité à absorber les concepts produits par la recherche publique.

Les développeurs de logiciels libres sont historiquement issus des milieux de la recherche et la recherche publique est toujours productrice d'une grande quantité de logiciels libres

[1] «Every good work of software starts by scratching a developer's personal itch» Raymond [1998a]. Edwards [2001] argumente de façon très convaincante ce point.
Si Hurd, le noyau développé par la F.S.F., ne fonctionne pas, au delà des possibles problèmes de management du développement, c'est peut-être parce qu'il était trop avancé techniquement. Si les technologies choisies étaient théoriquement les meilleures, elles n'étaient pas, pratiquement, les plus efficaces.
Le noyau Linux, a été critiqué pour son manque d'ambition technologique, mais adopté par les utilisateurs pour son efficacité à répondre aux besoins.

(Internet, en France CAML ou Scilab). Mais, après tout, les informaticiens qui travaillent dans les entreprises de production propriétaire de logiciels sont eux aussi issus du monde universitaire et peuvent eux aussi assurer le transfert des logiciels développés dans ce monde. Mais ce transfert se fait alors à un moment du temps et dans un sens seulement : de la production universitaire vers les producteurs industriels. Dans le cas de la production libre, et c'est le deuxième argument, il y a interaction permanente entre le monde de la recherche et le système de production des standards. Parce que le système de publication libre est plus proche dans son organisation de la culture universitaire, parce qu'il intègre de nombreux utilisateurs producteurs universitaires[2], il y a une interaction permanente entre ces chercheurs et les besoins exprimés par les utilisateurs. Ils peuvent donc plus rapidement répondre à une demande pour laquelle ils ont développé des outils. La recherche peut être aussi, consciemment ou non, orientée par les besoins des utilisateurs. Les producteurs propriétaires peuvent évidemment entretenir ce lien en développant des collaborations avec des universités, mais c'est plus coûteux et la collaboration se fait souvent autour de projets déterminés, ce qui limite l'apport. Surtout, chaque producteur est obligé de faire cet effort de collaboration, ce qui augmente le coût d'accès global à l'information, alors que dans l'organisation Libre, cette information est disponible, pour peu qu'on participe au développement.

Les liens entre développeurs de logiciels libres et recherche sont donc plus forts, plus réguliers et socialement plus efficaces car la diffusion d'informations est meilleure. Le Libre semble aussi plus efficace pour faire apparaître et pour diffuser des solutions développées originellement pour répondre à un besoin individuel.

L'«objectivation» d'une solution individuelle.

Il s'agit ici de transformer des produits initialement développés pour un utilisateur en un produit utilisable par plusieurs utilisateurs. La principale difficulté vient donc de la capacité des producteurs à identifier des besoins proches chez plusieurs utilisateurs et aussi à développer des logiciels qui sont construits dès l'origine pour pouvoir être la base d'une réponse standard à ces besoins. C'est ce que Horn a appelé l'objectivation du processus

[2]Voir, encore, l'interview de Stallman au journal du net (30/01/2001), (http ://solutions.journaldunet.com/0101/010130_it_fsf_stallman.shtml).

de production. C'est pour lui «la transformation d'un processus de production dédié au départ à un utilisateur particulier en méthodes et composants standardisés, utilisés pour la production de produits destinés à des utilisateurs divers» (Horn [2000b], p. 517). Et c'est un des atouts principaux du Libre que de permettre ce passage, plus facilement que l'organisation Propriétaire.

Nous avons vu que c'était le processus le plus courant dans le développement d'un logiciel libre : sans reprendre l'étude du chapitre précédent, il est bien souvent moins coûteux pour un utilisateur (marchand ou non) de publier son logiciel que de le conserver pour son usage propre (économie de maintenance, développement de nouvelles fonctionnalités, acquisition de renommée, etc.) Il existe aussi des entreprises, comme Alcôve, dont le métier est de faciliter ce transfert : elles adaptent des solutions libres aux besoins des clients, en développant des parties manquantes dans les logiciels libres, qui sont autant de logiciels libres et les publient sous licence libre. Enfin, surtout, le réseau permet de faire se rencontrer plus facilement des utilisateurs qui ont des besoins proches de ceux auxquels répond le logiciel et qui peuvent le faire évoluer.

Il est bien plus difficile de mettre en place ce mécanisme dans l'organisation Propriétaire. D'une part parce qu'on ne voit pas bien quelles raisons aurait un utilisateur à diffuser son logiciel en gardant les codes sources, sachant qu'il n'aurait alors aucun retour. Il lui est aussi difficile de vendre son logiciel, car ce n'est pas son métier. Les solutions hybrides, dans lesquelles le producteur garde le contrôle sur la diffusion de son logiciel et reste propriétaire des changements effectués par d'autres, n'apportent que peu d'avantage : elles diminuent l'incitation des autres utilisateurs à contribuer sans forcément augmenter la rémunération du producteur et en l'obligeant à une surveillance coûteuse de son logiciel. Enfin, traditionnellement, les entreprises qui développent des services ne sont pas celles qui proposent des logiciels ; ce sont deux métiers aux compétences différentes (voir Genthon [2001]). Donc il n'y a pas ce système organisé de transferts de la production «sur mesure» à la production «sur mesure de masse» qui caractérise le Libre.

Reste alors l'innovation que nous qualifierons d'«industrielle», c'est-à-dire le cas où l'innovation est réalisée par un producteur de logiciel, qui développe une nouvelle technologie d'utilisation.

L'innovation «industrielle».

C'est évidemment à ce niveau que l'avantage de la production propriétaire semble le plus important : lorsqu'une entreprise développe un logiciel, il est protégé et elle peut en obtenir une rémunération directe par sa vente, alors que la publication en libre ne lui assure que des revenus indirects, par la vente de services. Pourtant l'exemple de Digital Creation, entreprise qui a créé un logiciel libre, Zope[3], qui permet de développer et surtout de gérer un site Internet, amène à nuancer ce point de vue.

Deux choses peuvent donner un avantage à la production libre sur la production propriétaire. Tout d'abord, s'il existe une population potentielle d'utilisateurs-développeurs, la publication libre permet de recruter cette population et, partant, de diminuer les coûts de production. C'est surtout utile lorsque les logiciels ont une forte composante technique, qu'il faut les adapter aux configurations particulières des utilisateurs (ce qui génère des revenus grâce au service).

L'autre raison est que ces logiciels sont complémentaires à d'autres logiciels que l'on cherche à vendre (comme les navigateurs Internet complémentaires des logiciels serveurs). Dans ce cas, l'organisation Propriétaire, comme l'organisation Libre, permettent d'utiliser gratuitement ces logiciels, et apparaissent à peu près équivalente. Le seul avantage du Libre est que les utilisateurs, parce qu'ils peuvent modifier le logiciel, peuvent plus facilement l'adapter à leurs besoins propres.

Finalement, lorsque les revenus générés par la création d'un logiciel sont indirects, que l'objectif de la diffusion d'un logiciel est de créer une population potentielle de clients pour d'autres produits, le Libre apparaît comme aussi performant que le Propriétaire.

Une autre solution permet de s'assurer l'exclusivité de la commercialisation des logiciels en profitant des avantages de la production libre et diminue encore l'avantage de l'organisation Propriétaire dans la production des innovations industrielles : une entreprise peut publier la dernière version de son logiciel sous un format propriétaire et le libérer quelque temps (6 mois à 1 an) après. Mais cela ne fonctionne que si les utilisateurs considèrent comme important de disposer des dernières évolutions des logiciels, et il peut alors sembler plus efficace de leur vendre un service de mise à jour ; comme précédemment, le logiciel

[3]http ://www.zope.org/ pour le logiciel et http ://www.digitalcreation.com pour l'entreprise.

apparaît alors plus comme un outil permettant de vendre des produits complémentaires que comme le produit vendu.

Finalement, le meilleur système pour inciter à une innovation de la part des producteurs dépend donc de l'origine de leurs revenus : s'ils sont indirects, il peut y avoir des avantages à la production libre ou à des solutions hybrides, s'ils sont directs, alors la production propriétaire est la seule possible.

Synthèse.

Il est finalement difficile de confirmer la plus grande efficacité supposée de l'organisation Propriétaire par rapport à l'organisation Libre dans cette première étape.

Certes, une condition nécessaire à la publication libre est l'existence, parmi les utilisateurs de ce logiciel, d'un groupe de développeurs capables, soit d'initier sa production, soit d'y contribuer. Mais la diffusion d'Internet, qui permet de mettre en relation un plus grand nombre de personnes et de créer des groupes d'utilisateurs ayant des besoins proches à un niveau mondial, est très favorable au développement du libre[4] : cette masse critique d'utilisateurs développeurs est bien plus facile à atteindre et cela renforce les diffusions des travaux universitaires, plus facilement «appropriables» par l'organisation Libre. Certes, l'organisation Propriétaire reste la plus efficace pour inciter des entreprises à innover en produisant de nouveaux logiciels lorsque c'est la vente de ces logiciels qui va assurer la principale source de revenus. Mais, a priori favorable à l'organisation Propriétaire, cette première étape de la production semble en fait plus partagée : les logiciels initiés par des utilisateurs ou par des chercheurs sont plus facilement produits par l'organisation Libre, ceux initiés par les entreprises le sont (seulement !) parfois plus facilement par l'organisation Propriétaire. Avec le développement des relations de service et, plus généralement, du «sur-mesure de masse», avec la place toujours très importante de l'innovation universitaire dans la production de nouveaux logiciels, on peut même estimer que l'avantage de l'organisation Propriétaire dans cette première phase va devenir de plus en plus ténu, s'il ne s'est pas déjà transformé en désavantage.

Il est certainement plus facile de trancher dans la deuxième et dans la troisième étape

[4]Dang Nguyen & Phan [2000], chapitre 6.

du processus où l'organisation Libre apparaît bien comme plus efficace.

L'étape 2 de la production : gérer l'innovation incrémentale.

Il s'agit ici d'étudier la façon dont les deux systèmes permettent à un projet, une fois qu'il a été initié, de se développer, à de nouvelles fonctionnalités d'être ajoutées aux fonctionnalités initiales, etc. Bref, il s'agit de savoir comment, à partir d'une innovation de départ, l'innovation incrémentale est possible. C'est sans doute l'étape la plus importante dans la production du logiciel : nous avons vu dans le chapitre précédent que cette deuxième étape était celle où le logiciel se diffusait, mais aussi celle où la plupart de ses fonctionnalités étaient créées.

Le fait de diffuser les connaissances accélère le processus d'innovation et notamment le processus d'innovation incrémentale. Bessen et Maskin [2000] vont jusqu'à écrire qu'un système fort de propriété intellectuelle comme le brevet diminue la dynamique de l'innovation dans l'industrie du logiciel. Farrell [1989] (traduit par Zimmermann [1999]) affirme que «l'entrée sur le marché, la concurrence et l'innovation deviennent plus faciles si les concurrents potentiels peuvent se contenter d'améliorer un seul élément du système, dynamisant ainsi le marché de tels composants, plutôt que d'être obligés d'innover sur l'ensemble d'un système». Ce qui souligne le «caractère essentiellement cumulatif et incrémental de l'innovation» (Scotchmer [1991], traduit par Zimmermann [1999]), notamment quand il s'agit de logiciel. Nous allons montrer que le Libre est bien plus favorable à la logique cumulative que le Propriétaire[5].

L'innovation incrémentale, faiblesse principale de l'organisation Propriétaire ...

Deux systèmes de protection intellectuelle existent dans le cas du logiciel : le copyright (ou droit d'auteur), qui est le premier système de protection qui a existé et sur lequel repose aussi l'organisation Libre (qui détourne ses objectifs, mais utilise les principes légaux sur

[5]L'histoire de la protection intellectuelle du logiciel et ses conséquences économiques ont été étudiées par Zimmermann [1998], qui intègre la solution proposée par le logiciel libre dans Zimmermann [1999] ; l'Office for Technology Assessement (OTA) a aussi fait une étude économique du système de protection du logiciel aux États-Unis et s'est particulièrement intéressé aux conséquences de l'introduction du brevet comme système de protection. Nous nous appuierons sur ces travaux pour développer notre analyse.

lequel il repose, voir Clément-Fontaine [1999]), et le brevet, qui est autorisé depuis peu aux États-Unis et au Japon et que l'Europe se propose d'adopter.

Le brevet permet de garantir le monopole d'exploitation sur une idée, mais aussi sur ses développements. Mais il s'accompagne, théoriquement, d'une obligation de diffusion de la connaissance, par la publication des méthodes et des concepts, ce qui fait dire à ses défenseurs que c'est une bonne solution pour combiner incitation à l'innovation et diffusion de cette innovation[6].

En fait, cette publication ne permet pas vraiment la diffusion, pour plusieurs raisons. Tout d'abord, le pouvoir de monopole accordé sur les méthodes et les concepts peut être utilisé pour empêcher le développement d'innovations incrémentales sur des produits concurrents à celui du monopole, en n'accordant pas de licences aux concurrents. Les négociations pour accorder ces licences augmentent les coûts de transaction et font donc baisser la rentabilité de ces innovations marginales pour les concurrents, ce qui peut les décourager. Enfin, l'obligation de créer un organisme chargé d'étudier les demandes de brevet augmente encore les coûts sociaux d'un tel système[7].

De plus, pour obtenir un brevet, il faut justifier de la nouveauté de ce qui est breveté, d'un résultat industriel et d'une activité inventive. Tous les logiciels ne sont donc pas brevetables. Il faut combiner ce système à un autre si l'on veut pouvoir protéger l'ensemble des logiciels produits, y compris ceux qui ne font qu'appliquer une idée existante et donc pour lesquels il n'y a pas d'activité inventive. Le brevet ne se substitue pas au copyright,

[6] Breese [2000] développe ce type d'argumentation. On peut même imaginer que, si l'Europe autorisait la protection des logiciels par le brevet, cela impliquerait la publication des codes sources.

[7] On pourra consulter au sujet du brevet le rapport de l'OTA (OTA [1992]), ainsi que l'article de Kahin [1990], dont nous reprenons ici les conclusions, citées et traduites en français par Zimmermann [1998] :«Beaucoup [de programmes], loin des critères de nouveauté et d'originalité, apparaissent de nature conventionnelle ou évidente aux yeux des développeurs, qui craignent en conséquence de subir une pluie de procès individualisés, relatifs à telle ou telle procédure communément employée dans un logiciel.» (Zimmermann [1998], p. 104)

«Le problème se combine au fait que les suites logicielles modernes sont composées de milliers de processus brevetables, ajoutant chacun au risque d'empiéter sur un brevet en cours de protection. Dans la mesure où les applications logicielles sont interdépendantes et doivent être soigneusement articulées les unes aux autres, les développeurs informatiques risquent d'éprouver des difficultés à supprimer un processus qui ferait partie intégrante d'un programme original» Kahin [1990].

Au sujet de l'utilisation du brevet comme arme stratégique par les entreprises (protection vis-à-vis de la concurrence, protection complémentaire, sauvegarde des technologies futures, base d'alliances), on pourra aussi consulter Thumm [2000].

Pour l'OTA ([1992], p. 33), au vu des dysfonctionnements du système américain, la formation d'un collège d'examinateurs, suffisant en nombre et en compétences pour évaluer la nouveauté des logiciels soumis, et la formation d'une base de donnée permettant de connaître l'état de l'art dans ce domaine sont des présupposés indispensables à la mise en place d'un système de brevet efficace.

mais le complète, ce qui engendre un double système de protection, qui renforce encore les coûts de transaction (il faut savoir ce que protège le copyright, ce que protège le brevet avant d'essayer de développer des innovations incrémentales).

Le système de copyright semble plus favorable à l'innovation incrémentale et surtout à la concurrence par l'imitation car il ne protège pas les concepts (idées et principes, selon la loi française, Zimmermann [1998], p. 102), mais seulement la réalisation de ces concepts, c'est-à-dire le produit et le programme (code source). Parce qu'il permet de garder le secret sur le code source, ce système protège, de fait, une grande partie de la codification de la connaissance. Ce système n'est pas fait pour prendre en compte la partie innovante du développement du logiciel, en ce sens qu'il n'organise pas de façon active la publication et la diffusion des idées, des innovations, des nouvelles connaissances. L'innovation incrémentale se fait par défaut, parce que la protection est perméable, ce qui engendre une double inefficacité : les producteurs ne sont pas rémunérés pour les connaissances diffusées et il est coûteux pour les imitateurs et aussi pour les agents qui développent des innovations incrémentales de le faire car il faut franchir l'obstacle de la fermeture du code.
Des deux effets, c'est sans nul doute le second qui domine : même quand on dispose du code source des logiciels, ce qui facilite l'innovation incrémentale et la compatibilité (et on peut en disposer en France, pays où la «décompilation» des logiciels est autorisée en vue d'interopérabilité), au delà de quelques milliers de lignes de codes, un logiciel devient incompréhensible (aujourd'hui les logiciels ont plusieurs millions de lignes de code).

Par conséquent, les systèmes des licences hybrides, bien que donnant l'accès aux codes sources, ne permettent pas vraiment de remédier à ce manque d'incitation à l'innovation incrémentale. En effet, si la publication des codes sources favorise certes l'interopérabilité et donc la création de logiciels compatibles avec ce logiciel, elle ne signifie pas que la lecture du fonctionnement du logiciel soit aisée, donc qu'il soit effectivement possible de proposer des innovations. Rien n'empêche cependant un producteur de construire un logiciel en s'inspirant du système de fonctionnement de l'organisation Libre, mais, encore une fois, plus il attendra, plus le volume de code développé sera important et plus les utilisateurs

auront du mal à le comprendre lorsqu'il sera libéré. D'autre part, soit cette entreprise compte vendre ce logiciel et on ne voit pas pourquoi les utilisateurs paieraient deux fois le logiciel (une fois en contribuant à son développement, une fois en l'achetant), soit elle compte développer des activités de service autour de ce logiciel et dans ce cas, nous le démontrerons dans la troisième sous-partie, il est plus efficace de publier son logiciel sous une licence libre.

... et la force principale de l'organisation Libre.

Nous avons montré, au chapitre 2, que l'organisation Libre, parce qu'elle autorise quiconque à améliorer le logiciel, permet l'innovation collective. Mieux, la structure modulaire des logiciels, le noyau de développeurs chargés de garantir l'interopérabilité des différentes contributions, l'organisent. Rappelons que c'est autant la façon dont ces logiciels sont produits que l'ouverture du code qui permettent l'innovation incrémentale. Enfin, il existe un système d'incitation qui pousse les utilisateurs et les producteurs à proposer effectivement des contributions : les entreprises de services ont intérêt à contribuer à des projets libres (pour signaler leurs compétences) et leurs clients à ce que les adaptations qu'ils ont réalisées ou faites réaliser soient publiées en libre (pour assurer leur maintenance et donc leur pérennité).

Le Libre apparaît vraiment comme un système construit pour permettre, de façon plus efficace que le Propriétaire, l'innovation incrémentale. Cela ne doit pas surprendre qui se souvient que Stallman a initié le mouvement Libre justement parce qu'il ne pouvait pas améliorer les logiciels propriétaires qu'il était obligé d'utiliser[8]. C'est là son principal avantage, qui explique, nous le verrons, qu'il permet de construire de meilleurs standards, aussi bien en terme d'interopérabilité qu'en terme de qualité du logiciel produit (adéquation aux besoins, efficacité technique, etc.) Mais cette plus grande efficacité dans la construction des standards s'explique aussi par le fait que le Libre est plus efficace dans la troisième phase de la construction/diffusion d'un logiciel.

[8] Voir Stallman [1999].

L'étape 3 de la production : fausses innovations et maintenance du logiciel.

Cette troisième étape est celle où le logiciel est arrivé à maturité, c'est-à-dire qu'il répond aux besoins de la plupart des utilisateurs, que la plupart des innovations ont eu lieu. Cela ne veut pas dire qu'il n'y a pas de nouveaux adopteurs, mais leur nombre va en diminuant. Cela veut aussi dire que le fait qu'ils ne l'aient pas adopté plus tôt est dû à des raisons externes aux caractéristiques d'utilisation du logiciel (absence d'ordinateur, utilisation d'un autre logiciel, méconnaissance de l'existence de celui-ci, etc.) Le Libre permet de réduire le nombre des fausses innovations et améliore la prise en charge des logiciels qui n'évoluent plus.

Le problème des fausses innovations.

Comme nous le soulignions dans le chapitre précédent, le premier problème, dans cette phase, n'est pas tant d'inciter les producteurs à innover, que d'empêcher la prolifération de ce que nous appellerons des «fausses innovations», c'est-à-dire des évolutions dans les fonctionnalités qui n'apportent pas d'utilité supplémentaire aux utilisateurs, mais qui peut les obliger à changer de version du logiciel (nous pensons par exemple au changement dans les formats de stockage des informations, qui pousse à avoir la dernière version du logiciel pour pouvoir échanger avec les possesseurs de cette dernière version).

Le système Propriétaire produit «naturellement» cette tentation chez les producteurs : la phase 3 étant celle où le marché d'équipement se restreint, il leur faut développer le marché de renouvellement pour assurer leurs recettes. Le système Libre, beaucoup moins : les utilisateurs-développeurs n'ont aucun intérêt à y consacrer du temps ou de l'argent ; les entreprises productrices de services non plus car il n'est pas sûr que ces propositions soient acceptées par le noyau qui contrôle le logiciel[9] ou par les utilisateurs.

Le problème de la maintenance des logiciels qui n'évoluent plus.

Le deuxième problème, dans cette phase, est celui de la maintenance des logiciels qui n'évoluent plus. Dans le cas d'un logiciel propriétaire, la seule entreprise habilitée à effec-

[9]Sauf si elles contrôlent elles-même le logiciel. Mais il y a toujours le risque que des utilisateurs, mécontents de l'évolution de cette version, en proposent une autre, ou tout simplement refusent les nouvelles fonctionnalités.

tuer des changements, à faire évoluer un logiciel, est son propriétaire, qui a tout intérêt à vendre la nouvelle version de son logiciel (on est ramené au cas précédent). Les utilisateurs se plaignent d'ailleurs du coût de tels services de maintenance[10]. De plus, cette entreprise peut avoir disparu, ou, si le logiciel est vieux, ne plus savoir comment il fonctionne.

Le Libre améliore cette situation. Un logiciel libre étant gratuit, le coût de migration vers une nouvelle version est moins coûteux ; mais, cette migration n'est pas toujours possible (pensons aux systèmes embarqués dans les satellites). Il faut alors absolument trouver des informations sur la version du logiciel utilisée. Bien sûr, il y a le même phénomène de perte de la mémoire collective dans le cas du Libre et on peut s'inquiéter de ne pas avoir d'interlocuteur précis. Mais, d'une part, cette mémoire ne dépend plus d'une entité unique, mais est répartie parmi l'ensemble des utilisateurs-développeurs du logiciel (dont on a toujours une liste avec la version utilisée), ce qui augmente la probabilité de trouver une personne encore capable de comprendre le problème. D'autre part, la licence GPL permet à plusieurs entreprises de proposer des services de maintenance sur ces logiciels, augmentant ainsi la concurrence et faisant baisser les prix de tels services. C'est une fois de plus, un avantage du Libre sur les système hybrides.

Donc, dans cette phase aussi, le Libre apparaît plus efficace que l'organisation Propriétaire, même adaptée grâce aux licences hybrides.

Conclusion.

De cette analyse des mérites comparés des deux systèmes de production, trois points importants nous semblent émerger.

Tous d'abord, l'efficacité des incitations à innover, à initier le développement d'une nouvelle technologie d'utilisation dépend du public visé par cette technologie et de la façon dont chaque utilisateur peut se l'approprier. S'il n'existe pas de public potentiel d'utilisateurs-développeurs, c'est dans le système Propriétaire que les producteurs (marchands, par conséquent) trouveront les incitations à développer le logiciel. Si ce public existe, si l'utilisation demande la consommation de services supplémentaires, alors le Libre

[10]Voir le compte-rendu de la table-ronde d'«Autour du Libre 2000», http ://libre.enst-bretagne.fr/AL2000/.

sera sans doute le système le plus efficace.

À partir du moment où un logiciel a été initié, et ce sera la deuxième remarque, il apparaît que le Libre organise mieux son développement parce que c'est un système plus favorable à l'innovation incrémentale. Or, c'est sans doute le type d'innovation le plus important en informatique, surtout de la part des producteurs marchands de logiciels. Les innovations de produit sont toujours possibles, mais elles sont rares : peut-on dire que les versions successives de Solaris sont autre chose que des innovations incrémentales ? Bien sûr, Windows NT est une innovation importante par rapport à Windows 95, mais c'est aussi un logiciel inspiré du système VMS de Digital et il n'a pas réellement apporté de nouveaux concepts, de nouvelles fonctionnalités. Enfin, il a été assez dit que Linux n'était qu'un nouveau clone d'Unix pour ne pas y revenir. Il y a bien des innovations : Internet est une innovation, comme l'a été en son temps l'interface graphique, bien qu'on puisse aussi défendre que le premier est une évolution des protocoles de réseau existants et remarquer que l'innovation apportée par Apple avait été inventée quinze ans plus tôt par Xerox. Mais ce sont souvent des innovations produites dans des laboratoires de recherche (ce qui renforce d'ailleurs l'attractivité du modèle Libre, plus efficace pour intégrer ce type d'innovations).

On pourrait enfin imaginer que les logiciels produits par l'industrie soient initiés dans un système propriétaire et deviennent ensuite libres. Il ne nous semble pas, et c'est la troisième remarque, qu'il y ait là un véritable gain d'efficacité : soit ces logiciels se prêtent à la production de services, à beaucoup d'innovations incrémentales et intéressent des utilisateurs-développeurs, et alors il est plus efficace de les développer dès le départ en libre, soit la mise en libre n'apportera aucun bénéfice supplémentaire à l'inventeur. De même, la solution des licences hybrides ne permet jamais à l'organisation propriétaire de rattraper l'organisation Libre lorsque celle-ci s'avère plus efficace.

Mais étudier seulement, comme nous l'avons fait jusqu'à maintenant, la production d'un logiciel serait tout à fait insuffisant : souvent plusieurs solutions sont en concurrence. Il faut se poser la question de l'efficacité de ces deux systèmes à organiser cette concurrence, soit en sélectionnant la solution la meilleure, qui deviendra donc un standard, soit en

permettant à plusieurs solutions de coexister, voire de cohabiter (si elles sont amenées à entrer en relation). Bref, il faut étudier comment ces systèmes organisent les processus de standardisation.

3.1.2 La construction de standards.

Nous avons vu au chapitre 1 l'importance de ces phénomènes de standardisation dans l'évolution de l'industrie informatique. Si l'industrie du logiciel est dominée par les standards, tous les logiciels ne sont pas ce que l'on entend communément par «standard», c'est-à-dire des logiciels utilisés par tous : soit ils ne sont utiles qu'à des clubs très fermés d'utilisateurs, soit plusieurs logiciels coexistent pour rendre les mêmes services. Cela n'empêche pas que se posent, pour ces logiciels aussi, des problèmes de standardisation. En effet, en économie, la définition que l'on donne au mot standard est assez large : nous avons donné au chapitre 1 celle de David [1987], nous donnerons ici celle de Katz & Shapiro [1985] (qui proposent d'ailleurs quasiment la même définition que Tirole, [1995], p. 411) : «[...] products for which the utility that a user derives from consumption of the good increases with the number of other agents consuming the good». Les problèmes de standardisation concernent donc tous les logiciels qui sont utilisés par plus d'une personne, qui dépassent l'étape 1 du processus de production.

Les résultats des études théoriques des processus de standardisation, que nous présenterons rapidement dans le premier paragraphe, nous amèneront à étudier d'abord la façon dont ils sont construits et ensuite la façon dont un standard qui s'est imposé peut évoluer, puis être dépassé lorsqu'il ne convient plus aux besoins des utilisateurs.

Quelques résultats économiques au sujet des standards.

Finalement, les définitions d'un standard que nous avons prises s'appuient toutes sur les caractéristiques de son adoption : un standard est un standard car il est plus intéressant pour les nouveaux adopteurs de le choisir que de choisir la technologie concurrente et ce parce que, justement, il a déjà été choisi. Et il y a standardisation parce qu'il y a des «rendements croissants d'adoption» (pour reprendre la terminologie d'Arthur ([1987],

[1988 a,b], [1989])[11]).

Mais dans tous les cas, on va aboutir à un standard, à une «technique, un produit qui est utilisé par une forte proportion d'agents, étant donné le nombre d'utilisateurs potentiels» (Foray [1996], p. 257), et le chemin qui y mène a un impact sur ce que va être ce standard.

Arthur distingue cinq types de rendements croissants d'adoption d'une technologie ou d'un produit. Foray [1989] en a proposé un classement, que nous allons reprendre ici (mis à part le fait que nous les avons regroupés en quatre catégories au lieu de cinq) :

- l'apprentissage par l'usage et les rendements croissants d'information (voir Rosenberg [1982]) : plus une technologie sera utilisée, plus on saura s'en servir et donc plus on sera efficace pour l'utiliser, que ce soit au niveau individuel (apprentissage par l'usage) ou au niveau collectif (rendements croissants d'information) ;
- les externalités de réseau, dont nous avons déjà parlé : ce sont, pour nous, tous les systèmes d'échange d'informations entre utilisateurs des biens ;
- les économies d'échelle en production, qui font qu'il est moins cher de produire le même bien pour tous que d'en produire deux différents. Nous avons vu que c'était le cas du bien public logiciel ;
- les «interrelations»[12] technologiques, qui font que l'adoption d'un bien dépend aussi des biens «liés» à celui-ci. C'est particulièrement important dans le cas des logiciels, puisqu'un logiciel ne fonctionne jamais seul, mais avec une machine et avec d'autres logiciels.

Les rendements croissants d'adoption interviennent à des moments différents de l'histoire d'une technologie, pour construire ou maintenir sa supériorité. Foray ([1989], p. 29]) considère qu'au début du processus, l'apprentissage par l'usage et les externalités de réseau dominent. Viennent ensuite les économies d'échelle en production, les rendements croissants d'information et les interrelation technologiques, qui augmentent la supériorité d'une

[11] Ils sont aussi appelés «effets» ou «externalités de réseau», nous l'avons vu dans l'introduction. Nous préférons réserver, à partir de maintenant, ces termes aux interactions qui peuvent exister entre les différents adopteurs quand ces adopteurs connaissent les décisions des autres adopteurs, c'est-à-dire quand ils sont reliés par un réseau d'information. Nous suivons en cela Katz & Shapiro, Besen & Farrell [1994] et surtout Liebowitz & Margolis [1994].

[12] Néologisme utilisé par Foray [1989] ; on aurait pu lui préférer «relation» ou «interaction», mais, afin de conserver le lien avec la référence initiale, nous conserverons le terme «interrelation».

technologie, mais qui sont également dépendants de sa diffusion. Enfin, le «lock-in», c'est-à-dire la persistance d'une technologie, provient à nouveau des coûts individuels et collectifs de changement, c'est-à-dire de l'accumulation des apprentissages[13] et des interrelations technologiques. On peut facilement faire le parallèle avec les différentes étapes de la diffusion de logiciels (ce qui est normal, puisque c'est un bien public, donc soumis à de nombreux rendements croissants d'adoption), la différence étant qu'ici, plusieurs technologies sont en concurrence, comme on peut le voir dans le tableau 3.1.

Tableau 3.1 — Correspondance entre processus de production et de standardisation des logiciels.

	Construction d'un logiciel libre	Construction du standard
Phase 1	Expression d'un problème Construction d'une solution par un noyau 1^{er} noyau d'utilisateurs.	Différentes expressions d'un même problème Différents groupes de développement, différentes entreprises, travaillent isolément Proposition de plusieurs solutions
Phase 2	Recrutement des utilisateurs Structuration du projet (module) Développement des modules (nouvelles fonctionnalités)	Recrutement de la plus grande population possible Concurrence par l'innovation Imitation/copie des fonctionnalités les plus appréciées.
Phase 3	Maintenance de la cohérence du projet Diminution du nombre de développeurs Si besoin, organisation du transfert des compétences vers un nouveau projet.	Standard dominant Évolution de plus en plus lente du standard Possible expression d'un besoin de dépassement du standard.

Le processus de standardisation en lui-même commence au passage de l'étape 1 à l'étape 2, quand il s'agit de proposer la solution à d'autres que les développeurs initiaux, mais surtout de convaincre de nouveaux utilisateurs d'adopter cette solution plutôt qu'une autre.

Pour Foray ([1989], p. 21), ce processus de standardisation présente quatre caractéristiques importantes : le résultat (quelle sera la solution vainqueur) n'est pas prévisible au début du processus même si on sait qu'il y aura un vainqueur ; ce processus devient, au bout d'un certain temps, irréversible (effet de «lock-in») ; il est possible que le processus soit inefficient socialement, en éliminant une technologie ou un produit qui,

[13] «The improvement included in learning by using plays an important role in the decision to adopt new technologies» (Rosenberg [1982]).

avec le recul, apparaît comme potentiellement supérieur[14] ; enfin, ce résultat dépend des événements qui interviennent au début du processus et, plus généralement, de l'ensemble des événements passés.

C'est donc la capacité des deux organisations (Libre et Propriétaire) à assurer que le standard sélectionné est le meilleur possible qu'il nous faudra d'abord évaluer. Ce qui nous amènera ensuite à discuter du dépassement d'un standard, qui peut devenir nécessaire si les utilisateurs n'en sont plus satisfaits, car les améliorations possibles ont été épuisées, parce que le progrès technologique permet d'en créer un meilleur, ou tout simplement parce qu'il s'avère ne pas être le meilleur.

L'efficacité du processus de standardisation.

L'efficacité du processus de standardisation dépend de plusieurs facteurs : la qualité moyenne des solutions proposées, l'efficacité du processus d'innovation incrémentale, la facilité pour les utilisateurs de passer d'un standard à un autre.

La qualité moyenne des produits.

Il existe rarement de solutions uniques à un problème de programmation, donc pas souvent de «meilleure» solution initiale. Il s'agit plus d'évaluer la qualité moyenne des contributions et surtout la qualité moyenne des standards produits par les organisations.

Nous avons montré, dans le chapitre 1, que dans l'organisation Libre le niveau d'exigence technique supérieur et l'existence d'une population d'utilisateurs-développeurs, capable de traquer les erreurs, permettaient de proposer des logiciels de qualité supérieure à ceux de l'organisation Propriétaire. A priori, cette organisation peut être mise en place autour de logiciels propriétaires, si le code source est ouvert[15]. Mais, nous venons de le voir, il nous semble que le processus d'innovation incrémentale est meilleur dans l'organisation Libre. Nous avons expliqué cette plus grande qualité moyenne par le fait que l'ouverture du libre permet une plus grande division du travail chez les producteurs de technologies d'utilisation, donc une plus grande spécialisation de ces producteurs, chacun

[14] Ce problème a été illustré par Cowan [1988] avec l'exemple de la concurrence dans les technologies du nucléaire civil.

[15] Comme le font remarquer Lerner & Tirole [2000], en prenant l'exemple du logiciel de messagerie Eudora.

développant le module qui l'intéresse (chapitre 2) et par une meilleure gestion de l'innovation incrémentale (ce chapitre). Donc, en moyenne, les produits proposés par le Libre sont de meilleure qualité, ce qui laisse présager un standard techniquement de meilleur qualité. Mais, c'est surtout dans le processus de diffusion que l'organisation Libre apparaît supérieure.

La qualité du standard qui émerge.

Le risque d'aboutir à un standard technologiquement sous-optimal augmente avec la vitesse de standardisation parce que le processus de sélection de la meilleure offre n'a pas le temps de se faire et parce que les idées, les avancées contenues dans un projet n'ont pas le temps d'être incorporées dans un autre projet (David [1985], Arthur [1989])[16].

Or, l'existence de forts rendements croissants peut pousser à cette standardisation rapide. La diffusion d'Internet, qui augmente les effets de réseau, entraîne des effets de standardisation de plus en plus rapides. Une bonne illustration est le passage de Word 6 à Word 95 puis à Word 2000, passages imposés dans des temps très courts à cause de l'augmentation de la fréquence des échanges des données.

Lorsque la concurrence se situe au niveau des protocoles d'échange ou des formats de fichier, on peut développer des logiciels de traduction, qui permettent de faire cohabiter différentes solutions et donc de prolonger le processus de sélection. Lorsqu'il s'agit de produits, les fonctionnalités sont ajoutées en développant de nouvelles extensions au logiciel initial. Les trois facteurs favorisant cette concurrence par l'imitation ou par l'«hybridation», sont le respect des standards (ou des normes) d'échange entre les logiciels, la construction modulaire des logiciels et la disponibilité du code source.

Dans l'organisation Propriétaire, il y a un vrai problème de passager clandestin : chaque producteur est intéressé pour récupérer les nouvelles fonctionnalités proposées par ses concurrents, mais va protéger les siennes, qui sont la base de son avantage concurrentiel. Dans l'organisation Libre, le succès d'une entreprise ne vient plus du fait que sa technologie lui apporte un contrôle monopolistique sur un marché. Il vient au

[16] Dans ce paragraphe, on va globalement retrouver les mêmes arguments que dans le paragraphe sur l'innovation incrémentale (puisqu'il s'agit de concurrence entre produits en développement). La différence tient au fait qu'il existe plusieurs produits en concurrence.

contraire de ce que sa solution est adoptée par les concurrents, ce qui permet d'augmenter sa renommée, de diffuser une solution dont elle contrôle les évolutions, donc d'augmenter son pouvoir sur le marché de l'expertise. Elle a alors intérêt à développer les interfaces de compatibilité et à adopter des protocoles et des langages standards. Souvent, de plus, le noyau du logiciel, qui garantit cette interopérabilité, est contrôlé par les utilisateurs (comme c'est le cas pour Linux) ou par un comité, qui ont pour objectif que le logiciel reste indépendant du contrôle d'une entreprise. Enfin, la licence GPL, qui garantit l'ouverture du code source, la facilite aussi. Zimmermann [1999] insiste d'ailleurs sur l'importance de l'ouverture des interfaces de compatibilité pour assurer une concurrence plus efficace dans le logiciel et sur l'avantage de l'organisation libre dans ce domaine (c'est une des conséquences d'une meilleure organisation des innovations incrémentales par l'organisation libre). Donc, si l'on pouvait isoler les deux organisations, les standards produits par le Libre seraient en général de meilleure qualité.

Par contre, lorsqu'un logiciel propriétaire se trouve en concurrence avec un logiciel libre, on peut penser que le logiciel propriétaire est avantagé : son producteur peut récupérer un morceau de code pour incorporer une nouvelle fonctionnalité et surtout, il peut développer des formats de données qui ne seront pas imitables par son concurrent libre. La licence GPL permet, en partie, de protéger les projets libres de cette imitation : si le producteur du logiciel propriétaire récupère une partie du logiciel libre et l'incorpore dans son propre logiciel, il peut être obligé de distribuer ce logiciel modifié sous GPL (c'est ce qu'on appelle l'effet «contaminant» de la GPL, qui oblige à redistribuer un logiciel utilisant du code protégé par la GPL en GPL). C'est une faiblesse de l'organisation Libre, faiblesse d'autant plus atténuée que la dynamique d'innovation est forte car le producteur du logiciel propriétaire aura d'autant plus de mal à «suivre le rythme» de l'innovation.

Une autre faiblesse de l'organisation Libre est que la multiplication des projets peut faire qu'aucun groupe de développement n'atteint la taille critique (en terme de capacité de production ou en terme de population d'utilisateurs). C'est le problème du «split» dont nous avons parlé au chapitre 2. Cependant, le respect des standards de développement rend relativement facile l'intégration d'un module développé pour un projet à un projet

concurrent. Les développements sont donc plus rarement perdus que dans un système Propriétaire où les phénomènes d'incompatibilité rendent plus difficile cette transition. De plus, ce problème de concurrence entre projet existe aussi dans le système Propriétaire : à partir du moment où plusieurs offres se font concurrence, il y a, là aussi, duplication des investissements. Mais il est vrai que l'implication d'entreprises dans les projets libres, qui peuvent payer des développeurs pour atteindre cette taille critique, pour les faire avancer, ou même les sponsoriser en faisant de la publicité pour ces projets auprès des utilisateurs potentiels, est un appui important pour atténuer ce qui pourrait apparaître comme une des principales faiblesses du Libre. C'est surtout vrai pour les projets en concurrence avec un projet propriétaire.

Finalement, il apparaît que si le Libre produit des standards de meilleure qualité, il n'est pas sûr que ce soit l'organisation la mieux adaptée pour les imposer. Cependant, nous nous sommes pour l'instant placés plutôt du côté de l'offre, supposant que les utilisateurs étaient indifférents entre une offre libre et une offre propriétaire, à qualité égale. Or, les anticipations des utilisateurs sur le devenir du produit sont importantes dans le choix du sentier de standardisation. Et ceux-ci peuvent vouloir favoriser les offres libres.

Le Libre favorise la réversibilité des choix des utilisateurs.

Lors d'un processus de standardisation, il y a un risque pour un adopteur de choisir l'offre perdante et de devoir à terme changer de programme et, parfois, de perdre les données qu'il a produites avec. Ce problème des «orphelins furieux» (David & Bunn [1988], Dalle & Foray [1998b]) peut ralentir le processus de diffusion : les adopteurs attendent de savoir quelle va être la technologie concurrente pour s'engager (Katz & Shapiro [1986]). Le fait que les logiciels propriétaires soient fermés rend difficile le développement des outils de compatibilité, donc plus risqué ce choix.

Par contre, le fait que les logiciels libres respectent des standards, que l'on puisse inspecter le code pour savoir comment ils produisent ces documents, permet plus facilement ce changement. De plus, comme nous l'avons montré dans la sous-section précédente, parce que ces logiciels produisent des standards qui sont plus proches de normes que de standards

de fait, les utilisateurs sont moins dépendants des producteurs initiaux, ce qui leur assure une garantie meilleure sur la maintenance de leurs solutions.

Conclusion.

Finalement, il apparaît que le Libre permet de produire de meilleurs standards que le Propriétaire. Cependant, lorsqu'il y a une concurrence entre une offre libre et une offre propriétaire, l'ouverture du libre (qui facilite l'imitation) et l'absence de soutien des entreprises semblent désavantager l'offre libre. Pourtant, ce désavantage peut se transformer en avantage quand le respect des normes, la pérennité d'une solution sont des points importants dans les critères de choix des utilisateurs. C'est le cas pour les logiciels à forte interrelation technologique, les logiciels de réseau et les logiciels qui doivent pouvoir évoluer dans le temps.

À l'anticipation des utilisateurs à un niveau individuel, peut aussi s'ajouter l'anticipation collective (par exemple au niveau de l'État) sur l'offre qui permet de proposer le meilleur standard à un instant du progrès technique, mais aussi de préserver le futur. Il faut en effet s'assurer que, si une offre supérieure apparaît, elle puisse remplacer le standard existant. Le Libre semble aussi plus efficace pour garantir cette évolution et le dépassement d'un standard existant.

Le dépassement du standard.

La théorie économique (Farrell & Saloner ([1985], [1988]), Postrell [1986]) nous enseigne que lorsque l'adoption est due à des effets de réseau ou, dans le cas de l'industrie du logiciel, à des interrelations technologiques, il est possible de changer de standard en construisant des interfaces de compatibilité ou en permettant la coordination des agents[17]. Si la standardisation est due à la «construction» de la technologie (apprentissages par l'usage, économie d'échelle), Arthur [1988a] estime que ce changement est difficile : pour proposer une offre comparable, il faut surmonter le déficit d'information sur la technologie, il faut compenser les coûts de changement et anticiper sur les bénéfices distribués par les

[17] C'est une spécificité de l'industrie du logiciel que l'interrelation technologique soit du côté des effets de réseau. En effet, comme nous l'avons expliqué au chapitre 1, les biens liés fonctionnent en échangeant des informations qui sont définies par des protocoles et des logiciels, qui ne sont pas différents des protocoles d'échanges entre logiciels concurrents.

effets d'échelle. C'est effectivement très coûteux, pour un résultat incertain.

Si l'on compare maintenant nos deux modèles économiques, on peut d'abord constater un point qui reste valable : il est plus difficile d'assurer la compatibilité à un système dont on ne connaît pas le fonctionnement ; on reste proche de ce qui se passe pendant le processus de standardisation, et l'avantage va alors au Libre. Les bénéfices distribués grâce aux effets d'échelle peuvent faire baisser le prix du logiciel ou celui du service. Dans le cas de l'organisation Libre, où le logiciel est gratuit, tout se passe comme si les effets d'échelle sont complètement redistribués en ce qui concerne le coût de production du logiciel, et ne peuvent jouer sur le coût (et donc le prix) du service ; dans le cas de l'organisation Propriétaire, il n'est pas sûr que ces effets soient complètement redistribués, puisque le logiciel n'est pas obligatoirement gratuit. Or la gratuité du logiciel facilite la transition, que ce soit vers un logiciel libre ou un logiciel propriétaire. On l'a bien vu avec la concurrence entre Internet Explorer et Netscape car un essai ne coûte rien, ou plus exactement n'implique pas de dépenses. Donc, une fois de plus, l'avantage est aux logiciels libres, au moins tant que le prix des logiciels propriétaires est non nul (et s'ils sont gratuits, nous avons montré qu'il était souvent préférable qu'ils soient produits sous licence libre).

Par contre, l'apprentissage par l'usage, c'est-à-dire les efforts que les utilisateurs ont fait pour apprendre à se servir d'un logiciel, restera toujours un frein au changement de logiciel. Et nous ne voyons pas en quoi le Libre facilite (ou rend plus difficile ce changement), sinon qu'il est sans doute plus facile de développer des composants qui adaptent l'interface des utilisateurs pour faire fonctionner le nouveau logiciel libre comme l'ancien logiciel libre (car on peut parfois reprendre des morceaux de code, et notamment l'interface).

Finalement, même s'il semble plus facile de changer de standard quand celui-ci est libre, cela reste une opération coûteuse et donc sans doute relativement rare.

Conclusion.

Cette étude de l'efficacité des organisations Libre et Propriétaire à créer des standards a montré que, socialement, il était préférable que les standards soient des standards libres car

ils sont de meilleure qualité et parce qu'on peut les faire évoluer plus facilement. Mais que, parce que les logiciels propriétaires sont souvent plus soutenus, et qu'ils bénéficient de plus d'investissements de la part des producteurs au début du processus, les caractéristiques de ce processus de standardisation font qu'il y a un risque que ce soit ces logiciels qui s'imposent.

Ce risque est d'autant plus grand que le besoin d'interopérabilité et de service est faible : les utilisateurs sont alors moins attentifs au fait d'avoir le plus de contrôle possible sur l'évolution du logiciel dans le futur, donc à son indépendance vis-à-vis de son constructeur. Cette remarque nous amène à nous intéresser à un point qui a été approché plusieurs fois dans ce chapitre, mais que nous n'avons pas encore traité : le problème de l'organisation des relations de services, qui caractérise, avec la construction des standards, l'économie du sur-mesure de masse.

3.1.3 La qualité des services construits.

Cette partie va être relativement courte, car la production des services a été étudiée dans le chapitre 1 pour l'organisation Propriétaire et dans le chapitre 2 pour l'organisation Libre. Mais il nous a semblé important de les rapprocher ici, afin de mieux mettre en lumière les avantages de l'organisation Libre dans ce domaine.

La production de service est caractérisée par l'existence d'«asymétries d'information», qui rendent difficiles le choix d'un fournisseur et l'évaluation du travail effectué. Et nous avons constaté, avec De Bandt [1995], Dréan [1996] et bien sûr Horn [2000b], que ce marché, et particulièrement le marché des services informatiques, fonctionne mal, engendrant une profonde insatisfaction des utilisateurs des outils informatiques, notamment sur le service après-vente des logiciels[18].

Le Propriétaire, une organisation peu adaptée au service.

Les arguments que nous avons développés pour expliquer ce fait sont les suivants : du point de vue technique, il est plus difficile pour une entreprise de service d'intervenir sur

[18]Nous rappellerons ici l'enquête de satisfaction réalisée chaque année par l'hebdomadaire 01 Informatique auprès des grands comptes français (enquêtes de 1998, numéro 1521, de 1999, numéro 1566 et de 2000, numéro 1612).

des logiciels dont elle ne maîtrise pas la conception et qu'elle ne peut pas faire évoluer car elle n'a pas accès au code source. Rendre le code source disponible (comme c'est le cas pour les licences hybrides) facilite finalement assez peu ce travail, à cause de la taille de ces logiciels, à cause du fait qu'ils ne sont pas toujours écrits par module (et qu'il est donc difficile de savoir où il faut intervenir).

D'un autre côté, si les producteurs d'un logiciel vendent ce bien, ils n'ont pas forcément intérêt à répondre aux demandes de modifications des utilisateurs, mais plutôt à incorporer cette modification dans la version suivante du logiciel, moyen simple de revendre le même logiciel à leur client. Le fait que ces producteurs aient un contrôle monopolistique sur le logiciel lie ses utilisateurs à ses producteurs, et ce d'autant plus que le changement de logiciel est coûteux. Posséder une clientèle captive n'incite pas à fournir le meilleur service possible à cette clientèle.

Au contraire, l'organisation Libre semble avoir été construite pour faciliter relations de service et ne peut d'ailleurs fonctionner que quand elles sont importantes.

Le Libre, une organisation construite autour de la relation de service.

La production modulaire, combinée avec l'ouverture des codes sources permet à chacun d'intervenir et de faire évoluer les logiciels, à condition, bien sûr, d'en avoir les compétences techniques. Comme ces compétences ne sont pas celles de tous les utilisateurs, il y a (comme pour les logiciels propriétaires) des opportunités de marché pour des entreprises de service. Mais, à la différence du Propriétaire, le Libre est construit pour assurer que le service rendu sera le meilleur possible.

Le Libre fait baisser les asymétries d'information sur les compétences de ces entreprises : parce que leurs contributions sont publiques, ces entreprises peuvent prouver leurs compétences. D'autre part, quand les logiciels sont protégés par une licence comme la GPL, il sont inappropriables ; les utilisateurs, s'ils sont toujours plus ou moins captifs des logiciels qu'ils utilisent, le sont beaucoup moins des entreprises qui les leur fournissent : soit ces logiciels sont contrôlés par un noyau indépendant des entreprises, et alors plusieurs entreprises de service sont en concurrence ; soit ils sont contrôlés par une entreprise, mais

son monopole est «contestable» : si l'entreprise n'assure pas un bon niveau de service, les utilisateurs-développeurs ou éventuellement une autre entreprise risquent de proposer une version concurrente du logiciel.

Ce qui veut dire aussi que, pour que le Libre fonctionne, il faut qu'existent ces opportunités de service, sinon, il ne peut pas y avoir d'entreprises productrices (quels seraient leurs revenus ?). Cette absence peut rendre difficile l'accès de ces logiciels aux utilisateurs sophistiqués ou naïfs. En effet, les adaptations à réaliser pour cela prendront du temps et plus ces logiciels sont à forts rendements croissants d'adoption, plus grand est le risque qu'un concurrent propriétaire, sponsorisé par son producteur, s'impose avant que le noyau des utilisateurs-développeurs ait eu le temps de développer ces fonctionnalités.

Bien sûr, cela ne veut pas dire qu'utiliser un logiciel libre assure que la relation de service sera efficace, satisfaisante pour les deux parties : il faut toujours que le producteur comprenne les besoins de son client et les traduise en terme de technologies d'utilisation. Simplement, si le Libre n'apporte pas beaucoup d'avancées au niveau de la compréhension (mis à part le fait que les utilisateurs peuvent essayer les logiciels, sans les payer), il facilite le travail de traduction. C'est déjà un point important et c'est sans doute, avec la meilleure efficacité du processus de standardisation, la principale qualité du Libre.

3.1.4 Conclusion de la partie.

Dans cette partie nous nous sommes interrogés sur l'efficacité respective des modèles Libre et Propriétaire dans la production des technologies d'utilisation, logiciels et services.

Globalement, le Libre est apparu plus efficace, parce qu'il assure une meilleure diffusion des innovations développées par les chercheurs et les utilisateurs, parce qu'il assure une meilleure standardisation et parce qu'il permet de développer des services de meilleure qualité.

Un second résultat important est que les solutions hybrides, supportées par des licences «semi-libres», comme les licences de SUN ou d'Apple n'apportent pas une grande amélioration à l'organisation Propriétaire et construisent des modèles qui sont, suivant les logiciels, moins efficaces que le modèle Libre ou que le modèle Propriétaire «traditionnel».

Au delà de la meilleur performance globale du Libre, il nous est apparu que l'efficacité des différents modèles variait beaucoup, selon le type de logiciels et selon les caractéristiques des utilisateurs. Une condition nécessaire à l'utilisation de l'organisation Libre est qu'il existe une population d'utilisateurs-développeurs intéressée à participer au développement du logiciel. Ce sont, sans surprise, les situations où l'ouverture du code, l'utilisation de standards d'échange (qui facilite l'interopérabilité), la modularité (qui facilite l'optimisation des performances) sont importants, où la construction des solutions combine logiciels et services d'adaptation, que le Libre est le plus efficace. Bref, le Libre apparaît comme l'organisation du monde du «sur-mesure de masse», monde dont les logiciels caractéristiques sont des logiciels standards adaptables et à adapter à chaque utilisateur.

Ce qui veut dire que le Libre n'éliminera pas le Propriétaire, s'il se diffuse, mais diminuera son importance, le reléguant du statut d'organisation dominante à celui d'organisation spécialisée dans certaines tâches où ses spécificités font que c'est la plus efficace : essentiellement les tâches où la seule source de financement du développement du logiciel est la vente de celui-ci, parce que les services d'accompagnement ou de maintenance sont faibles ou nuls. Ce résultat n'est pas surprenant. Nous avons montré, dans le chapitre 1, que ce ne sera pas la première fois qu'une organisation économique en remplace une autre comme organisation dominante pour la production des logiciels, mais que quand cela se produit, elle ne fait jamais disparaître l'ancienne organisation, mais limite simplement son champ d'application.

Ce changement se fait toujours dans la durée parce qu'il faut parfois remplacer le standard existant, mais aussi faire évoluer les critères d'évaluation des produits proposés. Comme l'explique Callon [1991, 1992], ces conventions, ces choix technologiques construisent des réseaux «technico-économiques» les rigidifient aussi, limitant définitivement l'espace des possibles et rendant difficile l'adaptation des acteurs de ces espaces aux contraintes d'un nouvel espace technico-économique (rendant donc aussi difficile son développement). Les «mondes de production» décrit par Horn [2000b], mais aussi les grappes technologiques de Zimmermann [1995b] ou les régimes de concurrence de Genthon [1998] s'apparentent aux périodes où un tel réseau est dominant.

Abernathy & Utterback [1978], en s'appuyant notamment sur l'exemple de l'indus-

trie automobile, ont expliqué comment le passage d'un réseau à un autre se produisait : après une période de foisonnement des solutions techniques, des offres de produits et des organisations industrielles, une nouvelle organisation industrielle et les produits qu'elle propose finissaient par s'imposer. Les évolutions technologiques constatées, qui entraînent une évolution de la demande et, en réponse, un foisonnement des types d'organisation de la relation client-fournisseur (illustré par la multiplication des types de licences sur les logiciels), montrent que nous sommes actuellement au premier stade d'une nouvelle évolution. Reste à prouver que le Libre sera suffisamment plus efficace que le Propriétaire pour surmonter les effet de lock-in technologiques et organisationnels, dûs au fait que l'organisation Propriétaire existe depuis longtemps, que ses produits sont déjà diffusés et qu'elle «contrôle» la plupart des standards, et être la nouvelle organisation industrielle.

3.2 Diffusion des logiciels libres, diffusion du Libre : différentes stratégies pour différents produits.

La sélection des organisations industrielles se fait sur les produits, c'est-à-dire qu'une nouvelle organisation industrielle va s'imposer car elle est capable de produire mieux de nouveaux produits qui répondent à une nouvelle demande (Abernathy & Utterback [1978], Zimmermann [1995]) ; l'exemple de l'industrie du logiciel développé dans le chapitre 1 illustre bien ce fait. La diffusion du Libre dépend donc de la diffusion des logiciels libres. Les enquêtes montrent bien que les entreprises adoptent des logiciels libres, non pas parce qu'ils sont libres ou gratuits, mais parce qu'ils répondent mieux à certains besoins[19] que leurs concurrents propriétaires. C'est donc en étudiant la diffusion des produits que nous aurons une idée de l'impact du Libre sur l'organisation industrielle informatique.

Cette diffusion dépendra bien sûr des caractéristiques des logiciels, et principalement de l'importance des interrelations technologiques (qui joue sur le besoin de standardisation ouverte, donc sur l'attrait du Libre pour les utilisateurs) et de l'importance des besoins

[19] Voir, par exemple, l'étude d'IDC sur le marché asiatique, citée par le Journal du Net : http ://solutions.journaldunet.com/0102/010222etude_linuxasie.shtml
ou l'analyse d'IDG sur les raisons pour lesquelles les entreprises adoptent ce système d'exploitation : http ://www.idg.net/go.cgi ?id=160879, ou l'analyse d'Alain Lefèvre pour le Journal du Net intitulé «pourquoi les projets open source sont meilleurs» :
(http ://solutions.journaldunet.com/0102/010207decrypt.shtml).

spécifiques (qui joue sur le besoin de service, donc sur l'attrait du Libre pour les producteurs et les utilisateurs). D'autre part, plus les logiciels seront l'objet d'un renouvellement important, plus le Libre aura des facilités à s'imposer car, sur ces nouveaux marchés, il est à égalité avec le Propriétaire. À condition, bien sûr, qu'il y ait une offre libre, c'est-à-dire qu'il existe une population d'utilisateurs-développeurs susceptibles de la soutenir.

Il nous faudra distinguer, dans cette étude, les logiciels d'utilisation, où seuls interviennent les fabricants des technologies d'utilisation, des logiciels d'architecturation (système d'exploitation), lieu de rencontre des fabricants des technologies d'architecturation, des technologies élémentaires (et notamment du micro-processeur) et des fabricants des technologies d'utilisation. Nous serons particulièrement attentifs à l'évolution de l'organisation de production du système d'exploitation : historiquement, c'est toujours l'évolution de la production de ce «composant» qui a permis la réorganisation de l'industrie.

Nous allons donc étudier d'abord la diffusion des logiciels libres dans les technologies d'utilisation, puis dans les technologies d'architecturation. Cela nous permettra, dans un troisième point, de synthétiser les étapes de la diffusion du Libre.

3.2.1 La diffusion des technologies d'utilisation.

L'analyse de la diffusion des logiciels libres menée au chapitre 1 nous amène à distinguer naturellement deux types de logiciels : les logiciels utilisés uniquement par les utilisateurs designers ou les utilisateurs sophistiqués (ce qu'on pourrait appeler les logiciels «outils» ou encore les logiciels «techniques») et les logiciels qui sont utilisés par tous, les logiciels «grand-public». Nous allons montrer qu'autant la diffusion du Libre paraît naturelle pour les logiciels techniques, autant, pour les logiciels grand-public, elle va dépendre de façon déterminante du soutien des entreprises utilisatrices de logiciels.

Les logiciels «techniques», des logiciels facilement libérables.

Les logiciels techniques sont des logiciels pour lesquels le passage au Libre devrait se faire assez facilement, parce que c'est dans la production de ce type de logiciels que l'organisation Libre est la plus efficace, mais aussi parce que les modèles économiques

«Propriétaires» sont les plus proches des modèles économiques de service que le Libre peut proposer, donc que l'évolution peut se faire assez facilement.

Les logiciels pour lesquels l'organisation Libre est la plus efficace et la plus présente.

Les logiciels techniques sont les logiciels à la base de toute infrastructure informatique. Ce sont donc des logiciels qui doivent être parfaitement connus de leurs utilisateurs, pour pouvoir les configurer, garantir la sécurité des installations informatiques et adapter les services demandés aux besoins de ces utilisateurs ou des organisations qui les emploient. Les exigences de qualité, d'assurance-qualité, de respect des standards (telles que nous les avons définies dans les chapitres 1 et 2) prennent tout leur sens pour ce type de logiciels. On comprendra donc que leurs utilisateurs valorisent particulièrement l'ouverture de leur code-source et leur non-appropriabilité. Ces utilisateurs sont aussi les mieux informés de l'existence des logiciels libres, les plus à même de les évaluer et de les adapter à leurs besoins.

C'est aussi pour ces logiciels que les producteurs ont le plus intérêt à proposer des logiciels libres car c'est pour ces logiciels que les modèles économiques du Libre fonctionnent le mieux : les marchés sont des marchés de service (ce sont des outils qu'ils faut adapter aux besoins spécifiques des organisations), les utilisateurs sont capables d'évaluer la qualité technique des développement proposés par les producteurs et de proposer leurs propres contributions.

Enfin, c'est aussi dans ces domaines que l'utilisation de logiciels libres est la plus ancienne, notamment grâce à Internet, que les innovations venant du monde de la recherche et des utilisateurs sont les plus nombreuses, innovations favorables à la diffusion du modèle Libre.

Le Libre apparaît aussi, dans ce secteur, comme une évolution naturelle du mode d'organisation économique, ce qui facilite sa diffusion et le passage des logiciels (et des entreprises) fonctionnant autour du modèle propriétaire au modèle Libre.

Une simple évolution du métier de producteur, ce qui facilite le passage d'un modèle à l'autre.

Remarquons d'abord que, dans ce secteur des logiciels techniques, lorsqu'il existe un logiciel libre, il est, le plus souvent, contrôlé par une entreprise. Elle peut être à l'origine du logiciel (comme Digital Creation avec Zope ou Open Cascade avec Matra Datavision) ou avoir été créé par des personnes intégrées dans le noyau des développeurs (comme Ada avec ACT, TCL avec Scriptics, GCC avec Cygnus, etc.) Ces entreprises contrôlent et garantissent l'évolution du logiciel[20] et vendent du service d'assistance à l'utilisation de leur(s) outil(s). Pour ces entreprises, la marque est liée à la marque du logiciel. Elles cherchent souvent à être reconnues comme l'entreprise d'un logiciel.

Ce métier est assez proche de celui des constructeurs d'outils-logiciels propriétaires, comme Oracle ou Ilog[21] : toutes ces entreprises vendent du support, le plus souvent au forfait, ou de l'adaptation de composants aux utilisateurs finals ou à d'autres producteurs de technologies d'utilisation.

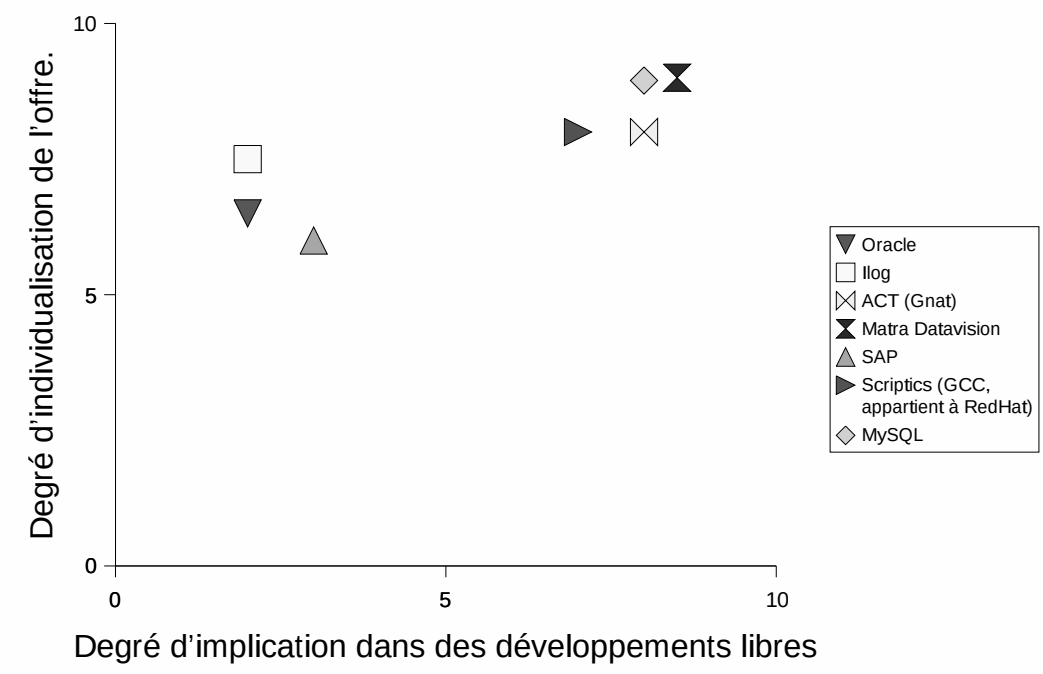
Cela explique que, sur le graphique 3.1, les entreprises soient toutes, peu ou prou, sur la même ligne en ce qui concerne l'individualisation de l'offre, la différence se faisant principalement sur la facilité de la paramétrisation et de l'interfaçage. Plus qu'une révolution, l'évolution vers le Libre ne fait qu'accompagner l'évolution vers la production de composants, qui augmente le besoin de certificat de qualité et surtout la garantie d'interfaces normalisées, bref de services de «mise à disposition de capacités techniques entretenues», pour reprendre la classification de Gadrey [1998].

Si ces composants libres s'imposent, on devrait les retrouver ensuite dans les logiciels développés grâce à eux ou qui communiquent avec eux, mais qui sont utilisés par des utilisateurs non-développeurs, notamment par les utilisateurs naïfs, donc dans les applications «grand-public».

[20] Dans le cas d'Ada, la société ACT maintient une base de données des erreurs et des problèmes rencontrés par ses utilisateurs. Lorsqu'elle incorpore de nouveaux composants à son logiciel, elle peut, grâce à cette base de données, vérifier qu'ils fonctionnent correctement et qu'ils ne recréent pas des problèmes corrigés dans la version précédente du logiciel. Ces tests, appelés «tests de non régression», sont un des éléments clefs de l'avantage concurrentiel de cette société.
La société Matra Datavision possède le même type de base pour son logiciel Open Cascade.

[21] Un producteur français de logiciels de développement d'application, issu de l'INRIA.

Figure 3.1 — Positionnement des fabricants de composants et d'outils.



Légende :
- le degré d'implication dans des développements libres mesure l'utilisation de logiciels libres dans l'offre (de 0 à 6), la recherche de compatibilité avec des produits libres (comme GNU/Linux, de 0 à 2) et le partage d'informations sur l'évolution des logiciels (de 0 à 2).
- le degré d'individualisation de l'offre mesure les possibilités de paramétrisation (de 0 à 4), l'aisance de cette paramétrisation (de 0 à 3) et la facilité pour interfacer le logiciel avec d'autres logiciels (respect des standards pour la production des données, facilité d'intégration de données produites par d'autres logiciels, etc., de 0 à 3).

Les applications «grand-public».

Il s'agit des logiciels du système d'information de l'entreprise, suite bureautique, logiciels clients pour Internet, logiciels de travail collaboratif, etc., mais aussi des logiciels utilisés exclusivement hors de l'entreprise, comme les jeux. Une première condition à l'existence d'un logiciel libre de ce type est qu'il existe une population d'utilisateurs-développeurs capable de le soutenir. C'est de plus en plus souvent le cas grâce à Internet (il y a, dans toutes les professions, des passionnés d'informatique)[22]. Si l'on suppose ce problème surmonté, nous allons voir qu'une condition nécessaire à la diffusion du logiciel libre est son utilisation dans les entreprises. Nous en tirerons les conséquences sur

[22]Même si, plus les professions ont des spécificités locales, plus cette taille critique est difficile à atteindre.

l'évolution de l'industrie.

Les conditions de mise en libre d'une application «grand-public» ; l'importance des entreprises.

Nous avons montré que deux choses s'opposent à la diffusion de logiciels grands-publics libres : d'une part, le problème de leur production et surtout de leur adaptation aux besoins des utilisateurs naïfs, d'autre part celui de leur diffusion dans une concurrence entre standards, face à des logiciels propriétaires soutenus par des entreprises. Parce que tous ces produits subissent de forts rendements croissants d'adoption (par interrelation technologiques, par effet d'apprentissage aussi), l'effet de lock-in est très important : il existe déjà des offres standards pour beaucoup de ces besoins. Leurs coûts de production et de distribution sont déjà amortis alors qu'il faut développer les logiciels libres. Enfin, l'adoption de ce type de logiciels par les particuliers demande un effort de soutien important de la part des entreprises (Microsoft consacre près de la moitié de ses dépenses à distribuer et à faire la promotion de ses produits !)

La diffusion des logiciels libres «grand-public» passe donc par leur adoption par les entreprises, qui sont capables de supporter les coûts de non adoption d'un standard si le concurrent leur apporte un meilleur service. Historiquement, d'ailleurs, les changements dans l'organisation industrielle sont survenus parce que les entreprises n'étaient plus satisfaites des logiciels produits par l'ancienne organisation et ont soutenu de nouvelles offres de logiciels, comme nous l'avons montré au chapitre 1.

On peut donc penser qu'il se développera une offre libre de suite bureautique, de logiciels de travail coopératif, bref de tous les outils servant à la production et au traitement de l'information dans l'entreprise : la pérennité des informations, la possibilité de pouvoir les échanger facilement sont des préoccupations de plus en plus importantes pour les entreprises. Ce sont aussi pour ces logiciels que les perspectives de marchés de services sont les plus importantes : il faut adapter les systèmes d'information aux besoins de chaque entreprise, c'est-à-dire, notamment, faire fonctionner ces logiciels «grand-public» avec les outils techniques qui forment le squelette du système d'information. Le rôle de l'entreprise de service est proche, dans ce cas, de celui des SSII traditionnelles,

puisqu'il s'agit d'adapter des technologies pour répondre à une demande en terme de caractéristiques d'utilisation. Et, comme pour les logiciels techniques, les contributions des entreprises et des utilisateurs-développeurs à un logiciel libre grand-public sont d'autant plus importantes que le logiciel évolue vite, c'est-à-dire que la dynamique de la demande est importante et hétérogène, car c'est cette hétérogénéité qui va créer les opportunités de marché. Le fait que les efforts de développement actuellement sponsorisés par les sociétés du libre portent sur les logiciels à la base du système d'information des entreprises (standard d'échange de données avec XML, suites bureautiques, systèmes de travail collaboratif, systèmes de stockage des données, etc.) est un signe très positif pour la diffusion de ces systèmes, notamment car cela permet de diminuer l'impact du sponsoring des logiciels propriétaires.

On ne peut cependant pas dire que cette adoption se fera contre ces utilisateurs naïfs. Ils n'ont pas vraiment de pouvoir de décision actuellement, ils sont dépendants de la stratégie des producteurs propriétaires de standards. Si les entreprises (et particulièrement les grands comptes), qui sont les plus intéressées au développement de solutions libres veulent pouvoir adopter ces solutions, il faut qu'elles les adaptent à tous leurs employés et donc aux utilisateurs naïfs. Si les sponsors du nouveau standard veulent les faire changer de standard, il faudra qu'ils fassent un effort particulier pour adapter leur offre à cette demande. Donc, il est probable qu'au final, si les logiciels libres «grand-public» sont adoptés, c'est qu'ils correspondront mieux aux besoins des utilisateurs naïfs que les logiciels propriétaires actuels. Ce qui nous amène à analyser les conséquences de cette diffusion sur l'organisation industrielle.

La nouvelle organisation industrielle.

La diffusion des logiciels libres chez les utilisateurs individuels, si elle a lieu, se fera lentement, d'autant plus lentement que les logiciels ne seront pas utilisés dans les entreprises. On peut même penser que si ces logiciels ne sont pas utilisés par les entreprises ou sont à rendement croissants d'adoption faible (comme les jeux, les encyclopédies), ils resteront produits sous forme propriétaire. Nous avancerons même l'idée que l'organisation Propriétaire devienne le système de production des logiciels qui ne sont pas des outils de

production pour les entreprises et notamment des œuvres de contenu. Ce qui implique une réorganisation de la production industrielle de progiciels en deux pôles, l'un s'appuyant sur la production des composants libres, l'autre sur la production de systèmes fermés se suffisant à eux-mêmes. Il nous semble que la dispersion des offres qui apparaît sur le schéma 3.2 annonce l'émergence de ces deux structures productives : d'une part, une production propriétaire (en bas à gauche) pour les logiciels répondant à une demande peu hétérogène, avec de faibles rendements croissants d'adoption (autres que les effets d'échelle), comme les encyclopédies ou les jeux[23] ; d'autres part, une production libre, fondée sur les composants libres, pour les logiciels de production et d'échange d'information, qu'il faut adapter à chaque entreprise (en haut à droite), avec sans doute aussi une déclinaison «sur étagère», produite par les vendeurs de machines ou les éditeurs de distributions, pour répondre aux besoins des utilisateurs naïfs.

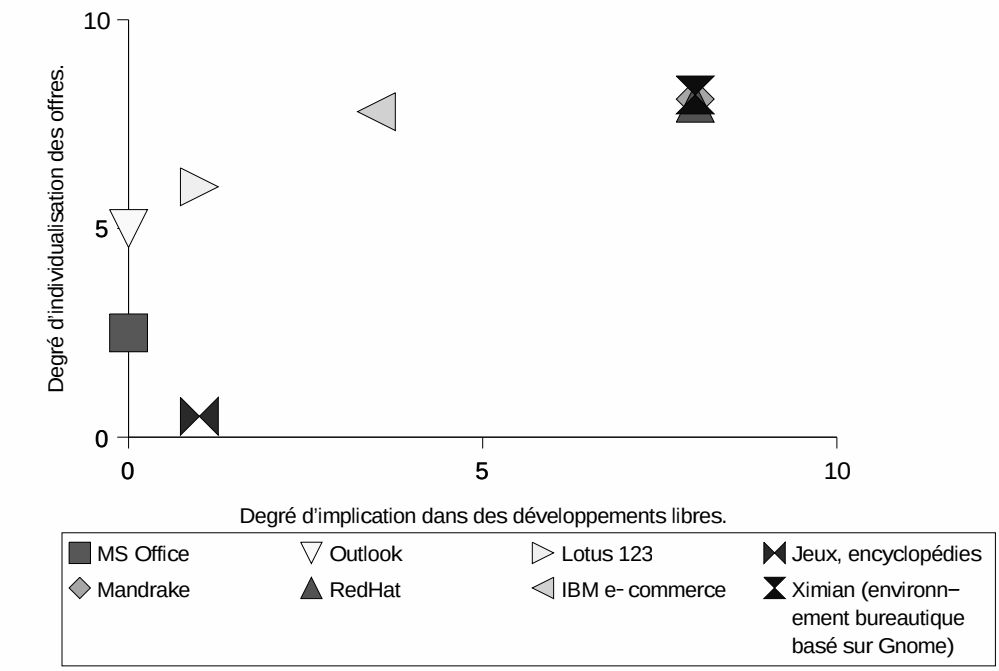
Mais cette réorganisation reste liée à l'évolution de l'élément clef de l'organisation industrielle en informatique, le système d'exploitation.

3.2.2 La diffusion d'un système d'exploitation libre : le point clef de la diffusion du Libre.

Les logiciels d'application sont en même temps complémentaires au système d'exploitation, liés technologiquement à celui-ci et relativement similaires en terme de processus de production. Un producteur qui propose système d'exploitation et logiciels d'utilisation peut utiliser une offre pour promouvoir l'autre. SUN, constructeur de machines sous Unix, qui développe son propre système d'exploitation, Solaris, utilise cette complémentarité : pour favoriser la vente de son système et de ses machines, il développe l'offre de logiciels d'utilisations (création du langage Java, rachat de la suite bureautique Star Office, etc.) ; Microsoft, avec sa suite Office, a exploité au maximum cet effet d'interrelation technologique : Office est la suite qui fonctionne le mieux avec Windows et qui utilise le plus rapidement les avancées de Windows et son succès favorise le système d'exploitation car

[23] Il existe des jeux développés en libre ; ce sont d'ailleurs souvent des jeux en réseau, donc pour lesquels les problèmes de standard sont importants. On peut tout à fait imaginer l'existence de «plates-formes» de jeux libres, financés par les producteurs de logiciels-scénarios développés sous forme propriétaires ou par les fournisseurs d'accès à Internet. C'est cohérent avec l'idée d'une infrastructure libre faisant fonctionner des contenus propriétaires.

Figure 3.2 — Positionnement des producteurs de logiciels «grand-public».



Légende :
- le degré d'implication dans des développements libres mesure l'utilisation de logiciels libres dans l'offre (de logiciels «grand-public», de 0 à 6), la recherche de compatibilité avec des produits libres (comme GNU/Linux, de 0 à 2) et le partage d'informations sur l'évolution des logiciels (de 0 à 2).
- le degré d'individualisation de l'offre mesure les possibilités de paramétrisation (de 0 à 4), l'aisance de cette paramétrisation (de 0 à 3) et la facilité pour interfacer le logiciel avec d'autres logiciels (respect des standards pour la production des données, facilité d'intégration de données produites par d'autres logiciel, etc., de 0 à 3).

la mise en réseau, qui fait que les échanges de fichiers augmentent, pousse les utilisateurs qui reçoivent des fichiers Word à adopter ce traitement de texte, qui ne fonctionne qu'avec lui.

Si le système d'exploitation des ordinateurs personnels reste contrôlé par Microsoft, il est douteux, à cause de ces interrelations technologiques, que les producteurs du Libre puissent fournir des solutions standards plus efficaces que celles de Microsoft et qu'il y ait une réelle diffusion des logiciels libres «grand-public». La rupture apportée par le Libre ne sera effective que s'il y a rupture du contrôle propriétaire du système d'exploitation pour un contrôle collectif de son évolution.

Nous allons d'abord voir que beaucoup d'acteurs ont un intérêt à cette rupture, parce que ce contrôle diminue leur propre contrôle sur leurs relations avec leurs clients. Ensuite, nous nous interrogerons sur la façon dont cette diffusion peut effectivement avoir lieu et sur son impact sur l'organisation industrielle.

La libération du système d'exploitation : un avantage pour beaucoup d'acteurs du monde informatique.

Parce que le système d'exploitation est à l'articulation entre logiciel et matériel, son étude recouvre une double complexité : la diversité des producteurs intéressés par son évolution est très grande et les utilisateurs ont des connaissances sans doute encore plus hétérogènes du fonctionnement de ce logiciel et de la façon dont il peut être adapté pour leur rendre des services (voir chapitre 1). Tous les acteurs de l'informatique, constructeurs des technologies d'utilisation, d'architecturation, de technologies élémentaires et utilisateurs sont intéressés, à des degrés divers, par son évolution et son contrôle. Tous sont concernés par l'ouverture de ce système et ont des raisons stratégiques pour contribuer au développement d'un système libre, ou au moins pour encourager ce développement. Les constructeurs de technologies d'utilisation et de technologies élémentaires sont, par exemple, particulièrement intéressés par le développement d'un système ouvert, qui leur permet de publier leurs produits sans tenir compte des éditeurs de système d'exploitation. Ainsi Intel a favorisé le développement de Linux pour son nouveau processeur, le Merced, afin de garantir un marché pour ce processeur car cette entreprise n'était pas sûre que Microsoft aurait terminé l'adaptation de ses systèmes au moment de sa sortie. Les raisons que peut avoir chaque catégorie de producteurs à contribuer à un système d'exploitation libre sont présentées dans le tableau 3.2.

Comme dans le cas du matériel, chaque acteur, à partir du moment où il ne contrôle pas le système d'exploitation dominant ou standard, a intérêt à ce qu'il soit le plus ouvert possible pour pouvoir diriger ses évolutions ou au moins s'assurer que les évolutions ne se font pas à son détriment. Ce type de processus de standardisation peut-être considéré comme un exemple typique de «bataille de sexes» tel que décrit par Besen & Farrell [1994], où chaque acteur veut pousser sa solution, mais valorise plus l'existence d'un standard que

	Fabricant de composants élémentaires	**Constructeurs d'architectures**	**Fabricants de technologies d'utilisation**	
			Fabricant de logiciels	Fournisseur de services
Objectif technique recherché :	compatibilité du composant avec le noyau.	contrôle de l'évolution du système, compatibilité de l'architecture avec le standard ouvert, volonté d'affaiblir les standards propriétaires des concurrents ou des fournisseurs.	compatibilité du logiciel avec la machine universelle.	qualité du service, qualité des logiciels utilisés.
Actions / contribution	développement du sous-système permettant cette compatibilité.	participation au développement général du système, financement de projets permettant d'en faire une offre crédible (développement des fonctionnalités), adaptation du système à ses architectures (voir l'action d'IBM).	travail sur les interfaces de compatibilité (voir l'action de Corel).	surveillance de l'évolution du noyau, participation si fournisseur de service d'ingénierie.
Raison stratégique	compatibilité avec une base installée (architecture mais aussi machine universelle).	pouvoir garantir une qualité de service, un niveau de performance sur la machine universelle, garder le contrôle sur l'évolution du système d'exploitation, pouvoir répondre à des demandes spécifiques des utilisateurs designers.	compatibilité avec une base installée.	signal de qualité / intégration dans un groupe susceptible de résoudre des problèmes techniques posés par le client / développement et maintenance de solutions pour les clients.

Tableau 3.2 — Contributions au développement d'un système d'exploitation libre et stratégies d'entreprises; l'exemple du noyau Linux.

le succès de sa proposition. La difficulté pour le Libre est donc d'imposer un standard là où existent déjà Unix et Windows NT/2000[24] pour les applications techniques (notamment les serveurs), Windows 98 pour les ordinateurs de bureau, sachant que c'est sans doute le logiciel pour lequel le lock-in est le plus important, du fait de l'interrelation technologique et des effets d'apprentissage.

Les conditions de diffusion d'un système d'exploitation libre.

Les conditions de diffusion d'un système d'exploitation libre ne sont pas très différentes des conditions de diffusion des logiciels d'utilisation : il faut d'abord qu'existent des marchés où les qualités propres de ces logiciels leur permettent de s'implanter et qu'ensuite ils soient adoptés et diffusés par les entreprises afin de conquérir les utilisateurs naïfs.

De nouveaux marchés, souvent techniques, où les systèmes d'exploitation libres ont des positions fortes.

Naturellement, lorsqu'on pense aux logiciels libres et à leurs qualités d'ouverture, de respect des standards, le premier marché qui vient à l'esprit est celui des serveurs de réseau (serveurs Internet, mais aussi serveurs locaux, serveurs d'imprimante, serveurs de fichier, serveurs de données, etc.) C'est effectivement là que les systèmes d'exploitation libres sont les plus utilisés (près de la moitié des serveurs Web utilisent un système d'exploitation libre). C'est aussi là que la standardisation autour d'un système d'exploitation est la moins forte : il existe de nombreuses versions d'Unix (Linux ou les systèmes d'exploitation du type BSD en sont d'ailleurs des déclinaisons), en plus de l'offre de Microsoft, Windows 2000. De plus, ces usages des systèmes d'exploitation sont réservés à des spécialistes de l'informatique, souvent à des utilisateurs designers, pour qui le passage d'un Unix à un autre est relativement aisé et qui sont capables de tirer partie de l'ouverture des logiciels libres.

D'autre part, sur les nouveaux marchés de l'informatique (PDA[25], systèmes embarqués, ordinateurs à faible coûts, etc.), aucun système d'exploitation propriétaire n'a de position dominante. Ce sont des marchés techniques, où il faut adapter et optimiser le système

[24] À l'heure où nous écrivons, le successeur annoncé de ces logiciels, Windows XP, n'est pas sorti.
[25] «Personal Digital Assitant», nom donné aux ordinateurs de poche comme le Psion.

d'exploitation aux capacités limitées de la machine. Ce sont donc des marchés où les qualités des logiciels libres peuvent s'exprimer. On peut noter que des entreprises comme RedHat s'y impliquent beaucoup et qu'il existe de nombreuses entreprises spécialisées dans ce domaine[26]. Ce sont aussi des marchés où les logiciels libres (et notamment Linux) sont particulièrement utilisés[27] .

La diffusion des systèmes d'exploitation libres peut aussi se faire à partir de niches de marché géographiques et particulièrement des marchés des nouveaux pays industrialisés : Linux a été choisi par la Chine et le Mexique comme le système d'exploitation officiel du pays parce qu'il permet de construire des solutions à faible coût et parce qu'il est indépendant des entreprises ; profitant de son faible coût d'utilisation et de sa flexibilité, des sociétés indiennes ont utilisé ce système pour construire des ordinateurs à bas prix (moins de 200$), adaptés aux besoins des communautés villageoises rurales. Si le Libre réussit à s'imposer sur ces marchés qui sont souvent des marchés «grand-public», cela permettra de développer des applications pour ce système et le passage des utilisateurs naïfs aux systèmes libres sur les machines informatiques traditionnelles s'en trouvera facilité.

On voit donc qu'on peut envisager la diffusion des systèmes d'exploitation libres dans des niches de marché. Mais, c'est une fois de plus l'action des entreprises utilisatrices puis productrices qui déterminera la place de l'offre libre et la possible «libération» du système d'exploitation.

L'implication des entreprises, facteur clef de la diffusion du modèle Libre.

Les entreprises utilisatrices représentent un marché important parce que c'est le lieu de rencontre entre les utilisateurs développeurs et les utilisateurs naïfs et parce que c'est aussi le principal marché pour les micro-ordinateurs (il ne faut pas oublier que, encore aujourd'hui, les achats des entreprises représentent les trois quarts des achats totaux de

[26] Il existe un site dédié à l'utilisation de Linux dans les systèmes embarqués : http ://www.linuxdevices.com/.

http ://www.eetimes.com/story/OEG20010125S0043 donne les raisons de l'intérêt des constructeurs automobiles pour ce système.

[27] Voir http ://www.idg.net/go.cgi ?id=538524.

PC[28]).

Comme pour les logiciels de traitement de l'information, les entreprises utilisatrices ont intérêt à encourager l'ouverture dans le domaine du logiciel car celle-ci leur permettra de s'affranchir de l'emprise des constructeurs (dans le monde Unix) et de Microsoft (dans le monde des PC), tout en garantissant une standardisation du système sur tout le parc de machine.

Pour qu'elles soutiennent cette diffusion, il faut d'abord que les systèmes d'exploitations libres (et les logiciels libres) aient prouvé leur efficacité sur les marchés techniques de référence, mais aussi qu'ils puissent cohabiter avec les logiciels et les systèmes d'exploitation propriétaires. Des projets (libres) comme CORBA, qui proposent d'insérer, entre les logiciels d'utilisation et les systèmes d'exploitation une couche logiciel supplémentaire[29] qui les rendrait indépendants, SAMBA, qui permet à un serveur sous Unix (par exemple sous GNU/Linux) de répondre à des requêtes de machines sous Windows, ou Wine, qui permet de faire fonctionner des logiciels développés pour Windows sous GNU/Linux, sont, à cause de cela, des projets très importants pour la diffusion du Libre.

La diffusion des systèmes d'exploitation libre sur les postes de travail dans les entreprises devrait entraîner, dans un second temps, leur diffusion chez les particuliers, par effet de contagion, car, comme le montre l'histoire du PC, pour minimiser les coûts d'apprentissage, ces utilisateurs adoptent souvent chez eux le même système que celui qu'ils utilisent sur leur lieu de travail. Le développement d'une demande de produits libres va aussi favoriser l'implication des producteurs, renforçant leur attractivité.

L'implication des producteurs.

Nous distinguons deux types de stratégies chez les producteurs de systèmes d'exploitation, stratégies qui apparaissent assez nettement sur la figure 3.3.

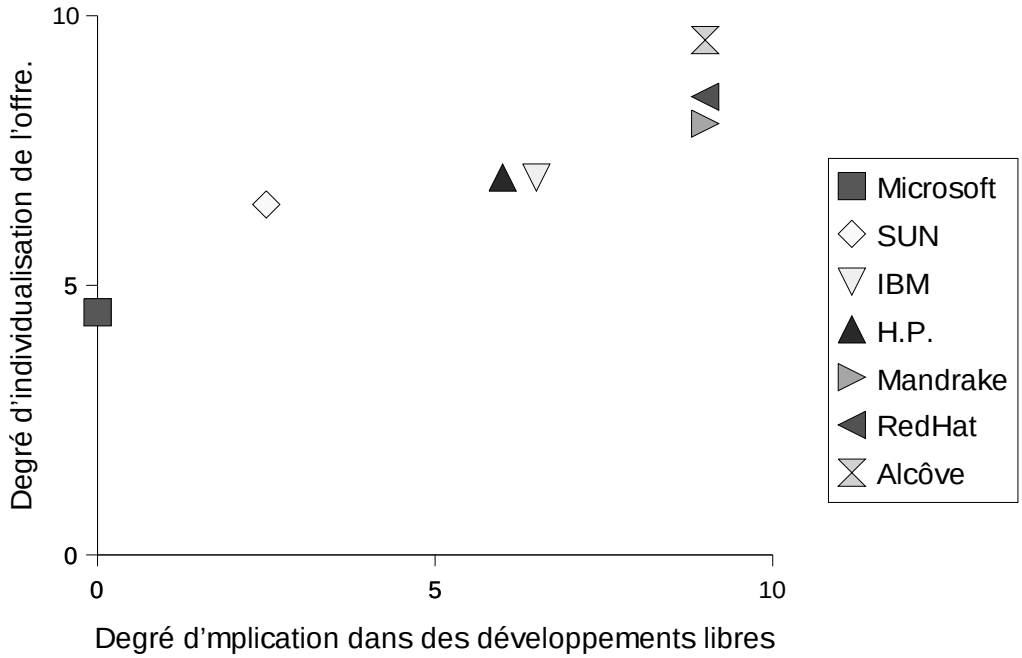
D'un côté, on trouve les entreprises spécialisées qui contrôlent un standard dominant (SUN et Microsoft), qui s'impliquent peu dans la production des systèmes libres[30] ; de

[28] El Pais, 12 août 2001.

[29] Gnome, qui est une des interfaces graphiques de Linux et qui a été choisie par SUN pour devenir l'interface graphique de ses machines, à été développé grâce à CORBA.

[30] SUN a cependant annoncé qu'elle se proposait de publier en libre son système d'exploitation, Solaris. Mais le type de licence utilisé, qui a évolué au cours du temps, le fait que cette annonce n'ait pas encore été suivie d'effets, le fait, enfin, qu'au lieu de participer à l'élaboration du standard qui semble émerger, Linux,

Figure 3.3 — Positionnement des producteurs de systèmes d'exploitation.



Légende :
- le degré d'implication dans des développements libres mesure l'utilisation de logiciels libres dans l'offre (de système d'exploitation, de 0 à 6), la recherche de compatibilité avec des produits libres (comme GNU/Linux, de 0 à 2) et le partage d'informations sur l'évolution des logiciels (de 0 à 2).
- le degré d'individualisation de l'offre mesure les possibilités de paramétrisation (de 0 à 4), l'aisance de cette paramétrisation (de 0 à 3) et la facilité pour interfacer le logiciel avec d'autres logiciels (respect des standards pour la production des données, facilité d'intégration de données produites par d'autres logiciels, etc., de 0 à 3).

l'autre, les entreprises qui se positionnent uniquement sur le marché du libre (comme RedHat ou Alcôve) et qui contribuent fortement au développement et à la diffusion de ce standard. Entre les deux, les fabricants traditionnels, qui ne fondent pas leur stratégie sur un standard et qui reproduisent les comportements qu'ils ont déjà eus dans le passé face aux innovations comme Unix ou les micro-ordinateurs : voyant qu'il existe une demande, ils cherchent à intégrer la nouvelles offre libre dans leur portefeuille d'offre, comme ils l'ont fait avec les innovations précédentes (Unix, micro-ordinateurs, etc.) C'est d'autant plus logique que ces constructeurs ne maîtrisent pas un standard dominant et que, dans ce cas,

SUN encourage son propre système montrent que cette entreprise recherche toujours à avoir le premier rôle dans le contrôle de l'évolution du système d'exploitation.

ils ont tout intérêt à favoriser le développement d'un concurrent à ce standard dominant. Ce faisant, ils légitiment l'offre libre en la mettant au même niveau que les autres offres de système d'exploitation et facilitent sa diffusion.

La conséquence : une bipolarisation de l'industrie informatique.

GNU/Linux, les Unix BSD, sont des systèmes de type Unix, qui fonctionnent selon les mêmes concepts que les Unix propriétaires de SUN (Solaris), IBM (AIX), ou HP (HP-UX), pour ne citer qu'eux. Les marchés «techniques» sur lesquels les logiciels libres sont les plus performants sont aussi les marchés traditionnels d'Unix. Nous l'avons vu au premier chapitre, si le marché des machines sous Unix est le deuxième marché en valeur après celui des micro-ordinateurs, c'est aussi un marché morcelé, plus proche dans son organisation de l'informatique traditionnelle que de l'industrie des PC. Par conséquent, comme le montre Genthon [1995], cette organisation industrielle est moins efficace en ce qui concerne la production de matériel que celle du PC, à tel point que les performances matérielles sont aujourd'hui devenues aujourd'hui comparables. Les fabricants de machines Unix sont aujourd'hui en concurrence avec les fabricants de stations de travail à architecture «Intel» qui utilisent principalement Windows 2000 comme système d'exploitation (au moins sur les marchés des petits serveurs).

L'arrivée des logiciels libres sur ce marché a un double impact : d'un côté, elle crée une offre d'un Unix standard, indépendant des plates-formes et elle renforce l'attractivité des systèmes Unix (qui sont le support privilégié des logiciels d'utilisation libres, même s'ils fonctionnent aussi pour la plupart sous Windows) ; d'un autre côté, elle accentue la pression concurrentielle sur les fabricants de machines sous Unix car GNU/Linux permet de proposer un Unix fonctionnant sur les PC.

Au niveau matériel, l'impact des logiciels libres ne fait que renforcer un processus déjà en cours : celui de la diffusion de la structure ouverte de l'industrie du PC à l'ensemble de l'industrie[31] (c'est ce que Horn [2000b] a appelé «la fusion tendancielle des réseaux»).

[31] Les ordinateurs utilisent de plus en plus les mêmes composants (IBM a annoncé il y a quelques années que toutes ses machines fonctionneraient en utilisant des puces PowerPC, par exemple). Pour pouvoir utiliser ces composants, leur structure interne se rapproche : SUN, comme Apple, utilise les standards des PC pour ses bus (PCI, USB), c'est à dire pour les systèmes qui transportent l'information entre les différents composants et périphériques de la machine. Enfin, la production de ces composants est de plus en plus ex-

Au niveau des logiciels, la diffusion de GNU/Linux, et notamment son adoption par les fabricants de PC[32] et par les constructeurs de machines Unix qui ne sont pas leaders sur ce marché (IBM, HP, Compaq, bref tous les constructeurs sauf SUN) devrait conduire à la disparition des Unix propriétaires.

À l'image de l'offre de Dell, ce marché devrait donc évoluer vers la constitution de deux offres, une centrée autour de GNU/Linux et des logiciels libres «techniques», l'autre autour des produits Microsoft[33]. Les qualités du Libre (qualité technique, meilleure standardisation, meilleures relations de services) laisserait supposer une domination forte de cette organisation sur ce marché technique. Cependant, il faut se rappeler que les phénomènes d'interrelation technologique donnent un avantage certain à l'offre de Microsoft, qui contrôle le marché des ordinateurs individuels. Comme nous le signalions plus haut, l'existence et l'efficacité des logiciels (Samba) et des protocoles (XML) «passerelle» entre les serveurs libres et les machines clientes propriétaires sera un des points clefs de la diffusion des outils libres.

Conclusion.

L'analyse de la diffusion des logiciels libres, tant au niveau des logiciels d'utilisation qu'au niveau des systèmes d'exploitation fait ressortir une constante forte : cette diffusion devrait se faire en deux étapes. La première étape correspond à des utilisations techniques, par des utilisateurs, designers ou sophistiqués. Si ces logiciels font leur preuve dans ces configurations, les entreprises pourront être tentées de les diffuser à tout leur système d'information, parce qu'ils garantissent mieux le respect des standards et l'adaptation des logiciels aux besoins spécifiques de l'entreprise (bref le «sur-mesure de masse»). Comme pour les changements d'organisation industrielle précédents, ce sont les choix des entreprises utilisatrices de logiciel qui vont déterminer l'évolution de l'industrie informatique.

ternalisée. Le meilleur exemple de ce phénomène est IBM, qui a abandonné une grande partie de son activité de production de composants pour la transformer en une relation de partenariat étroit avec ses fournisseurs. Voir : http ://www.manufacturing.net/magazine/purchasing/archives/1999/pur0916.99/092medal.htm pour une présentation de l'évolution stratégique d'IBM sur ce point.

[32]Dell propose aujourd'hui deux systèmes d'exploitation pour ses serveurs : Windows 2000 ou GNU/Linux.

[33]Ce processus illustrerait assez bien les théories évolutionniste et notamment les modèles «à la Nelson & Winter», où, lorsqu'il y a concurrence entre plusieurs types de standards, les offres les plus faibles sont éliminées en premier, laissant place à une concurrence entre un petit nombre d'offres qui peuvent parfois coexister.

Il nous semble possible que ces choix conduisent à la domination du Libre, dans la production des systèmes d'exploitation et plus généralement dans la production des logiciels techniques et des logiciels «grand-public» utilisés dans les entreprises. Nous allons, pour terminer ce chapitre, nous placer dans le cas d'une domination du Libre et essayer d'envisager quelle(s) pourrai(en)t être la ou les organisation(s) industrielle qui émergerai(en)t.

3.2.3 Quelle organisation industrielle en cas de succès du Libre ? L'importance du contrôle de la production du système d'exploitation.

Après avoir comparé l'organisation Libre face à l'organisation Propriétaire, en supposant implicitement que les acteurs de chacune de ces organisations poursuivaient tous la même stratégie, nous allons essayer d'approfondir ce que peuvent être ces différentes stratégies dans le cas d'une diffusion du Libre et leurs conséquences sur l'organisation industrielle. Nous allons d'abord constater que deux types d'organisation industrielle libre sont possibles, types qui diffèrent, une fois de plus, par la façon dont le système d'exploitation est produit. Nous montrerons ensuite qu'une fois de plus, ce sont les entreprises utilisatrices qui devront déterminer l'importance de chacune des organisations.

Deux stratégies pour deux types d'organisation industrielle.

Nous avions constaté, au chapitre 2, deux types de stratégies chez les entreprises qui proposent des solutions à base de système d'exploitation libre : d'une part des entreprises comme Alcôve, qui s'affichent comme des entreprises de service, et qui utilisent indifféremment plusieurs distributions de GNU/Linux (avec souvent une préférence pour la distribution Debian, qui n'est pas contrôlée par une entreprise) et parfois des systèmes d'exploitation BSD, d'autre part des entreprises comme Mandrake ou RedHat, qui éditent une distribution de GNU/Linux et qui proposent des services basés exclusivement sur cette distribution. Cette différence de stratégie illustre deux visions du système d'exploitation qui conduisent à deux organisations industrielles assez différentes.

La stratégie des fournisseurs de services.

Les entreprises comme Alcôve se positionnent sur des marchés d'expertise où il s'agit

de développer, à la place du client, des solutions individualisées et de maintenir ces solutions. On est proche de ce que Gadrey [1998] appelle la «mise à disposition de capacités humaines», en ce sens que ce qui fait la particularité de ces entreprises est qu'elles regroupent en leur sein une équipe de spécialistes de différents logiciels. Leur renommée se construit parmi les utilisateurs développeurs, designers ou sophistiqués, en même temps que leur expertise par des contributions diverses à de nombreux logiciels. Les logiciels libres sont utilisés parce qu'ils permettent de créer une solution plus souple, indépendante des éditeurs et des constructeurs et une relation de service plus équilibrée, à plus long terme. Ces entreprises participent donc au développement de différents systèmes d'exploitations et de différentes distributions sans être dépendantes d'aucun.

La licence qui protège le logiciel est importante pour ces fournisseurs de service : pour pouvoir garantir le contrôle et l'expertise sur les logiciels utilisés, il faut qu'ils soient assurés que ces logiciels sont et resteront libres. Ils préféreront donc sans doute des licences fortement contraignantes comme la GPL. Si cette vision, qui privilégient le côté technique des logiciels s'imposait, il est probable que seront aussi privilégiées les distributions les plus libres (comme Debian), c'est-à-dire celles qui ne sont pas contrôlées par une entreprise. Il y aurait alors une évolution de la filière qui considérerait le système d'exploitation comme un ensemble de composants, avec comme lieu de coordination le noyau, sur lequel interviendraient tous les acteurs de l'informatique (producteurs de technologies élémentaires, producteurs des machines universelles, etc.) Cette segmentation, annoncée par Zimmermann [1998] et Beugnard [2001], fait penser à celle qu'on trouve dans le matériel, où les entreprises se sont spécialisées autour de compétences technologiques et se regroupent en alliance, où les offres de produits, si elles sont de plus en plus standardisées au niveau des composants (systèmes d'exploitation, processeur, technologies élémentaires en général) sont ensuite déclinées en gamme, ce qui permet d'adresser des demandes différentes en terme de technologies et en terme de quantité de services complémentaires. On constate, en effet, une même évolution chez les entreprises du Libre : des fabricants de composants (comme ACT) maîtrisent les technologies élémentaires, des «architecteurs» comme Alcôve les assemblent pour leurs clients qui sont souvent des entreprises de service orientées «métier», qui ont les connaissances pour traduire les besoins de leurs clients en terme de

technologies d'utilisation.

Pour que ce système fonctionne, il faudra que soient particulièrement bien respectés les formats d'échange et la pratique de reverser toutes les contributions dans les versions officielles des différentes composants car, comme les offres seront adaptées à chaque client, les risques de divergence seront importants.

La stratégie des éditeurs de systèmes d'exploitation.

D'un autre côté, des entreprises comme Mandrake ou RedHat essayent d'introduire dans le métier d'ingénierie une dimension produit, en éditant des distributions, c'est-à-dire des assemblages de logiciels. Il s'agit, pour ces entreprises, de standardiser, d'industrialiser ce métier de constructeur d'architecture, qui comprend une forte dimension d'assurance, de garantie d'après-vente, en proposant des assemblages standards de logiciel et des services standards d'assistance. Pour ces entreprises, comme c'est le cas pour les éditeurs comme Microsoft, la construction d'une marque de distribution, signal de la qualité des produits, est importante : c'est elle qui va leur permettre d'atteindre une partie de leur public, celui des utilisateurs naïfs, qui ne disposent, pour évaluer la qualité des offres, que de ce signal en plus de l'information que donne le prix. La limite de ce positionnement réside dans le fait que, comme ces distributeurs s'appuient sur des produits libres, leurs produits peuvent être concurrencés, distribués par d'autres producteurs. Ils ne constituent pas une source de revenu mais simplement un produit d'appel pour leurs services de maintenance[34]. Ces entreprises font le pari de la victoire des logiciels libres dans la guerre de standard sur ces marchés sensibles aux rendements croissants d'adoption.

Si cette vision s'imposait, on risque d'assister à l'émergence de quelques distributions principales du système d'exploitation libre, avec le risque que ces distributions soient imparfaitement compatibles entre elles. Cependant, l'ouverture des logiciels devrait permettre de garantir une compatibilité suffisante entre les versions, faisant ressembler la concurrence à une concurrence entre producteurs de biens compatibles, donc permettant la survie de plusieurs offres. Dans ce cas, le système d'exploitation pourra être vu comme

[34]L'annonce par RedHat de son premier exercice trimestriel bénéficiaire au premier trimestre 2001, dans un contexte difficile de croissance très ralentie aux États-Unis, est indice encourageant de la rentabilité à long terme de ces entreprises. Mais ce résultat est dû au développement des activités de service (et aux résultats financiers de l'entreprise) et non pas à l'activité d'édition et de distribution de Linux RedHat.

un composant unique, proche de ce qu'est le système d'exploitation actuel des PC.

Le choix entre les deux solutions va dépendre de deux types d'acteurs : les constructeurs, qui intègrent les systèmes d'exploitation dans leurs machines et qui, à cause de cela, en sont les principaux distributeurs, et bien sûr les utilisateurs, c'est-à-dire principalement les entreprises clientes.

Le choix de l'organisation industrielle : entre routine et stratégie industrielle.

Le positionnement des constructeurs de machines.

Les constructeurs de machines doivent fournir avec celles-ci un système d'exploitation. La logique industrielle Unix veut que ce soit ces constructeurs qui maîtrisent le développement du système, quand la logiciel PC le délègue à un fournisseur externe. Actuellement, les constructeurs ont tendance à reproduire ces mêmes pratiques dans leur utilisation des systèmes d'exploitation libre : quand IBM ou HP proposent plusieurs distributions (HP allant même jusqu'à choisir la distribution Debian comme distribution de référence), Les constructeurs de PC comme Dell nouent des partenariats avec des producteurs de distribution (RedHat pour Dell)[35]. Ces choix étaient prévisibles : il est plus facile de faire évoluer les «routines» organisationnelles et commerciales[36] que de les révolutionner, mais ils proposent deux types d'organisation industrielle.

Si, dans une organisation industrielle proche de celle des PC, les constructeurs réintégraient une partie du contrôle sur le système d'exploitation, ce serait pour développer les offres de service, et notamment celle de mise à disposition de capacités techniques standards. On assisterait alors au développement des contrats de maintenance globaux (prise en charge de la machine et son système d'exploitation, voire de l'installation et l'«entretien» du système d'information dans son ensemble). C'est la stratégie d'IBM, qui est capable d'intervenir sur toute la chaîne de production, qui peut donc adapter son offre au niveau

[35]Cette opposition doit être nuancée : IBM a aussi noué des partenariats, mais avec plusieurs producteurs de distribution (SuSE, RedHat), et cette entreprise intervient directement sur le développement des systèmes d'exploitation libres.

[36]C'est-à-dire l'ensemble des règles, des pratiques internes à une entreprise et qui en assure le bon fonctionnement. En utilisant ce terme, nous nous référons au sens que la théorie évolutionniste lui donne et donc aussi aux enseignements que cette théorie a produits sur l'organisation et le fonctionnement des entreprises.

technique de la demande (de l'offre construite entièrement sur-mesure pour les utilisateurs designers à l'offre complètement standardisée pour les naïfs en passant par des offres adaptées du côté logiciel pour les sophistiqués). L'utilisation des logiciels libres permet à cette entreprise de construire plus facilement les offres sur-mesures et surtout d'économiser une grande partie des coûts de développement du système d'exploitation, en gardant un contrôle qui lui fait défaut dans l'industrie du PC. Les constructeurs se placeraient alors en position de concurrence avec les entreprises de service comme Alcôve. En même temps, l'utilisation d'un système d'exploitation qu'ils ne contrôleraient pas complètement rendra toujours possible l'utilisation d'un ou plusieurs fournisseurs de machines et d'une entreprise de service pour assurer leur cohabitation.

Si c'est la «culture PC» qui s'impose, les constructeurs risquent d'être relégués au rôle de simple assembleur de composants, les revenus de services étant captés par les fabricants de distributions, RedHat, SuSE ou Mandrake, qui seraient alors les entreprises dominantes de cette nouvelle période industrielle. L'organisation industrielle serait proche de celle qu'on trouve actuellement dans l'industrie du PC, où les marges sont extrêmement réduites pour les assembleurs de machines et très confortables pour les éditeurs de système d'exploitation et de logiciels. Mais l'ouverture, qui facilite la compatibilité, devrait permettre la survie de plusieurs fournisseurs, ce qui garantirait une certaine concurrence entre les distributions. On serait dans le cas où le système d'exploitation se rapprocherait d'un composant, qui serait installé par les fabricants des machines architecturées suivant les demandes des clients.

Les constructeurs de machines seront en partie responsables du sens que prendra l'organisation industrielle, car ce sont eux qui sont actuellement en position de force vis-à-vis des éditeurs de distribution, à cause de leur taille. Mais cette évolution dépendra aussi des entreprises utilisatrices, puisque ce sont elles qui vont sélectionner les offres.

Le choix des utilisateurs et ses conséquences sur l'organisation industrielle.

Le premier système devrait être favorisé par les utilisateurs-développeurs : c'est celui qui leur garantit le meilleur contrôle sur les évolutions des logiciels qu'ils utilisent, où le degré de normalisation est le plus fort et où le moins de choix techniques sont laissés au contrôle

d'une entreprise. Pour les mêmes raisons, cette organisation devrait être privilégiée par les entreprises qui choisissent les offres libres à cause des garanties sur la pérennité des logiciels et sur leur interopérabilité. Mais c'est aussi celle qui demande le plus d'investissement de la part des clients car il faut qu'ils soient capables de suivre l'évolution des standards, ou qu'ils confient ce suivi à une entreprise de service.

Le second système demande moins d'efforts aux clients pour adapter leurs routines : on est dans une organisation plus proche de celle du monde des PC, avec des fournisseurs-éditeurs de systèmes d'exploitation, dont les marques sont connues. Ce système serait sans doute plus lisible pour les utilisateurs naïfs, qui auraient à choisir entre différents produits et non pas à spécifier la façon dont ils veulent construire leur machine.

Suivant l'objectif recherché par les utilisateurs, l'un ou l'autre des systèmes semble meilleur. On peut imaginer qu'ils cohabitent : le premier système pour les applications les plus techniques, et notamment pour les serveurs (donc dans la première phase de la diffusion), le second lorsque le choix est moins technique, lorsqu'il s'agit de distribuer le même système sur un grand nombre de postes, ou lorsque les clients ont moins de compétences techniques (nous pensons, par exemple aux petites entreprises).

Conclusion.

Comme on peut le constater, prévoir l'évolution de l'organisation industrielle libre reste un exercice difficile car les inconnues sont nombreuses. C'est une fois de plus l'interaction des stratégies des constructeurs de machines et des utilisateurs industriels de ces machines qui définira la structure de l'organisation industrielle informatique. Néanmoins, il est certain que, si le Libre se diffuse, et quelque soit cette organisation industrielle, le système d'exploitation sera plus ouvert, moins dépendant des décisions d'une entreprise. Par conséquent, les revenus directs générés par le contrôle de ce système vont diminuer et se déplacer vers les revenus de service, suivant en cela la tendance actuelle de l'industrie.

3.3 Conclusion

Le fait que nous ayons pu identifier le Libre comme une nouvelle organisation économique ne signifiait pas que nous avions prouvé qu'il allait s'imposer, encore moins

que cela fût souhaitable.

Quelle sera alors sa place dans l'organisation de production du logiciel : concurrente de l'organisation Propriétaire, complémentaire de celle-ci ? Une chose est certaine : ces deux organisations sont fondées sur les mêmes institutions, et notamment le même principe légal, le droit d'auteur (ou le copyright aux États-Unis), qui donne un droit de propriété sur un logiciel à son producteur et qui lui permet de licencier ce logiciel à un utilisateur en lui imposant des conditions restrictives d'utilisation. Par contre, ils font une utilisation différente de ce droit d'auteur et construisent des modèles de relations différents. En effet, dans le cas du logiciel propriétaire, il est donné un droit d'utilisation contre rémunération, dans le cas du logiciel libre (et particulièrement de la GPL, voir Clément-Fontaine [1999]), les conditions restrictives ne sont pas sur l'utilisation individuelle, mais sur la diffusion et l'amélioration du logiciel : le licencié doit permettre les mêmes libertés aux personnes à qui il distribue le logiciel, modifié ou non.

Pour cette raison, ces deux organisations n'ont pas les mêmes qualités. Si le Propriétaire peut paraître plus efficace pour inciter les producteurs de logiciels à innover[37], le Libre est meilleur pour diffuser les innovations des utilisateurs et des chercheurs et pour organiser l'innovation incrémentale. Le Libre est aussi plus efficace pour organiser le processus de standardisation et les relations de service entre utilisateurs et producteurs. De ce fait, cette organisation nous semble plus adaptée au «sur-mesure de masse» (pour reprendre l'expression de Horn [2000b]), qui caractérise la production actuelle de logiciel.

Mais, parce que certains logiciels propriétaires sont déjà des standards, le fait que le Libre soit plus efficace ne suffit pas à assurer que cette organisation va se diffuser. Il faut qu'il le soit suffisamment plus pour surmonter les effets de lock-in dus à cette standardisation et qui caractérisent l'industrie du logiciel.

Nous avons identifié plusieurs «marchés», plusieurs types de logiciel, plus ou moins favorables à la diffusion du modèle libre : les marchés des outils, des technologies de base de l'informatique, où la diffusion de ce modèle est une évolution de l'organisation actuelle et devrait se faire assez rapidement, et le marché des applications «grand-public»,

[37] Nous avons montré qu'on peut douter du fait que cet avantage soit très important. Il est cependant sûr que ce système a permis la création d'une industrie dynamique, tant du point de vue du nombre de logiciels créés que des taux de croissance en valeur (on consultera à ce sujet les chapitres 4 et 5 de Horn [2000b]).

utilisées par une population importante d'utilisateurs-développeurs où la diffusion n'est possible que si ces logiciels sont adoptés par les entreprises, parce qu'il y a des besoins (techniques) d'adaptation et d'ouverture. Dans ce cas, la diffusion des logiciels libres parmi la population d'utilisateurs naïfs sera une deuxième étape de la diffusion du libre, qui devra d'abord s'implanter dans des niches technologiques ou sur des nouveaux marchés (comme les logiciels embarqués) avant d'espérer conquérir ces utilisateurs.

Enfin, il est apparu qu'une fois de plus, c'est la diffusion (ou non) des systèmes d'exploitation libres qui permettra (ou non) la diffusion de l'organisation Libre et donc la réorganisation de l'industrie informatique. Et que cette diffusion, si elle a lieu, aura les mêmes caractéristiques que celle des logiciels «grand-public» : une première phase, sur des marchés techniques, et notamment les marchés des serveurs, où les systèmes libres remplaceraient les systèmes Unix propriétaires et concurrenceraient les offres de Microsoft (Windows 2000 et son successeur, Windows XP) ; une deuxième phase, où les systèmes d'exploitation libre seraient installés sur les postes clients, dans les entreprises, puis adoptés par les utilisateurs chez eux, pour retrouver le même environnement que celui de leur travail.

Les deux variables clefs du succès de la première étape seront la satisfaction des clients, et d'abord des entreprises, vis-à-vis de l'offre construite autour des logiciels libres (c'est une condition nécessaire à court terme), et l'implication effective des entreprises de service et de logiciel dans le développement de projets libres (condition nécessaire à long terme).

Nous avons donc enquêté auprès des entreprises proposant des services basés sur des produits libres pour savoir si cette diffusion était en cours et si ses caractéristiques correspondaient bien à nos hypothèses. Ce sont les résultats de cette enquête que nous présentons dans le chapitre 4. Dans le chapitre 5, et parce qu'il s'agit d'une étude essentiellement prospective, nous avons choisi d'utiliser l'outil de la modélisation pour apporter une réponse à la question de l'adoption des systèmes d'exploitation et des outils libres par les utilisateurs naïfs. Après avoir montré de façon littéraire quels pouvaient être les paramètres influençant cette diffusion, nous tenterons d'évaluer l'impact de ces paramètres et la façon dont ils vont effectivement agir sur le processus de diffusion.

Quatrième Chapitre

Une étude empirique de l'interaction entre utilisateurs marchands et producteurs de solutions libres.

NOUS avons montré que la première étape de la diffusion du Libre était sa diffusion dans les entreprises, ce que nous avons appelé «les utilisateurs marchands». C'est aussi à cette étape que devraient se développer les entreprises productrices de technologies d'utilisation libres. Il s'agit du passage de la phase 1 à la phase 2 de l'organisation Libre, de la phase de développement coopératif interne à un groupe d'utilisateurs-experts à la phase d'utilisation de ces logiciels par les utilisateurs sophistiqués, avant les utilisateurs naïfs, grâce à leur passage dans les organisations.
La première variable clef de cette diffusion est l'adoption par les clients (et d'abord des entreprises) des logiciels libres dans les marchés des logiciels techniques, des logiciels de réseau ou des «plates-formes» (c'est une condition nécessaire pour initier la diffusion du libre hors de sa sphère originelle.) La deuxième variable est la création d'entreprises de service qui permettent de mutualiser les coûts de développement des logiciels libres et leur adaptation aux besoins des utilisateurs sophistiqués et des utilisateurs naïfs (condition nécessaire pour l'efficacité de long terme de l'organisation Libre).

Nous voulons, dans ce chapitre, vérifier la réalité de la diffusion du Libre, ainsi que ses caractéristiques. Pour cela, nous nous appuierons principalement (mais pas exclusivement) sur une enquête que nous avons menée auprès des entreprises françaises qui faisaient publicité d'utiliser des logiciels libres dans leurs activités commerciales. Nous avons cherché à savoir quelle était cette activité, comment elle était vendue aux clients et perçue par eux[1]. Nous détaillerons, dans la partie 1, les modalités de cette enquête.

Nous nous sommes d'abord intéressés au comportement des entreprises utilisatrices de logiciels libres. Nous avons cherché à vérifier trois hypothèses et nous présentons les résultats obtenus dans la partie 2. Nous allons d'abord voir que ces logiciels sont bien adoptés par les entreprises clientes des producteurs qui nous ont répondu pour des considérations techniques, parce qu'ils apportent un meilleur service à leurs utilisateurs. Ensuite, nous avons montré au chapitre 2 que, théoriquement, la diffusion devrait se faire d'abord chez les grands comptes. On verra que les résultats de notre enquête sont en accord avec cette hypothèse. Enfin, d'après nos hypothèses, ces entreprises devraient faire appel à des sous-traitants pour des problèmes techniques pointus. Nous vérifierons que

[1] Le détail des réponses se trouve en Annexe B.

c'est aussi le cas pour le entreprises que nous avons interrogées.

Nous nous sommes ensuite intéressés aux caractéristiques des producteurs, à leur capacité à répondre efficacement et durablement aux demandes des utilisateurs. Nous présenterons nos résultats dans la partie 3. Nous avons cherché essentiellement à vérifier deux hypothèses, l'implication des producteurs dans la production de logiciels libres et la rentabilité de leurs activités. Nous avons dit dans les chapitres précédents que, pour que l'organisation libre soit pérenne dans le temps, il fallait que les producteurs participent à la production de logiciel libre. Nous avons ensuite montré qu'ils avaient des incitations à le faire, incitations principalement de l'ordre du signal (signal de qualité, signal de garantie d'un service efficace, etc.) Nous étudierons ces deux points avant de vérifier la rentabilité de ces activités.

4.1 Modalité de l'enquête.

4.1.1 Panel et questionnaire.

Pour réaliser cette enquête, nous avons interrogé des entreprises françaises ou ayant une activité commerciale en France qui font publicité d'offrir des solutions utilisant du logiciel libre[2]. Nous nous sommes adressés exclusivement à des entreprises du secteur informatique, c'est-à-dire des constructeurs d'ordinateurs ou de matériels associés (périphériques, routeurs, etc.), des producteurs de logiciel des entreprises de service en informatique, etc.

Pour réaliser notre échantillon, nous avons sélectionné un «panel» d'entreprises ayant participé à des manifestations autour du logiciel libre («Autour du Libre» et «Linux Expo») ou s'étant inscrites sur des sites répertoriant les entreprises qui ont une activité commerciale utilisant des logiciels libres (voir http ://www.aful.org). Cela nous a donné une liste de 134 entreprises. Nous avons complété cette liste en annonçant notre enquête dans la liste de diffusion des professionnels de l'Aful (Association Française des Utilisateurs de Linux et des Logiciels Libres).

[2]Il était plus difficile de s'adresser aux entreprises utilisatrices de libre car celles-ci ne se font pas obligatoirement connaître. Cependant, il serait intéressant de réaliser une enquête auprès des grandes entreprises françaises pour étudier la réalité de la diffusion de ces logiciels chez elles (par exemple parmi les membres du CIGREF, qui a réalisé une étude sur l'usage de Linux, voir http ://www.cigref.fr).

Il est alors difficile de savoir le nombre exact des entreprises qui ont été approchées ; cependant, considérant que la «Linux Expo» est le plus grand salon professionnel dans le domaine et que l'Aful est l'association professionnelle la plus connue, nous estimons avoir contacté la plupart des entreprises travaillant significativement dans ce secteur.

Il a été demandé à toutes ces entreprises de répondre à un questionnaire en ligne[3] ; cherchant à faire une enquête quantitative, nous avons limité au maximum les questions ouvertes.
Nous avions regroupé les questions en trois catégories : celles qui concernent l'entreprise (taille, nombre de salariés, ancienneté) et son domaine d'activité (type de produits vendus, clientèle principale, etc), celles qui concernent la façon dont elle utilise les logiciels libres pour construire son offre (type de logiciels utilisés, type de produits construits avec ces logiciels) et la perceptions qu'elle a du marché du logiciel, de ses opportunités, de ses obligations, etc. Le compte-rendu des réponses ne suit donc pas l'ordre des questions.

4.1.2 Résultats obtenus.

Nous avons reçu, au 11 avril 2001, 36 réponses (24 appartenant au panel initial et 12 hors panel). Il est évidemment difficile de savoir si ces entreprises sont représentatives des entreprises intervenant dans le secteur des logiciels libre. Ce sont de petites structures : la moitié a moins de 25 salariés, plus des trois quarts moins de 50.

Si l'on met à part quelques réponse qui émanent des grands groupes (un grand constructeur informatique nous a répondu), on peut dire que ce sont plutôt des entreprises représentatives des nombreuses P.M.E. qui entrent sur le marché de l'informatique, dans le service ou dans la production de logiciel. Ce qui ne veut pas forcément dire que ces entreprises soient récentes : à notre surprise, dans les entreprises qui ont répondu, seulement un tiers a moins de trois ans, un peu moins de la moitié a même plus de cinq ans.

Ce sont donc des entreprises relativement établies dans leur métier, qui ont aussi une certaine expérience du marché. C'est plutôt encourageant pour le libre, car cela veut dire

[3]Ce questionnaire est disponible à l'adresse :
http ://www-eco.enst-bretagne.fr/cgi-bin/nj.pl

Figure 4.1 — Nombre de salariés début 2001.

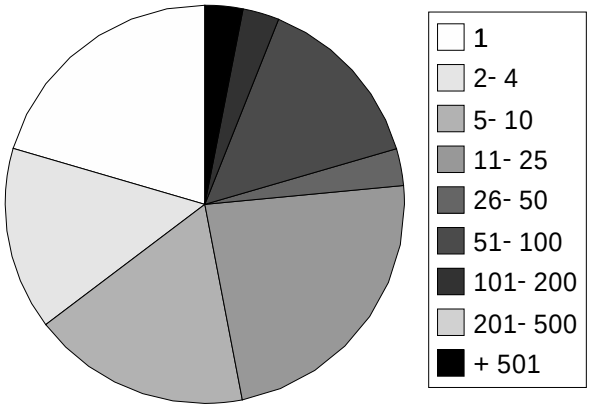
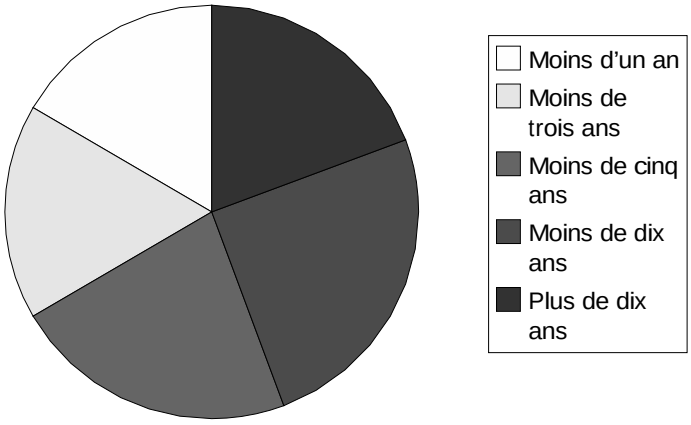


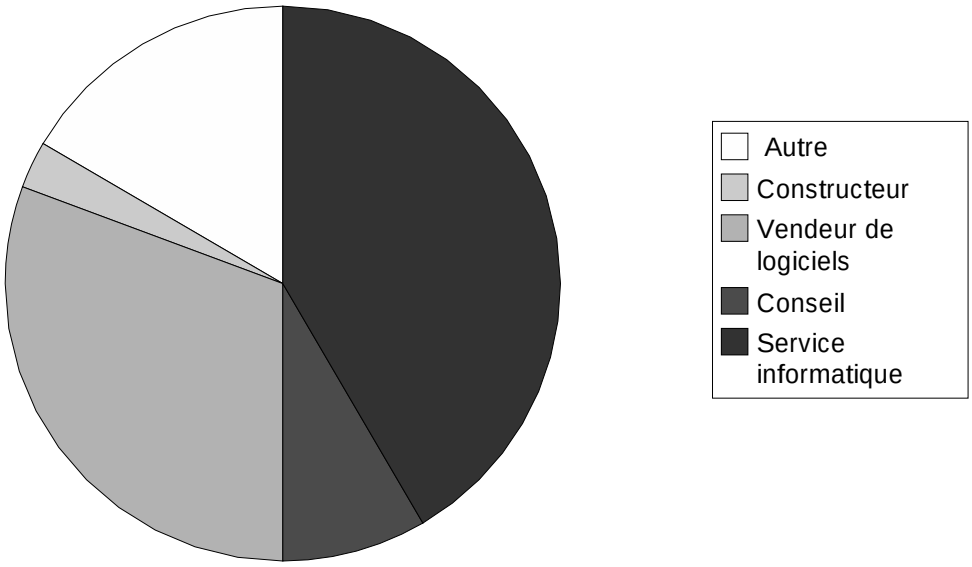
Figure 4.2 — Âge des entreprises ayant répondu.



que des entreprises déjà en place ont pu l'intégrer dans leurs offres, ce qui augmente la crédibilité des ces produits (c'est alors l'image de l'entreprise qui sert comme gage de qualité).

Mis à part les constructeurs et les assembleurs peu représentés (une réponse), nous avons pour les autres professions de l'informatique un nombre satisfaisant de réponses : une quinzaine pour les entreprises de conseil et de service, une dizaine pour les vendeurs de logiciels (les autres réponses, qui forment la catégorie «autres», regroupent des entre-

Figure 4.3 — Activité principale des entreprises ayant répondu.

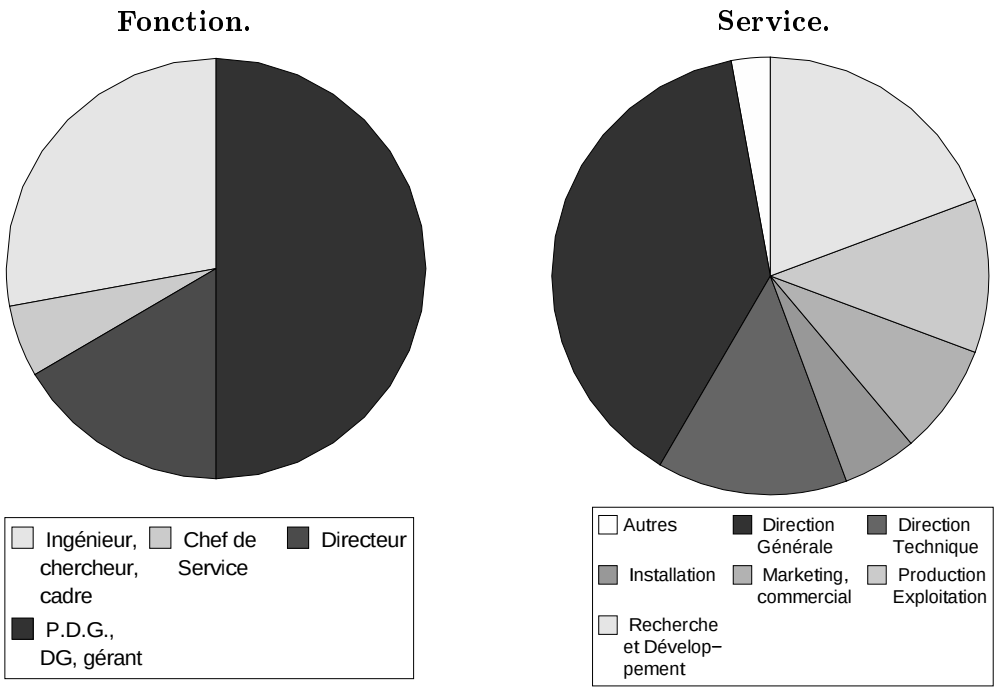


prises de formation, quelques entreprises qui font du service dans les secteurs connexes à l'informatique comme les télécommunications, en plus de celles qui font de l'hébergement et qui auraient pu être intégrées dans les entreprises de service).
Les réponses semblent couvrir relativement bien le spectre des métiers de l'informatique, même si leur nombre reste faible.

Enfin, nous pouvons estimer que la qualité de nos réponses est bonne et qu'elles reflètent bien la stratégie des entreprises : toutes les personnes interrogées ont répondu à toutes les questions (sauf celles sur les données financières de l'entreprise) et ces personnes occupent des postes de direction dans plus des trois-quarts des cas (le reste des réponses venant de cadres), à la direction générale ou à la direction technique dans plus de la moitié des cas.

Nous allons détailler ces réponses en commençant par les entreprises utilisatrices du libre, les clientes de ces producteurs.

Figure 4.4 — Fonction et service des personnes qui ont répondu.



4.2 Les entreprises utilisatrices du libres.

Nous voulions vérifier trois points :

- que ces utilisateurs sont plutôt des grands comptes ;
- que ces entreprises adoptent les logiciels libres pour des raisons techniques, parce qu'ils leur apportent un meilleur service ;
- qu'enfin, ces entreprises font appel à des sous-traitants pour des problèmes techniques pointus.

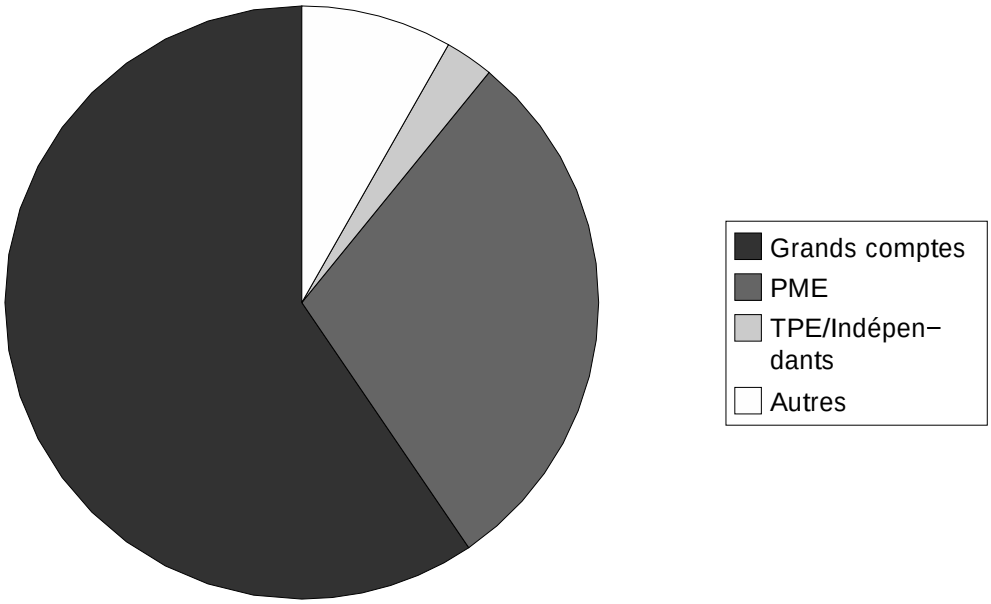
Les réponses semblent confirmer ces trois points.

4.2.1 Les utilisations des logiciels libres : comme objets techniques dans les grands comptes.

Les utilisateurs des logiciels libres.

Nous avons montré qu'une condition nécessaire de la diffusion des logiciels libres était que ces logiciels soient adoptés par les grands comptes. C'est en leur sein que peut se faire la rencontre des demandes des utilisateurs designers et des utilisateurs experts, voir naïfs. Ce sont aussi eux qui ont les capacités financières pour financer les développements, les adaptations nécessaires à l'utilisation des logiciels libres dans les entreprises. Les clients des entreprises «libres» sont quasiment exclusivement de grosses organisations (grands comptes ou P.M.I.) et très majoritairement des grands comptes.

Figure 4.5 — Répartition des marchés commerciaux autour du libre.



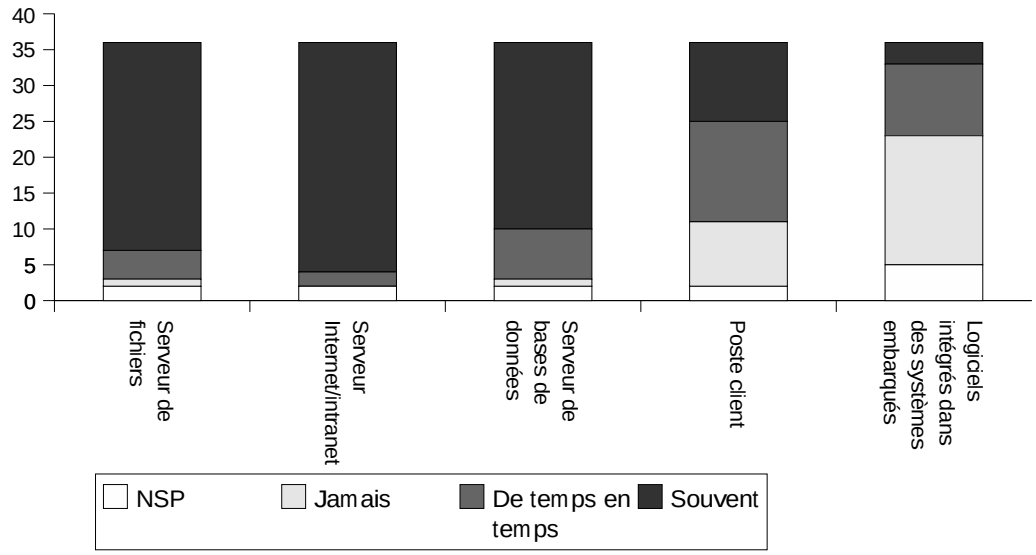
Et elles utilisent les logiciels libres car ce sont des objets techniques efficaces.

L'intérêt pour le Libre : d'abord un intérêt pour les objets techniques produits.

Parce que le logiciel libre arrive dans les entreprises avec Internet, il semble logique que ce soit le principal marché des fournisseurs de solutions. Nous avons aussi émis l'hypothèse que ces logiciels devraient se diffuser par proximité technologique, vers les serveurs de

fichier, puis les serveurs de bases de données, avant de conquérir les autres domaines de l'entreprise et notamment les postes de bureautique. On constate effectivement ce phénomène ; plus surprenante est sa vitesse : la diffusion vers les postes de travail semble déjà en cours, alors que les développements autour des serveurs de base de données sont déjà presque aussi courants que ceux autour d'Internet.

Figure 4.6 — Types de produits construits avec du libre.



C'est peut-être un signe de la diffusion rapide des logiciels libres.
Mais il faut rester prudent car on a peut-être affaire à un artefact de l'enquête : parce que nous ne nous sommes adressés qu'aux entreprises spécialisées dans le logiciel libre, il peut y avoir une sur-représentation des projets innovants, des projets d'évaluation dans les projets réalisés par celles-ci par rapport à l'activité réelle autour des logiciels libres.

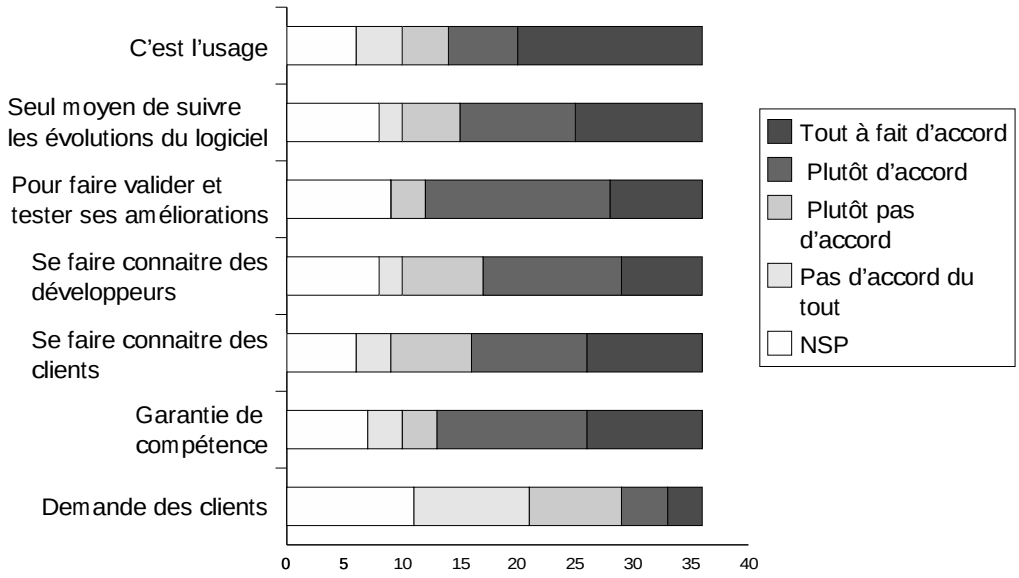
De la même façon, le faible nombre de projets autour des logiciels embarqués tient sans doute plus au faible nombre d'entreprises travaillant dans le domaine qu'à un manque d'activité (lorsqu'on les interroge sur les opportunités commerciales, ces professionnels répondent d'ailleurs que le marché des logiciels embarqués est un des plus prometteurs, voir le paragraphe suivant)[4]. Par contre, si le logiciel libre semble bien se diffuser dans les

[4]De plus, les dernières enquêtes sur le sujet montrent bien que les logiciels libres, et particulièrement Linux, sont de plus en plus utilisés dans ce domaine, voir :
http ://www.idg.net/go.cgi ?id=538524.

entreprises, nous allons voir qu'il le fait essentiellement grâce à ses qualités techniques et non pas grâce à ses qualités d'ouverture.

Les logiciels libres ne sont pas demandés en tant que tels par les clients : nous l'avons demandé à deux endroits (figure 4.7 et figure 4.10) et par deux fois les entreprises nous ont répondu que ce n'était pas le cas.

Figure 4.7 — Raisons pour lesquelles il faut contribuer à des projets libres.



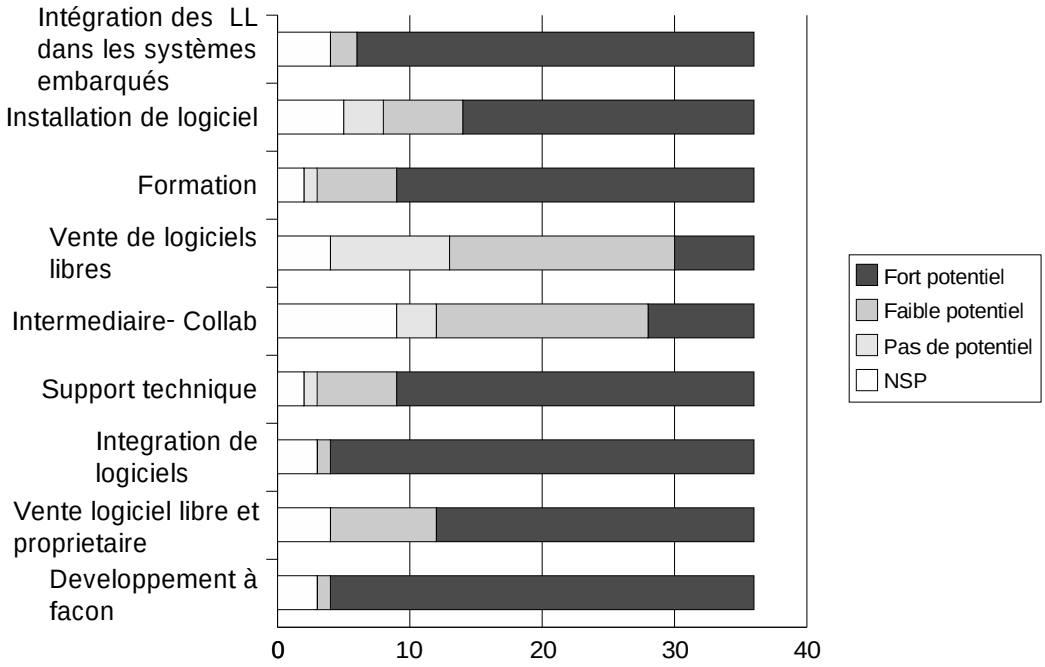
Il ne semble pas que le côté stratégique de l'indépendance vis-à-vis des fournisseurs soit l'argument principal de vente. Ce sont les caractéristiques techniques de ces logiciels, leur adaptabilité plus que le côté libre, qui intéresse les clients. Nous rappellerons cependant que les constructeurs comme IBM, Dell ou HP ont intégré une offre autour de Linux car c'était une demande de leurs clients. S'ils ne demandent pas d'utiliser des logiciels libres, les clients demandent donc bien certains logiciels, libres, qui sont devenus des références, comme Apache, Sendmail ou Linux.

D'autres réponse nous font penser que ces logiciels sont d'abord pour les clients des objets techniques :

– les entreprises contribuent à des projets de développement en libre pour des raisons techniques : en dehors des raisons culturelles (nous avons dit qu'elles étaient im-

portantes pour renforcer la cohésion de l'organisation mais qu'elle ne suffirait pas à assurer le développement), les entreprises nous ont répondu que c'était le seul moyen de suivre le projet et une manière de faire accepter leurs contributions ;

Figure 4.8 — Potentiel de différentes activités autour du libre.



– ce sont aussi les services techniques qui ont le plus de potentiel de développement pour les entreprises interrogées : développements à façon, support technique, intégration de logiciels et formation, si on laisse de côté l'utilisation des logiciels libres dans les systèmes embarqués.

Les logiciels libres sont bien utilisés comme des outils techniques adaptables et c'est sans doute pour cela qu'ils sont acceptés par les clients, à défaut d'être demandés.
C'est heureux dans le sens où nous avons dit que le logiciel libre ne s'imposerait que s'il apporte une satisfaction supérieure en terme d'utilité intrinsèque.
Mais cela veut aussi dire que, bien qu'il se diffuse dans l'entreprise, cette diffusion reste plutôt cantonnée à des applications techniques, pour des clients experts ou sophistiqués. On est encore dans le passage de l'étape 1 à l'étape 2 ou au début de l'étape 2 de la diffusion des logiciels libres.

C'est un passage obligé, mais cela veut aussi dire que les logiciels libres sont toujours des

objets nouveaux pour les entreprises, qui les jugent d'abord sur leur efficacité immédiate. Si elles ne dépassent pas ce stade en prenant en compte l'implication des producteurs dans la sélection de ces producteurs et en finançant des développements, il n'est pas sûr que les logiciels libres se diffusent hors de la sphère technique initiale. Car, nous l'avons écrit dans le chapitre 2, c'est à cette étape que doivent se construire les relations de service, l'utilisation de l'organisation libre comme organisation de normalisation et d'assurance qualité.

Nous avons aussi dit dans le chapitre 3 que c'était à cette étape que devrait se décider l'organisation de la filière libre et notamment le fait de savoir si le système d'exploitation est plutôt un composant, fourni par une entreprise ou plutôt un assemblage de composants, assemblage réalisé à la demande par des «architecteurs». Nous n'avons pas de réponse à cette question ; nous avons bien interrogé les entreprises sur le fait qu'elles utilisent une ou plusieurs distributions de Linux, mais sans demander s'il s'agissait de distributions «commerciales» (RedHat, SuSE, ...) ou de distributions «libres» (Debian). Il sera intéressant de poser ces questions si les logiciels libres se diffusent effectivement, si se construisent effectivement des relations de service basées sur une organisation libre d'assurance qualité.

Il semble bien que cette construction avance, même si les clients ne la valorisent pas encore explicitement puisqu'ils ne demandent pas explicitement l'utilisation de produits libres. En effet, les relations clients-fournisseurs sont bien des relations de services. Les clients s'adressent aux fournisseurs pour leur expertise sur les logiciels.

4.2.2 La relation clients-fournisseurs : une relation de service.

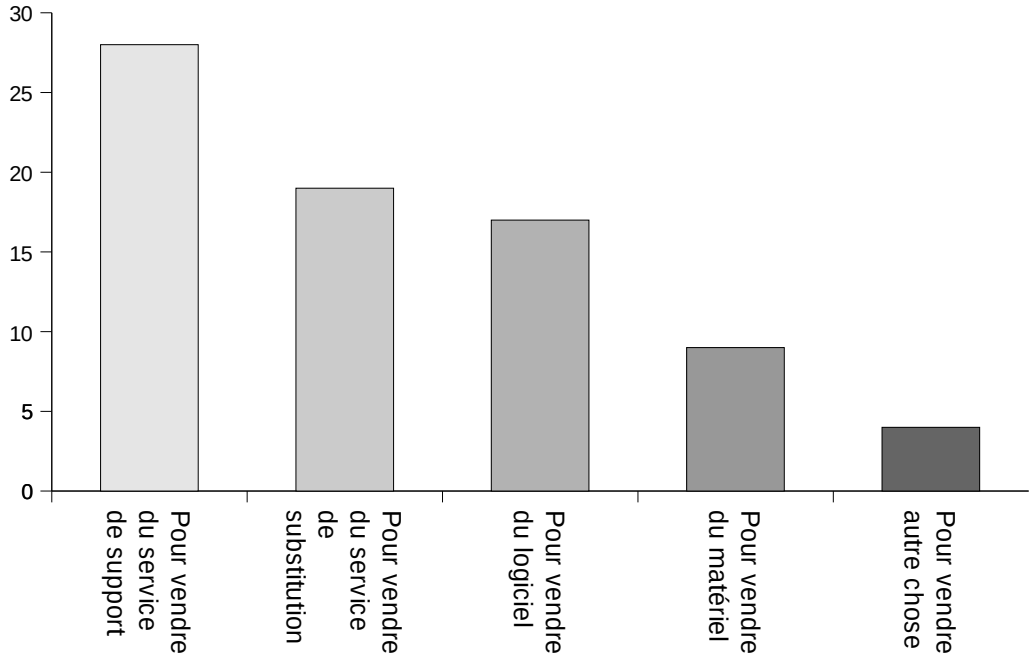
Vendre du Libre, c'est vendre du service.

Nous avons supposé que le libre devait d'abord être un support pour vendre du service.

Les entreprises nous ont répondu qu'elles vendaient d'abord du service de substitution ou service de support. Pourtant 1/3 d'entre elles se présentent comme des vendeurs de logiciels. Mais parmi ces entreprises, seules trois affirment ne vendre que du logiciel (voir les détails des réponses dans l'annexe).

De plus, nous l'avons vu (figure 4.8), les activités qui ont le plus de potentiel commercial

Figure 4.9 — Utilisation du libre par grandes catégories commerciales.



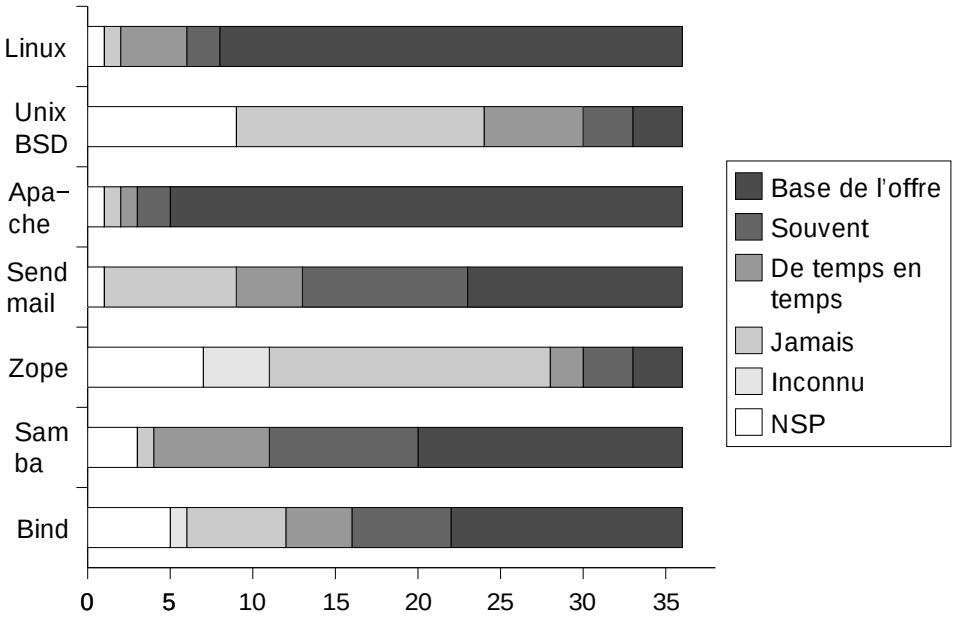
(Les réponses peuvent être multiples).

sont toutes des activités de service ; la seule pour laquelle les entreprises estiment qu'il n'y a pas de potentiel est l'activité de vente de logiciels libres.

À partir de là, nous pouvons proposer une explication au fait que les producteurs ne voient pas le libre comme adapté pour développer des logiciels standards, alors que ses plus grands succès sont quand même la création des standards d'Internet comme Apache ou Sendmail et que Linux cherche à s'imposer comme un standard dans les systèmes d'exploitation. Les logiciels les plus utilisés pour construire les offres commerciales sont d'ailleurs les plus connus : Linux, Apache et Sendmail. Les Unix de type BSD, qui sont pourtant particulièrement bien adaptés pour construire des serveurs Internet sont très peu utilisés, par exemple.

Il se peut que les entreprises qui ont répondu assimilent la notion de standards à des progiciels fermés tels ceux de Microsoft, ou même d'Oracle. Il est plus difficile d'adapter ces logiciels à la demande alors que c'est la raison pour laquelle les utilisateurs producteurs ont développé l'offre libre. La construction de logiciels standards est pourtant importante dans l'optique de leur diffusion chez les utilisateurs naïfs. C'est donc un point dont il

Figure 4.10 — Logiciels utilisés pour construire l'offre commerciale.

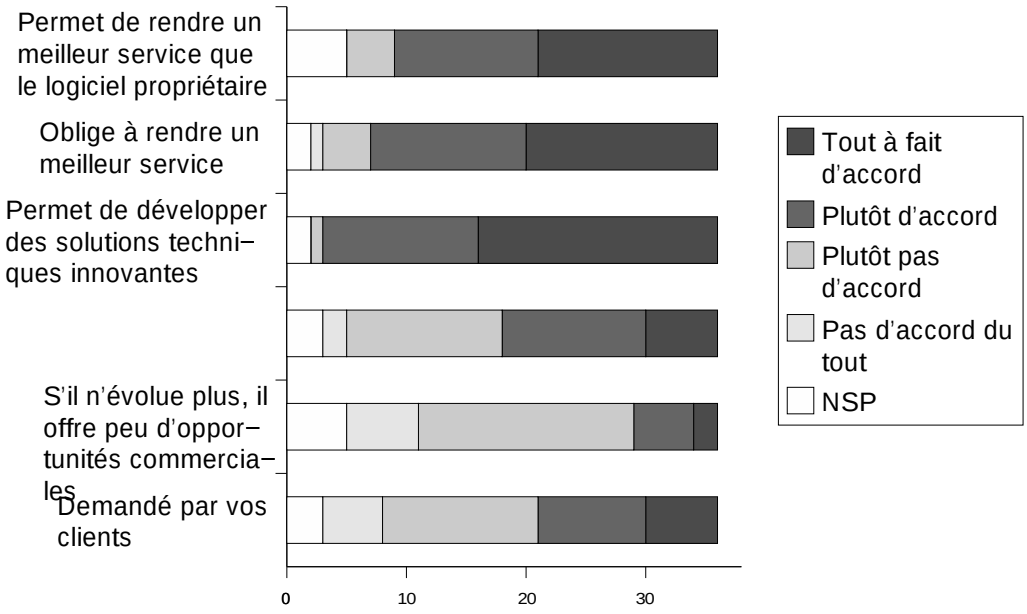


faudra suivre l'évolution dans les années futures.

Mais c'est la deuxième étape de la diffusion, la première étant que le Libre s'impose dans la production d'objets techniques dans les organisations, celle qui nous occupe ici. Et, de ce point de vue là, le libre semble jouer son rôle : il permet aux fournisseurs de rendre un meilleur service, mieux, il les y pousse (c'est au moins leur point de vue, voir figure 4.10).

C'était une de nos hypothèse : nous avions supposé (chapitres 2 et 3) que le libre, parce qu'il rendait les utilisateurs plus indépendants de leurs fournisseurs, les obligeait à travailler mieux, faisant baisser le hasard moral. Bien qu'il y ait sûrement une part de prosélytisme de la part des producteurs, c'est un indice du bon fonctionnement du système. Il faudrait bien sûr prolonger cette enquête du côté des utilisateurs, des clients, pour s'en assurer. On peut se demander s'il fonctionne aussi comme système permettant d'abaisser la sélection adverse, de construire des éléments d'assurance-qualité.

Figure 4.11 — Points de vue sur le libre.



Le Libre comme système assurance-qualité.

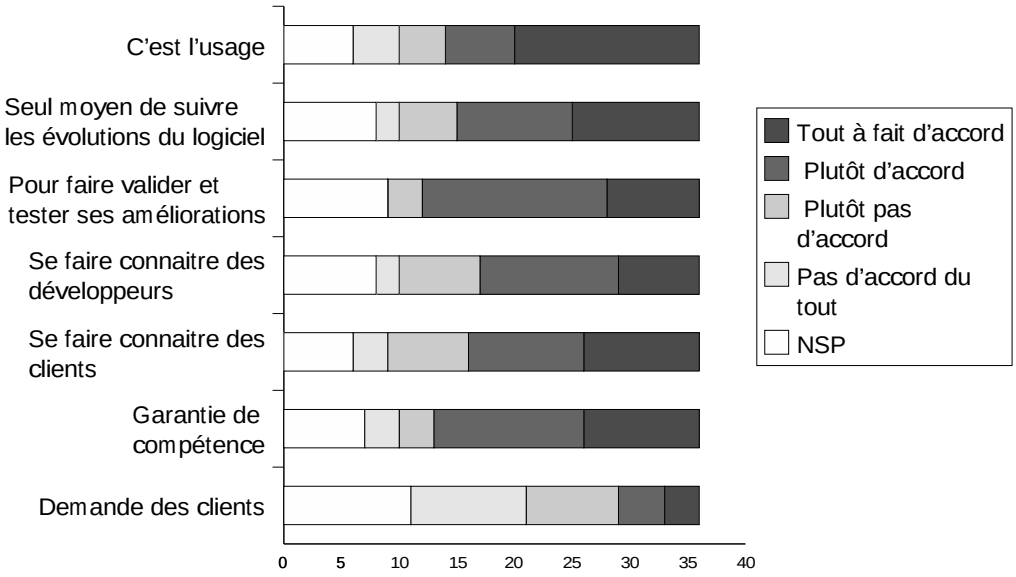
Ce système d'assurance-qualité semble fonctionner aussi, avec les restrictions qui ont été évoquées ci-dessus : si les clients ne demandent pas l'utilisation du libre, ils sont sensibles au fait que les entreprises qui leur proposent ces logiciels contribuent à leur développement. C'est, semble-t-il, une garantie de compétence et un moyen de se faire connaître pour les fournisseurs qui contribuent.

Nous verrons (figure 4.14) que les entreprises ne semblent pas faire qu'annoncer l'importance de leur implication dans des projets libres, elles la mettent effectivement en pratique : pour plus des trois quarts des entreprises ayant répondu, le temps de travail laissé libre aux développeurs pour participer à des projets libres dépasse les 5 %, une moitié affirmant même que ce temps dépasse les 10 %.

Le Libre, une coproduction industrielle de logiciels.

Cette implication grandissante des entreprises (utilisatrices et productrices) dans la coproduction des logiciels qu'elles utilisent dans leurs activités commerciales est sans doute

Figure 4.12 — Raisons de contribuer au développement du Libre.



plus visible au niveau global. Nous prendrons ici l'exemple de l'évolution des activités professionnelles des contributeurs au noyau Linux (figure 4.13).

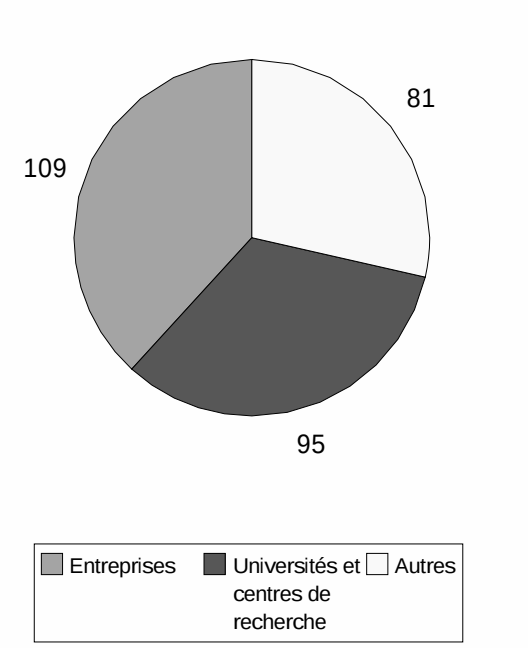
La part des entreprises (producteurs et utilisateurs) augmente, parce que de plus en plus des anciens contributeurs travaillent pour une entreprise (+10 entre les deux noyaux, contre -10 pour les personnes appartenant à une université ou un centre de recherche), et surtout parce que les nouveaux contributeurs appartiennent majoritairement à une entreprise.

Conclusion.

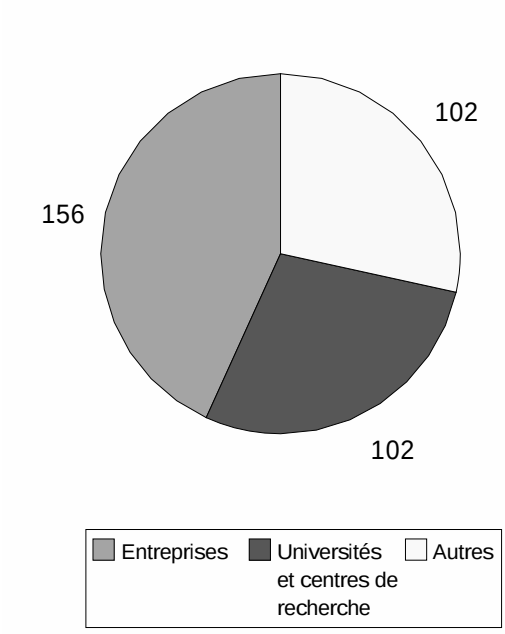
On peut donc conclure, à la lecture de ces quelques résultats, que, au moins en France et parmi les clients des entreprises de notre panel, la diffusion du Libre a bien commencé. Que cette diffusion correspond bien aux caractéristiques de la fin de la phase 1 et du début de la phase 2, l'utilisation des logiciels libres en tant qu'objets techniques, plus facilement adaptables aux besoins spécifiques de l'entreprise. Qu'enfin elle est accompagnée de la création d'entreprises de services qui proposent leur expertise pour adapter ces logiciels aux besoins des entreprises.

Figure 4.13 — Évolution des fonctions des contributeurs au noyau Linux.

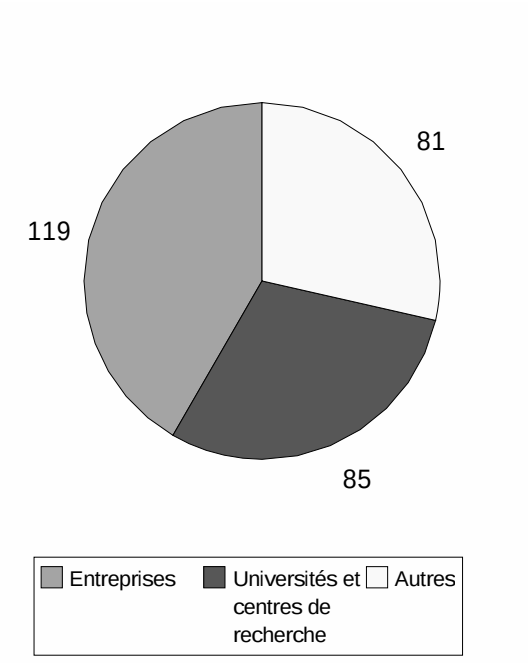
Employeurs des contributeurs principaux au noyau Linux 2.2.



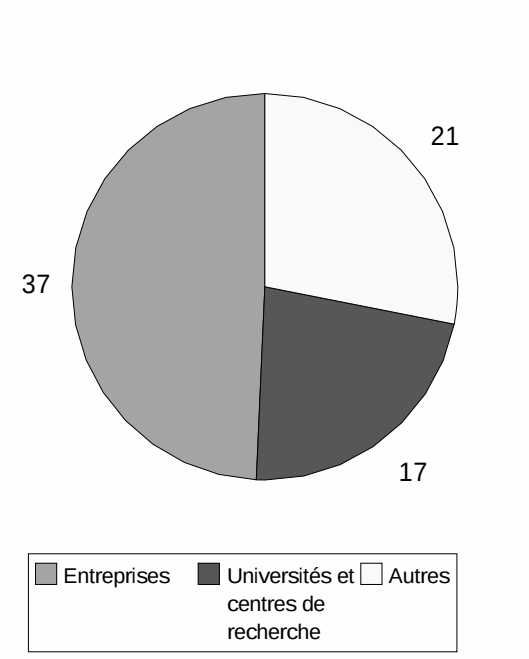
Employeurs des contributeurs principaux au noyau Linux 2.4.



Employeurs des contributeurs au noyau Linux 2.2 au moment du noyau 2.4.



Employeurs des nouveaux contributeurs au noyau Linux.



source : fichier «credits» des noyaux 2.2 et 2.4.

Dans la troisième partie, nous allons étudier le comportement de ces entreprises, voir si leur comportement (implication dans les projets libres, positionnement) correspond bien aux hypothèses que nous avons avancées dans les chapitres précédents.

4.3 Étude des caractéristiques et de l'implication des producteurs.

Nous avons cherché à étudier si l'engagement des fournisseurs sur les offres libres était pérenne. Pour cela, nous avons étudié deux choses : le fait que ces fournisseurs s'impliquent effectivement dans le développement des projets libres et dans la construction de ce que nous avons appelé un système d'assurance qualité, d'une part, et le fait que cet engagement permette de créer des activités économiques viables, d'autre part.
Nous allons voir que les fournisseurs apparaissent un peu en avance par rapport à leurs clients sur la construction du système d'assurance qualité. Par contre, il est difficile de conclure avec les données que nous avons sur la pérennité de leurs modèles économiques et, partant, sur celle du modèle économique du libre. Une fois de plus, nous sommes pénalisés par la jeunesse du phénomène, qui ne permet d'avoir que des indices sur son évolution.

4.3.1 Implication et positionnement des entreprises du Libre.

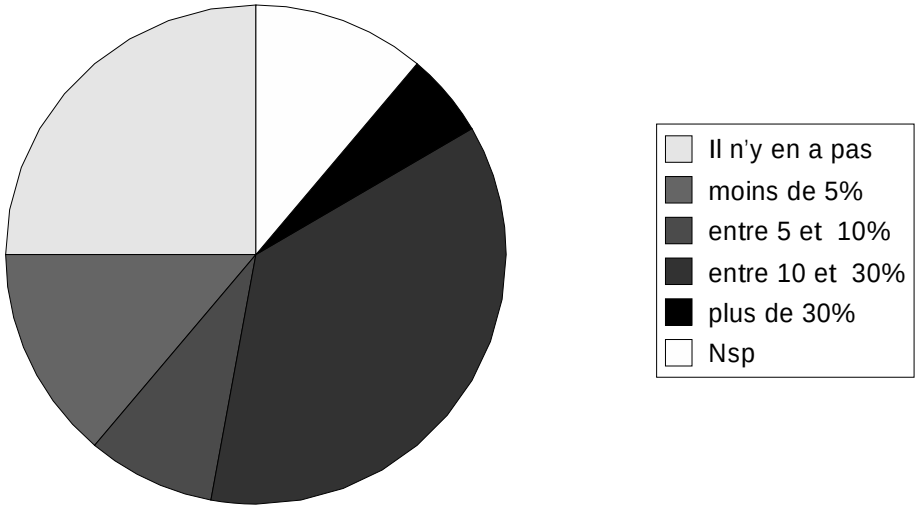
L'implication dans la dynamique libre.

Les entreprises du Libre apparaissent très impliquées dans la diffusion et la préservation du modèle ouvert du libre.
Nous avons déjà cité l'exemple d'ACT Europe, entreprise qui contrôle le compilateur du langage Ada 95 Gnat[5], ainsi que celui de l'entreprise Digital Creation, qui contrôle le logiciel Zope[6], sans parler des écrits de RedHat dans les documents publiés lors de son entrée en bourse.
Les entreprises ne semblent pas faire qu'annoncer l'importance de leur implication dans

[5]Pour mémoire : Gasperoni [2001].
[6]Pour mémoire toujours : le site du logiciel, http ://www.zope.org et la position de Digital Creation vis-à-vis de ce logiciel, http ://www.digicool.com/solutions/.

des projets libres, elles la mettent effectivement en pratique. Leur activité de recherche est importante, surtout si on la compare aux habitudes du secteur des services (moins de 1 % du chiffre d'affaire, voir Genthon [2001]). Pour plus des trois quarts des entreprises ayant répondu le temps laissé libre dépasse les 5 %, une moitié affirmant même que ce temps dépasse les 10 % du temps de travail.

Figure 4.14 — Temps laissé libre aux employés pour participer à des projets libres.

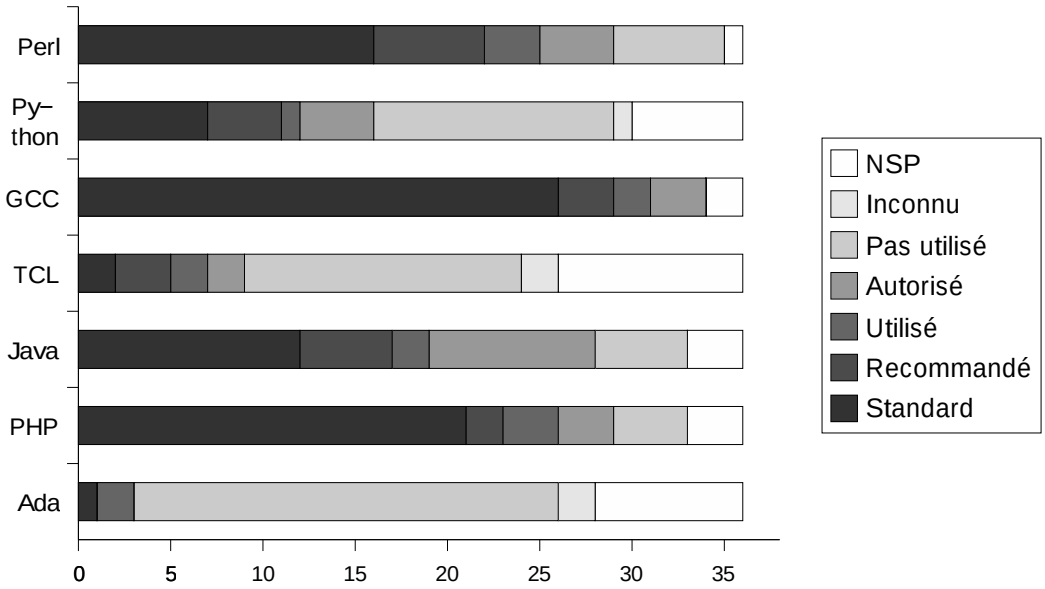


Un positionnement basé sur la maîtrise d'outils techniques.

Il est remarquable de noter que ces entreprises ont collectivement une bonne vision de leur marché, des difficultés qu'elles peuvent rencontrer et des raisons qui poussent à publier en libre un programme. À la question ouverte des critères à considérer dans la mise en libre d'un logiciel, elles donnent des réponses variées, mais qui regroupées, soulignent les principales difficultés de l'exercice : la nécessité de trouver une base d'utilisateurs développeurs, l'intérêt stratégique du logiciel pour des concurrents de l'entreprise, la collectivisation des efforts de développement, etc.
Mais elles se positionnent sur des marchés techniques, nous l'avons vu, et sont attentives aux évolutions technologiques de leur marché.

Si on trouve sans surprise le langage libre Perl comme un des langages utilisé par le plus grand nombre d'entreprise, le langage qui arrive en tête après le C (et son compilateur libre

Figure 4.15 — Utilisation des langages de programmation.



GCC) est PHP, un langage de génération dynamique des pages HTML, langage inventé en 1998. On ne peut pas expliquer cela uniquement par le fait que ces entreprises auraient été développées par des techniciens qui cherchaient à promouvoir le libre à tout prix. Elles semblent avoir une vraie stratégie et une bonne vision de leur marché. Si certaines entreprises font profession de n'utiliser que du logiciel libre (comme Alcôve), c'est pour des raisons stratégiques (pour obtenir un signal maximal sur leur implication dans le service au client). D'autres sont prêtes à utiliser des standards du marché à condition qu'ils soient relativement ouverts (comme Java, voir ci-dessous), s'ils sont demandés par les clients ou s'ils sont efficaces pour répondre au problème posé.
Il nous semble que ces entreprises ont bien une stratégie de positionnement sur de marchés techniques, des marchés d'expertise.

On peut alors se poser la question de savoir si émergent bien les deux métiers que nous avons mis en avant (chapitre 2) comme caractéristiques de l'organisation Libre, le métier de fabricant de composant et le métier d'architecte, d'assembleur de composants libres.

Spécialiste ou généraliste, deux métiers qui émergent ?

Signalons, d'abord, un résultat qualitatif intéressant : le grand constructeur généraliste qui nous a répondu a indiqué, comme autre raison pour contribuer à des développements logiciels, que c'était «une évolution irréversible du développement logiciel». Ce n'est évidemment qu'un point de vue, mais il vient renforcer l'impression que les grands généralistes comme IBM, HP ou Compaq voient bien dans le libre l'avenir de l'informatique et le secteur dans lequel il faut actuellement investir.

Il est pourtant difficile de faire émerger de nos résultats agrégés une réponse à la question de l'organisation des producteurs de libre. Tout juste pouvons-nous constater que les entreprises utilisent beaucoup de logiciels, qu'une des activités principales d'une partie d'entre-elles consiste à amalgamer des logiciels.

Nous savons que dans notre panel, qui ne concerne que des entreprises travaillant en France, au moins deux entreprises sont spécialisés dans l'installation, le paramétrage et la maintenance d'un logiciel : l'une autour du compilateur GNAT (ACT Europe), l'autre autour du logiciel Zope (Nuxeo). Il est certain que la plupart des entreprises de ce type (Cygnus, Digital Creation, Scriptics Corp., Covalent technology, Sendmail Inc., Zend, ACT, etc.) ont été créées aux États-Unis.

Faut-il y voir une illustration de la culture industrielle française en informatique qui a privilégié le développement d'entreprises de service ? Peut-être est-ce plutôt dû à la jeunesse du phénomène, à la plus faible habitude des transferts de technologies de la part des laboratoires de recherche. Ou peut-être est-ce simplement dû à notre échantillon, au fait que nous n'avons pas su atteindre ces entreprises. Il faut en tout cas noter qu'une des entreprises phare de cette catégorie est une entreprise française, Matra Datavision, qui propose son logiciel de CAO en libre depuis la fin de 1999[7]. C'est un point qu'il faudra suivre pour estimer la maturité du marché du Libre (et aussi la validité de nos hypothèses).

Un autre point important de la maturité, de la pérennité de cette organisation est la rentabilité des entreprises du marché libre. Ce sont elles, après tout, qui supportent une partie des risques de ce modèle. Là encore, si les résultats sont favorables au modèle Libre,

[7] Voir la présentation qui a été faite de ce cas au workshop du projet «nouvelle économie du logiciel», http ://parmentille.enst-bretagne.fr/~njullien/rntl/worshop1/Matra_Bruno.pdf

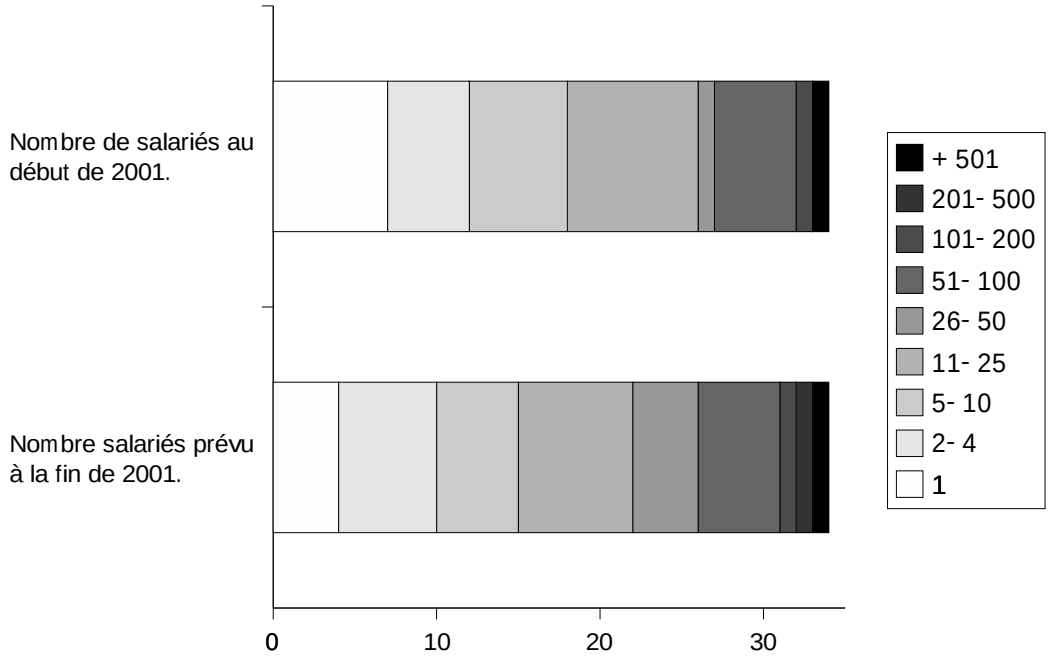
ils sont bien trop partiels pour nous permettre de conclure.

4.3.2 La pérennité des entreprises.

Trois critères ont été utilisés pour évaluer la pérennité de ces entreprises : l'évolution de leur chiffre d'affaires, l'évolution de leurs bénéfices et l'évolution du nombre de leurs employés.

Le premier et le troisième permettent plutôt d'évaluer la croissance de l'activité en général, le second permettant d'avoir une première indication sur la rentabilité des entreprises. Autant les questions sur la masse des employés n'ont pas posé de problèmes, autant les questions financières ont souvent été ignorées.

Figure 4.16 — Évolution du nombre d'employés entre début et fin 2001.



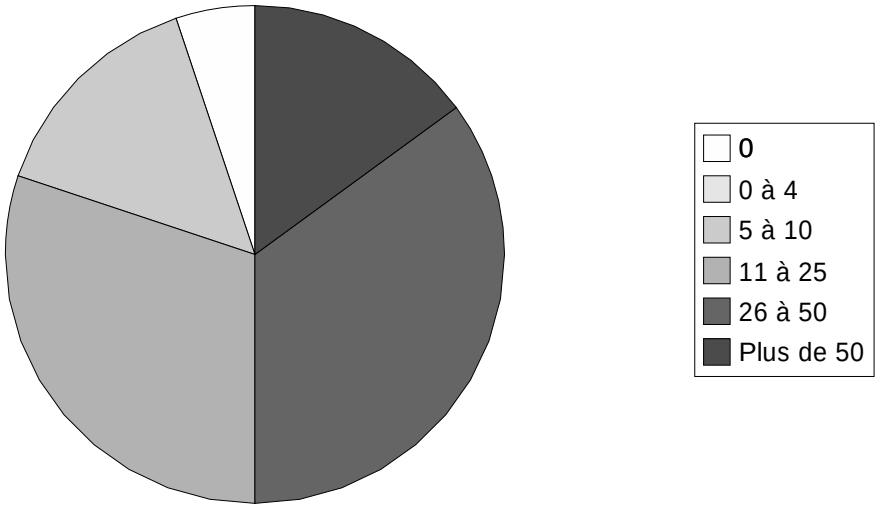
(Sur 34 réponses).

On constate qu'il y a bien une croissance de ces entreprises, même si la vision de cette croissance est en partie masquée par le fait que nous avons pris des intervalles pour définir le nombre des employés.

Mais il y a une baisse nette des catégories inférieures au profit des catégories supérieures.

L'évolution des paramètres financiers apporte une indication supplémentaire de la

Figure 4.17 — Croissance prévue du C. A. entre 2000 et 2001.



(Sur 20 réponses).

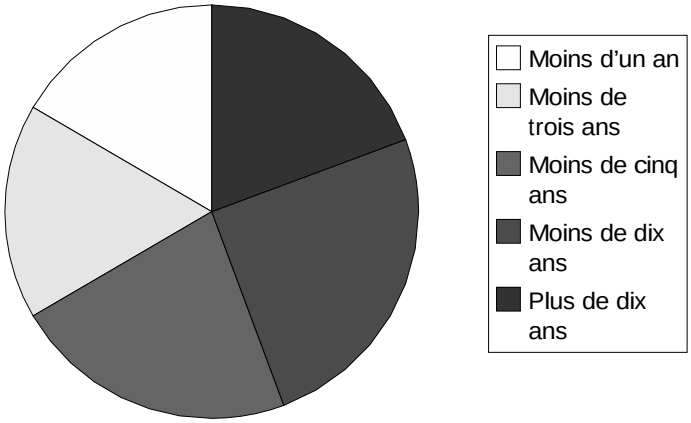
croissance des entreprises, même si les données sont moins nombreuses. Nous n'avons en effet recueilli que 20 réponses concernant l'évolution du chiffre d'affaires.
La population qui nous a répondu affiche des taux de croissance importants puisque la moitié des entreprises espèrent un taux de croissance supérieur à 25 % et surtout plus des 3/4 ont un taux de croissance supérieur à 10 %. À titre de comparaison, on peut rappeler que la croissance moyenne des services informatiques était en 1999 de 13,5 % (Blonde & Aguer [2000]). Il semble bien que le marché du libre profite de la croissance de ces services, progresse même plus vite que la moyenne des entreprises de service[8].

Il ne nous a par contre pas été possible, au vu du nombre des réponses sur les bénéfices (8 réponses exploitables), de conclure sur la rentabilité de ces activités.

Mais nous remarquerons qu'environ deux tiers des entreprises qui nous ont répondu ont dépassé le «cap» des trois ans d'activité (voir figure 4.2). C'est un indice sérieux de la rentabilité de leur activité car on sait que la plupart des entreprises n'atteignent pas cet âge.
Si cette étude mériterait d'être renouvelée afin de suivre les évolutions de ces entreprises, il nous semble que les différents facteurs présentés montrent qu'il existe bien des possibilités

[8]D'autant plus qu'il ne s'agit là que des chiffres d'affaires d'entreprises déjà installées, qu'il faudrait comparer à la croissance du chiffre d'affaire des entreprises du secteur déjà installées et non pas à la croissance du chiffre d'affaire du secteur. Car les nouveaux entrants génèrent un chiffre d'affaire qui est inclus dans la croissance de ce chiffre d'affaires général et qui augmente l'ampleur de cette croissance.

Figure 4.18 — Âge des entreprises ayant répondu.



de développer des activités de service pérennes basées sur des produits libres.

4.4 Conclusion.

On peut conclure de cette étude, qui à cause du faible nombre de résultats doit être vue plutôt comme une étude qualitative du marché du libre en France, qu'il existe bien, aujourd'hui, un marché du logiciel libre, même si ce marché est plutôt dans la phase initiale de son développement.

Nous nous sommes d'abord intéressés au comportement des entreprises utilisatrices de logiciels libres.
Nous avons vu qu'elles adoptaient ces logiciels pour des considérations techniques, pour des utilisations du type serveur de données. Ces adopteurs sont d'abord des grands comptes, ce que nous avons expliqué par le fait qu'ils possèdent les compétences nécessaires pour évaluer ces solutions et financer leur adaptation à leurs besoins. Enfin, ces entreprises font appel à des sous-traitants pour des problèmes techniques pointus.
Ces trois résultats sont en accord avec les hypothèses que nous avions faites aux chapitres 2 et 3 sur la diffusion du libre. Ils semblent indiquer que le passage de la phase 1 à la phase 2 de la diffusion de ces logiciels est en cours.

Sachant cela, nous nous sommes intéressés aux caractéristiques des producteurs, à leur capacité à répondre efficacement et durablement aux demandes des utilisateurs. Nous avons vu, comme nous le supposions dans les chapitres 2 et 3, que ces producteurs étaient des producteurs de service, qui participent à la production de logiciel libre parce que cela leur permet de maintenir leur expertise sur ces logiciels et de la signaler à leurs clients.

Par contre les clients ne semblent pas encore sensibles à la garantie de pérennité que devrait leur apporter l'ouverture des logiciels et il est trop tôt pour se prononcer sur la viabilité économique de ces producteurs. Nous ne pouvons qu'appeler à renouveler cette enquête dans un ou deux ans et à la compléter par une enquête similaire auprès des grands clients de l'informatique. Le marché sera sans doute plus mûr et les résultats plus complets.

Cinquième Chapitre

Une modélisation de la concurrence entre les modèles économiques de production de logiciel.

NOUS avons montré que, même si l'organisation Libre réussit à conquérir une part de marché importante dans les solutions techniques, cela ne voulait pas dire qu'elle puisse sortir de cette niche pour diffuser ses produits parmi les utilisateurs naïfs. C'est une seconde étape dans le processus de diffusion. Le chapitre précédent a montré que la première étape commençait seulement, la deuxième ne démarrera donc pas tout de suite. Ne pouvant procéder par le biais d'enquêtes, nous nous proposons, dans ce chapitre, d'essayer de modéliser les principales caractéristiques des organisations Libre et Propriétaire et de la concurrence entre ces deux organisations pour étudier les caractéristiques de cette seconde phase du processus de diffusion.

Après avoir étudié les enseignements qu'on peut retirer des modèles de concurrence technologique existants, sachant les caractéristiques de la concurrence dans le logiciel, nous utiliserons ces enseignements pour définir les caractéristiques de notre propre modèle (partie 1), qui s'inspirera du modèle formel de concurrence entre technologies que nous avons proposé dans Dalle & Jullien [2000] et développé dans Dalle & Jullien [2001]. Nous l'appliquerons ensuite à la concurrence entre deux systèmes d'exploitation, Linux et Windows 2000[1], qui est une des clefs de la diffusion du Libre. Ce modèle va ensuite nous permettre d'identifier, suivant les caractéristiques des logiciels, les stratégies les plus efficaces pour les imposer quand la concurrence se fait entre deux offres technologiques dont les caractéristiques sont déterminées ex-ante (partie 2). Dans la partie 3, nous nous placerons dans une perspective d'évolution dynamique des standards, où les entreprises peuvent améliorer leurs technologies au cours du processus de diffusion. Nous avons en effet montré au chapitre 3 que la façon dont étaient gérées l'innovation incrémentale et la dynamique de standardisation était une des principales différences entre l'organisation Libre et l'organisation Propriétaire. Nous étudierons finalement la façon dont fonctionnerait le système de concurrence technologique dans le cas où les logiciels libres se seraient imposés.

[1] Et ses successeurs. Dans la suite de l'exposé, nous appellerons ces produits W2000.

5.1 Construire un modèle de concurrence entre organisations de production.

5.1.1 Analyse.

Les conséquences des processus de «standardisation» ont été étudiées par la micro-économie classique (Katz & Shapiro [1985, 1986] et Farrell & Saloner [1986]) et par le courant «évolutionniste». Un autre courant d'analyse, proche du courant évolutionniste, s'intéresse, à la suite d'Arthur et de David, à la dynamique qui conduit à (et explique) ces phénomènes de standardisation. D'après Foray [1989], la principale différence entre ces deux approches réside dans l'identification de l'élément qui explique la supériorité de la technologie qui s'impose (et donc devient le standard). Dans le premier cas, il s'agit d'étudier les processus d'adoption (ou de diffusion) d'une technologie ou de technologies en concurrence, technologies dont on connaît les caractéristiques. Dans le second on défend l'idée que les caractéristiques et l'efficacité d'une technologie se construisent en même temps qu'elle se diffuse. Il résume cette hypothèse en disant que «the increased adoption of a technology paves the way for the improvement in its characteristics». La diffusion d'une technologie est alors «un processus dynamique, dont le moteur réside dans l'action d'adopter» (Foray [1989], p. 17).

Nous nous placerons plutôt dans le second cas, même si les travaux réalisés dans l'autre cadre vont nous servir à définir les paramètres à prendre en compte dans notre modèle et notamment les paramètres qui influent sur la décision d'adoption d'un agent. Nous étudierons, dans un premier temps, la façon dont le choix des autres acteurs influe sur le choix de l'adopteur (et donc sur le processus de diffusion global), dans un second temps sur les caractéristiques individuelles qui influent sur ce processus de diffusion et dans un troisième la façon dont on peut prendre en compte les stratégies que les entreprises peuvent mettre en oeuvre pour soutenir leur offre.

L'influence des autres adopteurs : influences globales et locales, influences directes et indirectes.

Les différents types de rendements croissants.

Nous avons distingué, au chapitre 3, quatre sources de rendements croissants d'adoption qui font dépendre le choix d'un utilisateur des choix des autres utilisateurs et donc qui influent sur la standardisation : les rendements croissants d'information, les externalités de réseau, les économies d'échelle en production et les interrelations technologiques (l'offre de technologies, de produits, compatibles avec le bien).

L'économie des réseaux (voir, par exemple, Dalle [1995], Steyer & Zimmermann [1996]) a montré qu'il convenait de distinguer les externalités directes «globales», quand la valorisation dépend effectivement du nombre d'individus qui consomment le bien (on pense, par exemple, à la pollution ou à l'encombrement d'une piscine ou d'un réseau de télécommunications, mais aussi, pour partie aux rendements croissants d'information), et les externalités directes «locales», lorsque la valorisation ne dépend que d'un sous groupe de la population avec lequel l'individu est en contact (c'est ce qui se passe dans les réseaux de télécommunications et notamment lorsqu'il s'agit d'un individu qui échange des fichiers ou des informations).

Les autres rendements croissants d'adoption (économie d'échelle, interrelation technologique, rendements croissants d'information), dépendent pour partie du nombre d'adopteurs[2] et pour partie de la stratégie des entreprises productrices du bien : elles peuvent «sponsoriser» l'offre de biens complémentaires, anticiper sur les effets d'échelle pour diminuer le prix du bien, faire de la publicité pour informer sur les qualités du bien et enfin investir pour développer ces qualités[3].

[2]En ce sens que «les consommateurs vont préférer un produit principal lorsque les fournisseurs de produits complémentaires sont nombreux sur le marché» (Le Nagard [1997], p. 107), car cela va engendrer une plus grande concurrence, un plus grand nombre de produits complémentaires, etc., et que «les fournisseurs de biens complémentaires [sont] d'autant plus nombreux que le volume du marché du bien principal est important» (ibid)

Church & Gandall [1992, 1993] ont fait remarquer que, pour relâcher la concurrence, les fournisseurs de biens complémentaires pouvaient avoir intérêt à proposer des produits compatibles avec le bien le moins diffusé et donc, celui où les produits complémentaires étaient moins nombreux. Mais les enquêtes montrent que les éditeurs considèrent bien la base installée (actuelle et anticipée) comme le premier critère stratégique (Le Nagard [1997], p. 111).

[3]En ce qui concerne les effets d'annonce qui accompagnent souvent la mise sur le marché d'un produit standard, on consultera, outre Farell & Saloner [1986], Tellier [1996] et Bloch & Mancea [1996], par exemple.

Mais finalement, il semble qu'on puisse classer ces rendements croissants d'adoption suivant deux critères : la taille de la population concernée (grossièrement, l'ensemble de la population ou une petite fraction de celle-ci) et le fait que ces effets soient directement perçus par l'utilisateur ou qu'ils soient médiatisé par une entreprise (c'est à dire que les utilisateurs perçoivent entièrement les bénéfices d'une adoption ou seulement une partie). Reste à déterminer leur importance.

Discussion sur l'importance de chaque type d'influence dans le processus de diffusion.

Katz & Shapiro [1994], Besen & Farrell [1994] et Liebowitz & Margolis [1994] ont étudié ces différents effets. Il ressort de leurs travaux que si l'on constate bien que le nombre d'adopteurs joue un rôle, il est plus difficile d'identifier des diffusions où ce rôle est dû à des externalités. Les rendements croissants d'adoption qui ont conduit par le passé à des standards ont plus souvent été des économies d'échelle, des effets d'interrelation technologiques ou des effets d'apprentissages par l'usage que des interactions directes entre adopteurs. Nous avons fait la même analyse dans le cas particulier de l'industrie informatique, constatant que jusqu'à une époque récente, la coordination entre les adopteurs était médiatisée par le produit, contrôlée par les producteurs qui géraient la redistribution des rendements croissants générés.

Liebowitz & Margolis [1994] ont donné l'explication de ce phénomène : pour que ces effets aient une importance, il faut qu'existe une structure d'interaction, d'échange d'informations, qui permette aux adopteurs de se coordonner sur les produits qu'ils souhaitent voir développés (avant la diffusion) et d'échanger ces produits (ou des informations créées par ces produits pendant la diffusion). Avec l'arrivée d'Internet, ces échanges sont possibles, l'importance des externalités en tant que facteur explicatif des trajectoires de standardisation augmente. Les utilisateurs peuvent se regrouper par affinité (Bakos & Brynjolfsson [2000]), créer des clubs et développer dans ces clubs des solutions adaptées à leurs besoins. Ensuite, le même média informe de l'existence du produit, ce qui assure sa diffusion hors du club, de la sphère des primo-adopteurs.

Et il s'agit essentiellement d'externalités locales. Car, comme l'ont aussi fait remarquer

Liebowitz & Margolis [1994], reprenant les travaux de Buchanan & Stubblebine [1962], même dans le cas où ces interactions existent, les utilisateurs n'interagissent qu'avec une partie de la population : les agents qui n'appartiennent pas à l'environnement proche d'un utilisateur n'ont qu'un effet direct «infra-marginal» sur la décision d'adopter, c'est-à-dire que l'utilité marginale de leurs externalités est nulle pour l'agent concerné.

Si nous nous plaçons alors dans la lignée des travaux d'Arthur, qui a en partie initié la réflexion et la modélisation sur les effets de standardisation en présence de rendements croissants d'adoption, nous nous appuierons surtout sur les auteurs qui, en même temps qu'ils ont souligné l'intérêt des propositions d'Arthur, ont fait remarqué leurs faiblesses, et notamment le fait qu'elles négligent l'importance des structures d'interaction locales[4]. Mais à partir du moment où l'on considère que les interactions locales jouent un rôle important dans le processus de diffusion, chaque individu a lui aussi son importance, et il faut tenir compte de ses caractéristiques propres.

L'importance de prendre en compte l'hétérogénéité des préférences.

Bakos & Brynjolfsson [1998], Kirman [1992b] ont montré que les hétérogénéités des agents ne pouvaient être négligées si l'on voulait comprendre les processus de construction des standards.

Même si Arthur [1989], Katz & Shapiro [1994], Besen & Farrell [1994] et Liebowitz & Margolis [1994] tiennent compte de cette hétérogénéité, en supposant par exemple que les agents ont une préférence pour l'une ou l'autre des technologies, ces auteurs, parce qu'il s'intéressent essentiellement à des effets globaux, donc aux demandes agrégées, sous-estiment les effets que peut avoir une hétérogénéité forte sur le processus de diffusion. C'est, dans le cas de l'apparition du Libre et plus généralement du logiciel, un paramètre fondamental et très variable : les règles de décision sont, en effet, très différentes suivant les adopteurs : la qualité, les performances (en terme de fréquence d'erreurs), la disponibilité et la variété des logiciels compatibles, la facilité d'installation et d'utilisation, les coûts directs et indirects (achat, formation, maintenance, correctifs) seront des paramètres plus ou moins importants dans leur évaluation de l'utilité qu'apporte une solution logicielle. Ces

[4] Voir, par exemple, Dalle [1995, 1997, 1998ab], Durlauf [1993], David [1994, 1998], Kirman [1993, 1998].

paramètres interviendront de façon complètement différente dans l'utilité d'un «hacker» et dans celle d'un utilisateur sans compétence en informatique, ou dans celle d'un utilisateur individuel et dans celle d'une grande entreprise, et il faudra en tenir compte dans le modèle.

L'autre point important dans la standardisation concerne la façon dont les producteurs redistribuent les bénéfices des rendements croissants qu'ils perçoivent, comme les économies d'échelle. Il y a, dans la littérature, différentes façons de les prendre en compte.

Les stratégies des producteurs.

Chez Arthur, la stratégie des producteurs concernant la redistribution des rendements croissants d'adoption n'est pas prise en compte. Il suppose que ces rendements croissants sont automatiquement et d'une certaine façon «magiquement» réinvestis dans des améliorations technologiques, c'est-à-dire redistribués aux consommateurs. Le modèle d'Arthur est essentiellement un modèle de demande, où les stratégies des entreprises sont fixées et où on étudie la façon dont la demande sélectionne ces stratégies.

Ce n'est évidemment pas toujours le cas et les résultats d'Arthur apparaissent, de ce point de vue, trop spécifiques. C'est pourquoi nous nous inspirerons du modèle de Katz & Shapiro [1986], modèle d'offre, pour étudier différents positionnement stratégiques des entreprises. Dans ce modèle, en effet, les entreprises peuvent subventionner les produits, ce qui revient à pouvoir choisir le niveau de redistribution des externalités.

Finalement, si notre modèle reste proche de celui d'Arthur dans son objet d'étude (il s'agit de s'intéresser au processus dynamique de construction du standard), il nous semble indispensable d'intégrer des paramètres qui ont été mis en avant par d'autres auteurs, et notamment Katz & Shapiro [1985, 1986] sur les stratégies des entreprises ou Dalle [1995, 1997, 1998ab], Durlauf [1993], David [1994, 1998], Kirman [1993, 1998] sur l'importance des interactions locales.

5.1.2 Présentation du modèle.

Le modèle que nous utilisons est un modèle de concurrence entre technologies : des offreurs de technologies vont s'adresser à une même population d'utilisateurs, avec des

stratégies d'offre différentes, pouvant évoluer au cours du temps. Nous étudions le succès de ces stratégies sachant les caractéristiques de la demande et l'évolution de leur part de marché au cours du processus d'adoption. Ce modèle est donc défini par la donnée des stratégies de production et de commercialisation des producteurs de technologies (leur positionnement), des caractéristiques de la population et de la structure d'interaction qui relie les membres de cette population, et enfin de la façon dont nous modélisons la dynamique d'adoption.

La stratégie des producteurs.

Ces producteurs peuvent choisir deux choses dans une concurrence entre technologies : le positionnement de leur technologie par rapport aux besoins exprimés et la façon dont ils vont redistribuer les rendements croissants d'adoption qu'ils contrôlent. Ils pourront d'abord influencer l'utilité propre de chaque adopteur concernant les technologies : une forte baisse de prix, si elle n'influence pas de la même manière tous les consommateurs, a un impact sur la valorisation que chacun fait de la technologie. Ils pourront aussi changer de stratégie concernant les rendements croissants d'adoption. Les producteurs ont, par exemple, le choix entre les intégrer dans une distribution future en la faisant payer ou répondre directement à chaque demande en en faisant profiter l'ensemble des consommateurs. Dans le premier cas l'externalité marginale qu'un nouvel adopteur distribue aux autres adopteurs est quasiment nulle, elle est accaparée par le producteur, dans l'autre cas elle est forte. On trouve ces deux paramètres dans les modèles de Katz et Shapiro [1985, 1986]. Dans le modèle de 1985, ils étudient la stratégie optimale, pour un entrant, entre être compatible avec le standard existant (stratégie d'imitation) ou se différencier. Dans celui de 1986 ils s'intéressent à la stratégie de producteurs qui peuvent sponsoriser ou non une technologie, c'est à dire (dans leur modèle) verser une partie des économies d'échelle en production aux primo-adopteurs, en anticipant ces économies d'échelle.

Ces deux choix stratégiques n'interviennent pas tout à fait au même moment du processus de diffusion : même si l'on peut adapter son produit, développer ses fonctionnalités, le positionnement technologique originel est difficile à remettre en cause de façon radicale. Si une part des rendements croissants d'adoption ne dépend pas de la stratégie de l'entre-

prise, elle peut à tout moment décider de sa stratégie de redistribution pour ceux qu'elle maîtrise (subvention de son produit au début ou au cours du processus pour anticiper les économies d'échelle, subvention des offres complémentaires dans le cas de la vente d'un bien lié, etc.) même si, là aussi, le poids des routines est important. On ne passe pas sans coûts importants du métier d'éditeur de logiciels au métier de fournisseur de services basés sur des logiciels quand on n'a jamais proposé de services (nous avons vu au chapitre 3 que la difficulté de la transition dépendait du secteur sur lequel opéraient les producteurs de logiciel).

C'est pourquoi la plupart des modèles font l'hypothèse de la stabilité des caractéristiques technologiques et ne font évoluer, dans le temps, que les stratégies des entreprises. Nous avons vu qu'un des avantages de l'organisation Libre était de favoriser les évolutions technologiques, grâce à un système de normalisation dynamique, à ce que Foray [1990] a pu appeler «un processus dynamique de production des standards». Ce qui nous amène à préciser la façon dont les adopteurs valorisent les technologies et dont cette valorisation évolue au cours du temps.

Les adopteurs.

Les caractéristiques des adopteurs.

Pour rendre compte de leur hétérogénéité, nous avons décidé d'adopter un point de vue statistique, en prenant en compte la propension statistique qu'un adopteur, choisi au hasard dans une population donnée, a à préférer l'un ou l'autre des deux standards en concurrence[5]. Cette propension dépendra de la valorisation que chaque adopteur fera des technologies, valorisation qui elle-même dépendra de l'utilité propre, intrinsèque, qu'il retire de chaque technologie et des influences qu'il subira de son environnement (entreprises et autres adopteurs).

Nous aurons donc une distribution S des préférences personnelles, de l'utilité idiosyncrasique de chaque adopteur (ou plutôt de la différence des utilités des deux technologies), variable aléatoire que nous noterons «ΔU». Elle dépendra des positionnements techno-

[5] A priori, rien n'empêche d'avoir plus de deux technologies. Cependant, on se heurte alors à des problèmes de classement des préférences (paradoxe de Condorcet), qui rendent complexe la représentation formelle de cette distribution des préférences, ce qui compliquerait l'étude mathématique, au détriment de l'étude des caractéristiques économiques de ces processus de concurrence.

logiques initiaux et des caractéristiques de la population (et notamment de son niveau de connaissance technologique). ΔU peut potentiellement varier de $-\infty$ (préférence absolue pour l'une des technologies, appelée technologie 2) à $+\infty$ (préférence absolue pour l'autre technologie, la technologie 1). Une distribution normale des préférences centrée en 0 signifiera que, globalement, la population des adopteurs est indifférente entre les deux technologies, alors qu'une distribution bimodale (avec un mode positif et un mode négatif) sera la marque d'une segmentation de la population en deux sous-groupes intéressés par des offres technologiques différentes.

On peut exprimer la probabilité que la différence d'utilité d'un agent tiré au hasard dans la population des adopteurs soit inférieure à un certain E :

$$P(\Delta U \leq E) = \int_{-\infty}^{E} dg$$

avec g la fonction de densité de la distribution G. Cette fonction peut varier au cours du temps et de l'évolution des technologies c'est-à-dire essentiellement des caractéristiques d'utilisations auxquelles répondent ces technologies. La valeur de E dépendra de l'environnement de l'agent, c'est-à-dire de la stratégie commerciale des producteurs et des actions des autres adopteurs.

Les éléments qui interviennent dans la valeur de E.

Les effets globaux dépendent, en première approximation, de la part de marché de chaque technologie, puisque, globalement, plus une technologies est adoptée, plus les perspectives pour les producteurs de technologies complémentaires sont vastes et mieux la technologie est connue (nous suivons là les hypothèses d'Arthur et de Katz & Shapiro). Les effets locaux dépendent de l'influence des agents avec qui on échange directement des données (fichier informatique), des informations (existence d'un logiciel) ou des connaissances (aide pour installer un logiciel, conseil d'achat, etc.). Il est coutume d'appeler l'ensemble de ces agents le «voisinage» de l'adopteur (David [1988]).

Pour modéliser ce voisinage il faut choisir, pour chaque agent, un sous-ensemble d'autres adopteurs potentiels et l'influence que ces adopteurs auront pour l'agent. Nous le représenterons de la façon suivante : les individus sont répartis sur la surface d'un tore

à deux dimensions, la «structure d'interaction», et chaque adopteur possède un certain nombre de voisins locaux, typiquement 4 ou 8, qui influent de la même manière sur sa décision, donc qui sont à prendre en compte dans la fonction E.[6]. Si l'on appelle «A» l'individu tiré au hasard dans une population d'individus notés «X» et «V» ses voisins, la représentation graphique des voisinages est la suivante :

X X X X X X X X X	X X X X X X X X X
X X X X X X X X X	X X X X X X X X X
X X X X *V* X X X X	X X X *V* *V* *V* X X X
X X X *V* **A** *V* X X X	X X X *V* **A** *V* X X X
X X X X *V* X X X X	X X X *V* *V* *V* X X X
X X X X X X X X X	X X X X X X X X X
Structure d'interaction locale à quatre voisins, dite «de von Neumann».	Structure d'interaction locale à huit voisins, dite «de Moore».

Suivant les caractéristiques d'utilisation, c'est-à-dire suivant le bien échangé, les influences respectives du local et du global seront plus ou moins importantes et les agents y seront plus ou moins sensibles. Ces deux paramètres sont intrinsèques aux caractéristiques d'utilisation étudiées, mais suivant les stratégies des producteurs de technologies, suivant la part des effets de réseau qui est accaparée par le producteur, ces effets seront plus ou moins fortement ressentis par les adopteurs.

Nous donnons à la fonction E, qui évalue le poids de l'environnement, les caractéristiques suivantes.

Caractérisation du seuil d'adoption.

Nous proposons de noter a le paramètre qui mesure l'importance des effets de réseaux et b celui qui mesure la répartition des effets de réseau entre influence locale et influence locale. Nous prendrons comme convention que $0 \leq b \leq 1$, 1 s'il n'y a que des effets globaux,

[6] Certains travaux ont montré que la structure du réseau d'interaction était aussi un paramètre important de la diffusion. Voir par exemple Plouraboué, Steyer & Zimmermann [1998], Deroïan [1999], Deroïan, Steyer & Zimmermann [1999], Watts [1999]. Pour un résumé, on pourra consulter Cohendet & Schenk [1999] et Vicente [2000].
Une des pistes d'extension de ce modèle consiste bien sûr à améliorer la structure des interactions locale.

0 s'il n'y a que des effets locaux.

On peut alors écrire que :

$$E = E\left[a,\, b * I(X_1,\, X_2, t),\, (1-b) * i(x_1,\, x_2, t)\right],$$

$$avec,\, a\, priori,\, -\infty \leq E \leq +\infty.$$

X_1(resp. X_2) est la part de marché globale de la technologie 1 (resp. 2), x_1 (resp. x_2) est la part de marché locale de la technologie 1 (resp. 2). I (resp. i) est la fonction qui agrège ces influences globales (resp. locales). I et i sont fonctions du temps puisqu'il est possible qu'au cours de la diffusion les entreprises changent leurs stratégies de redistribution des externalités.

Nous voulions, dans ce modèle, pouvoir comparer l'impact des influences globales et locales. C'est pourquoi nous avons choisi de prendre en compte des paramètres qui rendent comparable les valeurs des deux influences, les parts de marché (de la même façon que dans les modèles d'Arthur ou dans celui de Durlauf [1993]), et de faire porter par un paramètre, b, la comparaison de l'importance respective des influences locales et globales. Enfin, nous postulons que la fonction E est symétrique par rapport à ses deuxièmes et troisièmes variables (sinon, cela voudrait dire que les influences globales interviennent différemment sur le comportement des utilisateurs et on ne pourrait pas facilement les comparer). Nous suivons, là encore, la modélisation de Durlauf [1993].
On a donc :

$$E\left[a,\, (1-b) * i(x_1,\, x_2, t),\, b * I(X_1,\, X_2, t)\right] = E\left[a,\, b * I(X_1,\, X_2, t),\, (1-b) * i(x_1,\, x_2, t)\right]$$

Comme il s'agit de parts de marché, on peut aussi écrire que :

$$X_1 = 1 - X_2\,;\, x_1 = 1 - x_2$$

Ce qui donne, pour E :

$$E = E\left[a,\, b * I(X_1, t),\, (1-b) * i(x_1, t)\right],$$

et

$$\frac{\partial E}{\partial X_2} \geq 0\,;\, \frac{\partial E}{\partial x_2} \geq 0$$

Nous prenons E continue en X_1 et x_1 ; nous nous limiterons effectivement, par souci de clarté dans l'exposé, aux fonctions continues (et même C^∞), même si nous n'utiliserons que quelques valeurs de ces fonctions (puisque la taille de la population est finie, que le nombre de voisins de chaque agent est, lui aussi, fini et que l'on tirera les agents suivant un temps discret).

Plus la part de marché de la technologie 2 sera importante, plus la valeur d'utilité idiosyncrasique pour laquelle on sera indifférent entre choisir celle-ci ou la technologie 1 sera élevée (un individu qui a une utilité idiosyncrasique de ' $+\infty$ choisira toujours la technologie 1). Comme nous l'avons mentionné plus haut, cette proportion dépend et du niveau de diffusion local et du niveau de diffusion global. Eux-mêmes dépendent des rendements croissants d'adoption et du fait que ces rendements ont ou n'ont pas été redistribués aux utilisateurs par les producteurs.

Lorsque nous avons analysé la production de logiciel libre, nous avons constaté que le gain marginal d'un contribution augmentait puis diminuait avec le nombre de ceux qui contribuent. De la même façon, il est admis que les effets marginaux d'une adoption augmentent puis diminuent lorsque la part de marché d'une technologie croît (c'est la courbe de diffusion en «S» des modèles de marketing). C'est la remarque de Liebowitz et Margolis [1994], remarque que nous prenons en compte en considérant que :

$$\begin{aligned}
\frac{\partial^2 E}{\partial X_2^2}\left[a, b * I(0,t),\, (1-b) * i(0,t)\right] &\geq 0 \\
\frac{\partial^2 E}{\partial x_2^2}\left[a, b * I(0,t),\, (1-b) * i(0,t)\right] &\geq 0
\end{aligned}$$

mais que :

$$\forall\, a,\, b,\, t\,, x_1, \quad \exists\, X\, tq \forall\, X_1 \geq X, \quad \frac{\partial^2 E}{\partial X_2^2}\left[a, b * I(X_1, t),\, (1-b) * i(x_1, t)\right] \leq 0$$

$$\forall\, a,\, b,\, t\,, x_1, \quad \exists\, x\, tq \forall\, x_1 \geq x, \quad \frac{\partial^2 E}{\partial x_2^2}\left[a, b * I(X_1, t),\, (1-b) * i(x_1, t)\right] \leq 0$$

La fonction E est indépendante des utilisateurs car elle mesure l'impact des facteurs externes sur le choix des technologies. Certes, l'importance des effets externes peut varier suivant les utilisateurs et dépendre de leurs caractéristiques intrinsèques : certains, parce qu'ils échangent souvent des fichiers, seront plus sensibles à ces facteurs que d'autres. Mais ils sont pris en compte dans la distribution des préférences : un individu qui valorise les influences externes sera relativement indifférent aux technologies, sa différence d'utilité sera proche de zéro ; s'il ne les valorise pas, il aura tendance à choisir les technologies uniquement en fonction de ses préférences idiosyncrasiques, ce qui veut dire que sa différence d'utilité sera élevé en valeur absolue.

Nous ne prenons pas non plus en compte explicitement les anticipations des agents alors que c'est un des caractères spécifiques à la concurrence entre des offres soumises aux rendements croissants d'adoption (Katz & Shapiro [1986], Foray [1989]). Notre point de vue est le suivant : les agents font des choix en univers incertain, ils sont donc caractérisés par une certaine myopie. Sachant cela, ils font des anticipations, des hypothèses sur le futur, hypothèses qui dépendent de leur information publique et privée (l'influence des autres agents), mais aussi de leur capacité idiosyncrasique à traiter cette information, c'est-à-dire de leurs connaissances. Tout ceci construit le seuil d'adoption, au même titre que la valorisation des caractéristiques techniques des produits. Les anticipations des agents sont donc intégrées dans ΔU. Si elles ne sont pas explicitées, c'est dû à la myopie du modélisateur, qui ne connaît pas les anticipations individuelles et qui les modélise, au même titre que les autres caractéristiques de la demande, de façon statistique.

Par contre, la fonction E est dépendante de la position «géographique» des utilisateurs : en tirant un utilisateur, on va tirer un voisinage d'adopteurs qui se répartit entre l'une et l'autre des technologies. La définition de la répartition géographique des utilisateurs

est donc un point important de la construction du modèle. Nous y reviendrons dans la section suivante, où nous proposons une mise en œuvre du modèle, ce qui nous oblige, bien entendu, à discuter de l'organisation géographique retenue.

Enfin, la fonction de seuil évolue à chaque adoption si les effets de réseau sont redistribués immédiatement, ou par période si les entreprises ne les redistribuent que périodiquement. On touche là à l'étude de la dynamique d'adoption, c'est-à-dire à la façon dont les agents révisent leur choix et sur la prise en compte de cette révision par les autres agents.

La dynamique de diffusion.

Nous simulons des trajectoires technologiques en choisissant à chaque intervalle de temps un adopteur au hasard, c'est-à-dire une position sur la structure d'interaction, ce qui nous permet de calculer la valeur du seuil d'adoption.

Si le tirage de la variable aléatoire ΔU donne une valeur inférieure à ce seuil d'adoption, l'agent choisira la technologie 1, si elle est supérieure l'agent choisira la technologie 2. Ce choix aura un impact local puisqu'il peut changer la part de marché locale de chaque technologie pour ses voisins et, bien sûr, un impact global puisqu'il va changer la part de marché totale de chaque technologie.

Le paramètre agrégé qui nous permet de comparer les processus de diffusion est la vitesse de diffusion : si une technologie ne s'est pas imposée au bout d'un certain temps, même si on peut prouver que mathématiquement la technologie peut se diffuser, elle le fera hors d'un temps économiquement pertinent et on peut alors conclure à une cohabitation technologique durable (voir Dalle [1998b] pour une discussion plus approfondie sur la signification du temps de diffusion).

Nous allons appliquer ce modèle dans le cas de la diffusion des logiciels libres.

5.2 Mise en œuvre du modèle.

La mise en œuvre du modèle nécessite de spécifier la fonction de seuil et l'organisation «géographique» des utilisateurs, ce qui veut dire spécifier les caractéristiques des technolo-

gies et des modèles économiques étudiés. Nous allons le faire dans le cas de la concurrence entre Linux et Windows sur les systèmes d'exploitation, même si cette spécification sera suffisamment souple pour être adaptable aux autres logiciels libres. Nous en tirerons ensuite des enseignements qui concerneront l'ensemble de ces logiciels.

5.2.1 Une application dans le cas de la concurrence Windows vs Linux.

Dans la suite du chapitre, la technologie 1 représentera la technologie déjà installée (ici Windows, noté W2000) et la technologie 2 représentera le nouvel entrant (ici Linux).

Spécification de la fonction de seuil et de la répartition géographique des adopteurs.

Pour pouvoir réaliser les simulations, nous avons besoin de spécifier le voisinage local, la taille de la population et enfin la distribution des préférences intrinsèques de ces utilisateurs. Il faut aussi choisir une fonction qui aura les bonnes propriétés pour représenter la fonction E, qui modélise l'influence externe. Nous allons commencer par ce point.

La fonction E.

Nous la définissons de la façon suivante :

$$E = \tanh\left[a * \left[(1-b) * (x_2 - \alpha * x_1) + b * (X_2 - \beta * X_1)\right]\right]$$

On vérifiera que cette fonction tanh permet à E de vérifier toutes les spécifications du paragraphe précédent. Elle varie entre -1 et 1, ce qui signifie que les agents qui ont une utilité personnelle (ΔU) supérieure en valeur absolue à 1 ne changeront jamais de technologie. Ces agents sont les prosélytes de chaque technologie, et quelleque soit l'influence extérieure, ils resteront «accrochés» à leur technologie. Dans le cas de la concurrence entre Windows et Linux, on peut penser que les premiers adopteurs de Linux étaient de ce type : même s'ils avaient pu obtenir Windows gratuitement, ils n'auraient pas été satisfaits par le produit parce que ses caractéristiques techniques étaient trop éloignées de leurs besoins.

Nous avons introduit deux paramètres supplémentaires, α et β, qui représentent les

différences de stratégies dans la redistribution des externalités et donc, plus généralement, les modèles économiques en concurrence : α estime statistiquement l'influence relative de la part de marché locale de W2000, comparée à celle de Linux; β compare statistiquement celles des parts de marché globales. Du côté des produits de Microsoft, α et β sont influencés par le choix de la sortie régulière et de la vente de versions «améliorées» mais imparfaitement compatibles et par une stratégie monopolistique de fixation des prix, bref par un comportement de monopole en ce qui concerne les prix et la qualité des produits. On peut considérer qu'une part importante des externalités n'est pas redistribuée ou réinvestie mais appropriée. Dans le cas de Linux, α et β sont fortement influencés par la créativité de la population des adopteurs et des développeurs. C'est surtout vrai pour β, car il s'agit d'un effet global : plus grande est la population d'utilisateurs actifs de Linux capables d'utiliser leurs talents pour corriger des bogues et proposer des améliorations, plus petit sera β, l'adoption de Linux engendrant plus d'externalités. Les entreprises dédiées au logiciel libre, comme RedHat ou VA Linux, permettent aussi, nous l'avons vu, d'abaisser β. Le prosélytisme pour Linux est essentiellement local et jouera sur la valeur de α. Pour présenter les choses différemment, la demande statistique est plus ou moins élastique en fonction des caractéristiques des technologies et des modèles économiques qui leurs sont associés. L'élasticité pourrait, au demeurant, se mesurer à partir des dérivées partielles de α et β par rapport à de nombreux paramètres, à commencer par le prix, mais sans devoir s'y limiter. La stratégie de versions de Microsoft, la créativité des utilisateurs de Linux, les comportements anti-Microsoft ou le développement d'entreprises spécialisées autour de Linux sont des facteurs tout aussi pertinents dans le cas présent.

La répartition des adopteurs.

Nous avons choisi de répartir les adopteurs sur un tore en deux dimensions de 30x30 (soient 900 adopteurs potentiels) et un voisinage de 4 voisins pour chaque adopteur. Comme pour le voisinage, si cette modélisation a le mérite de la simplicité, elle n'a pas celui du réalisme (la spécification du voisinage et de la répartition géographique des utilisateurs vont souvent de pair). Nous avons préféré, ici, nous concentrer sur la modélisation des différentes stratégies des producteurs, plus centrale dans notre propos, même si, une fois de plus, l'hétérogénéité des acteurs (que nous prenons en compte) et l'hétérogénéité

de leurs interactions (que nous prenons en compte de façon plus frustre) sont des éléments clefs du processus de diffusion et de sélection des stratégies victorieuses. Ce sont des pistes qu'il faudra explorer dans une version ultérieure du modèle[7].

Le déroulement des simulations et les résultats attendus.

Nous commençons les simulations avec une population ayant adopté W2000 de façon uniforme, ce qui représente une base installée très importante, et nous effectuons une suite de tirages aléatoires d'individus.
Nous mesurons le «temps» qu'il faut à Linux pour atteindre 70 % de part de marché (sachant que nous stoppons le processus au bout de 50000 tirages si ces 70 % n'ont pas été atteints).

Comme la redistribution des externalités globales et locales est supposée plus forte pour Linux que pour W2000, nous avons $\alpha \leq 1$ et $\beta \leq 1$, ce qui veut dire que l'adopteur potentiel adoptera plus souvent Linux que W2000 quand le nombre des adopteurs précédents de Linux et de W2000 sera le même, que ce soit dans son voisinage ou dans la population entière. Ce n'est pas vraiment une hypothèse restrictive : nous étudions une concurrence entre deux modèles économiques, entre deux systèmes de production, l'un déjà en place, l'autre émergeant. Si le deuxième système n'est pas plus efficace que le premier (au moins à court terme), il n'y a aucune chance qu'il s'impose.

Nous répéterons le processus complet pour toutes les valeurs de α et de β comprises entre 0 et 1 avec un pas de 0,1. Nous étudions, sachant la distribution de la population, quel «niveau» d'efficacité supplémentaire est nécessaire pour que Linux diffuse.

Résultats dans le cas d'une distribution uniforme.

La figure 5.1 représente les résultats dans le cas où la variable aléatoire ΔU est distribuée uniformément entre -1 et 1.
Comme nous étudions un cas où les effets de réseau sont importants, a est élevé (égal à 5 et à 2,5) et comme les interrelations technologiques sont sans doute aussi importantes que les

[7]Les pistes de modélisation ouvertes par Plouraboué, Steyer & Zimmermann [1998], Deroïan [1999], Deroïan, Steyer & Zimmermann [1999], Watts [1999], seront alors à étudier.

effets de club, on accordera autant d'importance à l'influence locale qu'à l'influence globale ($b = \frac{1}{2}$). Ces choix semblent appropriés pour une technologie sensible aux phénomènes de standardisation comme les systèmes d'exploitation.

On constate l'existence d'une transition de phase, en ce sens que α et β sont des paramètres d'état associés à une forte discontinuité dans les régimes de diffusion. Au dessus de valeurs critiques en α et β, la diffusion est infiniment longue (elle ne se produira pas dans le temps économique), alors qu'en dessous de ces valeurs, la diffusion est presque sure et même rapide.
Par conséquent, on peut dire que la situation de lock-in en faveur de W2000 n'est pas irrémédiable si Linux peut bénéficier (suffisamment) plus que W2000 des externalités globales et locales associées à sa diffusion. Dans ce cas, la diffusion sera même relativement rapide.

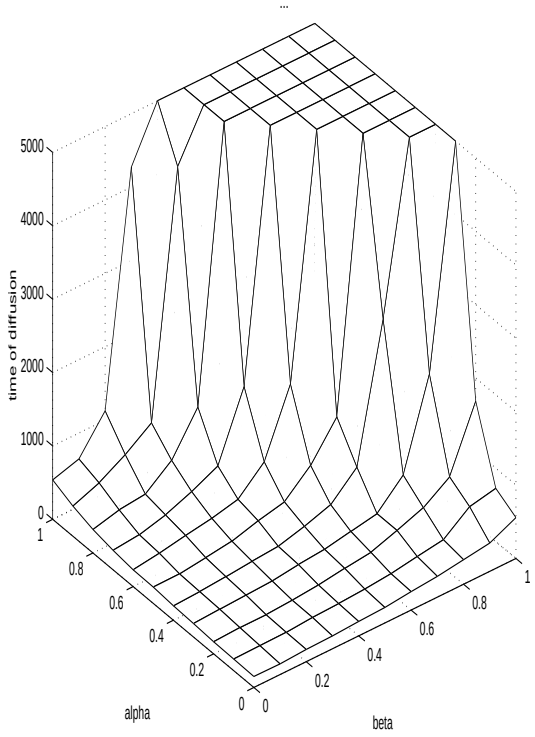
Nous nous sommes cependant placés dans une situation très favorable au logiciel libre, puisque les influence locales sont importantes, ce qui permet de diffuser le logiciel en construisant des niches d'adopteurs autour d'un zélote de ce logiciel. Mais surtout, la distribution uniforme signifie qu'une grande partie de la population a une préférence très marquée pour Linux, hypothèse hautement improbable au regard de la taille de la population des utilisateurs naïfs qui sont indifférents aux caractéristiques technologiques propres à ce système. Nous allons étudier, dans la suite, la diffusion de logiciel libre dans une population d'utilisateurs aux caractéristiques d'utilité différentes, sans doute plus proches de la réalité. Avant cela, nous voudrions dire deux mots sur la robustesse du modèle et de ses résultats.

La robustesse du modèle.

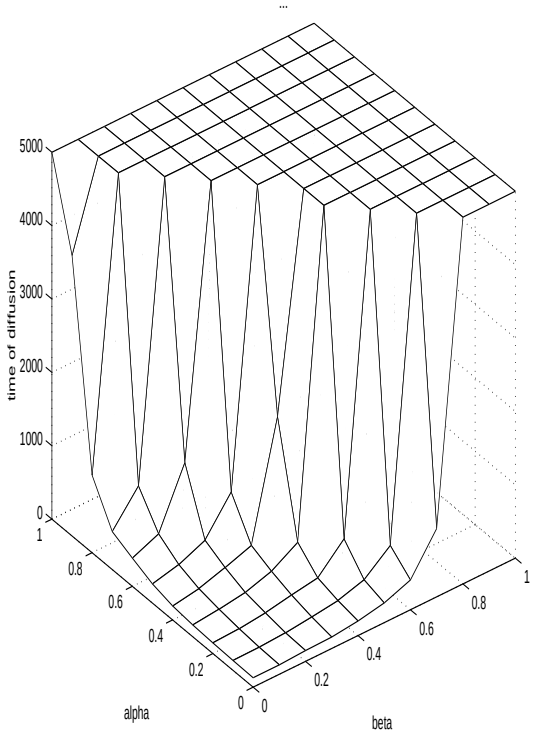
Nous n'avons pas étudié chaque valeur du couple (α, β) sur un grand nombre de diffusion. Lorsque nous étudions une série de diffusion en fonction des paramètres α et β, nous nous sommes même limité à une diffusion. Si répéter les diffusions permet d'éliminer des effets dûs à un tirage particulier des préférences, cela augmente aussi beaucoup le temps de simulation. Le risque d'un tirage particulier étant surtout important pour des valeurs de α et de β proches de la transition de phase, nous avons répété les diffusions uniquement

Figure 5.1 — Temps de diffusion de Linux, distribution uniforme ($0 \leq \alpha \leq 1$ et $0 \leq \beta \leq 1$).

$b = \frac{1}{2}$

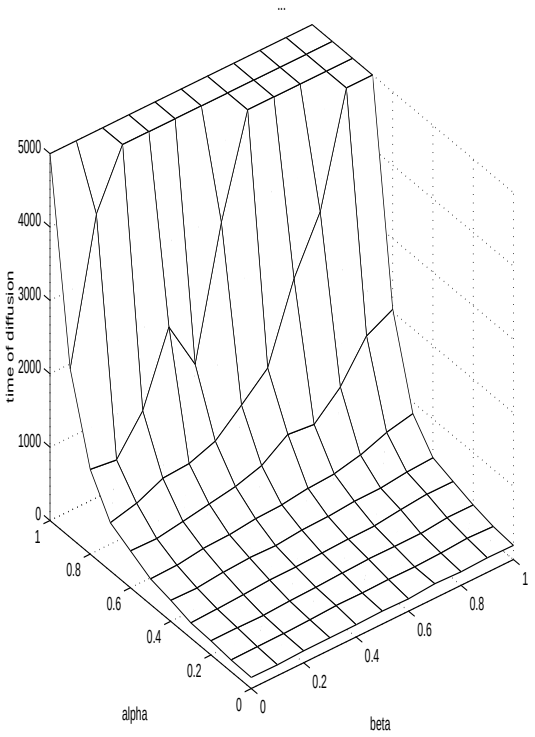


$a = 2, 5.$

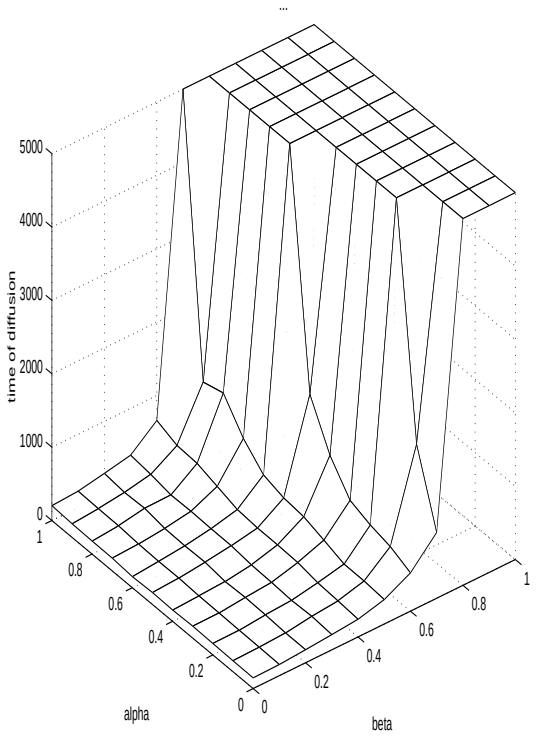


$a = 5.$

$a = 2, 5$



$b = 0, 2.$



$b = 0, 8.$

pour ces valeurs (typiquement sur une vingtaine de diffusion). Nous avons constaté une chose : il s'agit bien d'une transition de phase, au sens où lorsqu'il y a diffusion, il y a diffusion presque tout le temps, même si le moment où commence la diffusion varie suivant les tirages (voir figure 5.2).

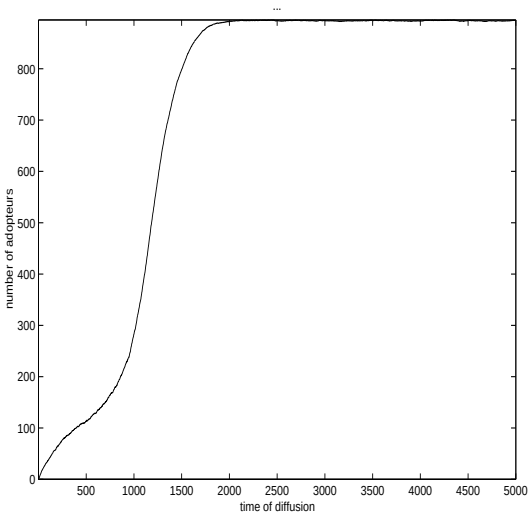
Cette étude nous a aussi permis de vérifier que la diffusion se faisait bien autour de niches d'adopteurs, ce qui souligne l'importance du caractère local du processus de diffusion technologique. Une fois que les niches ont atteint un certain seuil, la diffusion est très rapide et prend typiquement l'allure d'une courbe en «S».

Pour mettre en lumière ces deux phénomènes (apparition de niches et diffusion presque tout le temps, mais avec un temps qui peut varier un peu), nous avons étudié l'évolution de deux variables de la diffusion : ce que nous avons appelé l'«indice de cohérence» de la diffusion, qui mesure le nombre moyen de voisins d'un agent qui ont adopté la même technologie que lui (divisé par le nombre de voisins) et l'écart type du nombre d'adopteurs de Linux dans les 20 tirages. Si l'«indice de cohérence» d'une technologie est plus important que sa part de marché, c'est que les adopteurs de cette technologie sont regroupés, qu'ils forment des niches. On peut constater sur les graphiques de la figure 5.2 que le regroupement en niches précède la diffusion de la technologie. C'est une propriété que nous avons constatée, avec plus ou moins de force, dans toutes les diffusions. Au moment de la diffusion, il y a une augmentation forte de l'écart-type, ce qui signifie que, suivant les tirages, le processus de diffusion ne commence pas exactement au même moment. Mais au bout d'un moment, cet écart-type devient inférieur à 1 : pour tous les tirages, il y a eu diffusion (si, pour une simulation, il n'y avait pas diffusion, l'écart-type serait plus proche de 200).

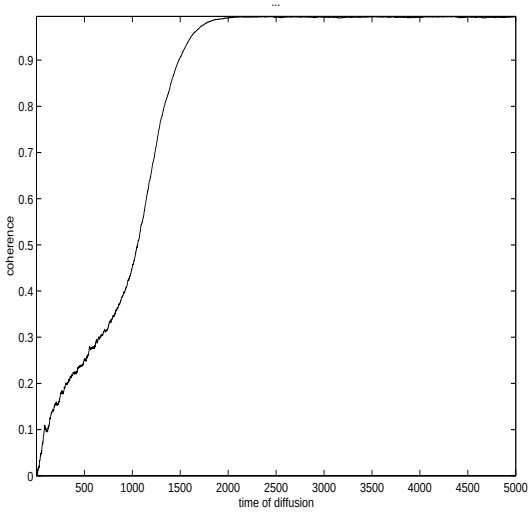
Ces résultats permettent de conclure que les risques de résultats «parasites» lorsqu'on étudie les chances que Linux diffuse en fonction des paramètre α et β sont faibles.

Figure 5.2 — Courbes de diffusions dans une population dont les préférences sont distribuées uniformément. Moyenne de 20 tirages, $\alpha = 0,6$, $\beta = 0,6$.

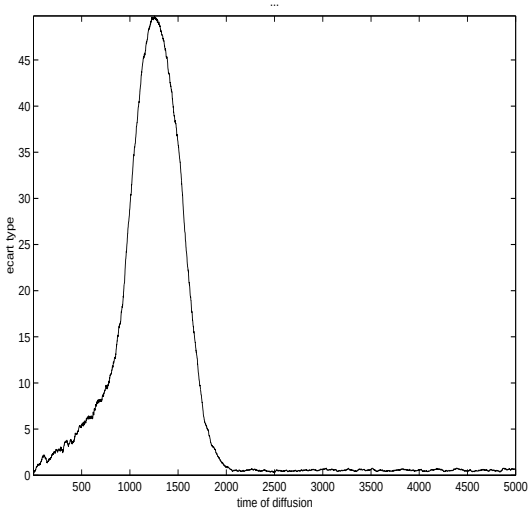
Première colonne : $a = 2,5$, $b = \frac{1}{2}$.



Courbe de diffusion.

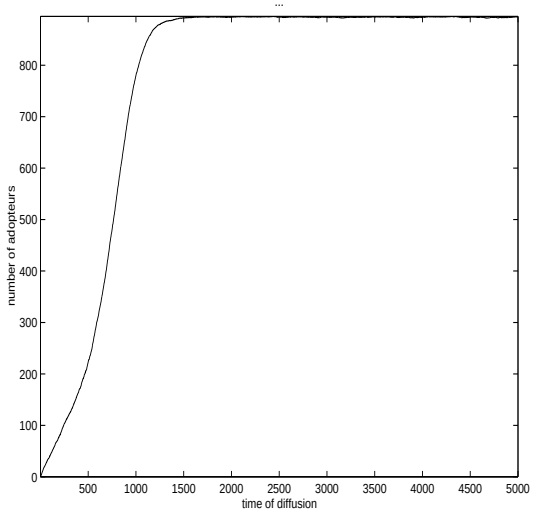


Évolution de la cohérence.

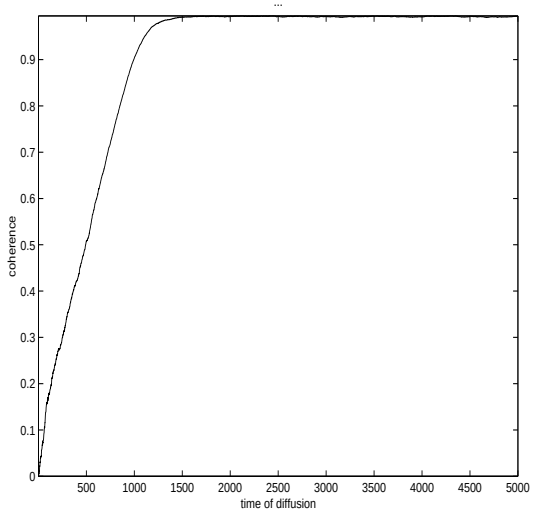


Variation de l'écart-type.

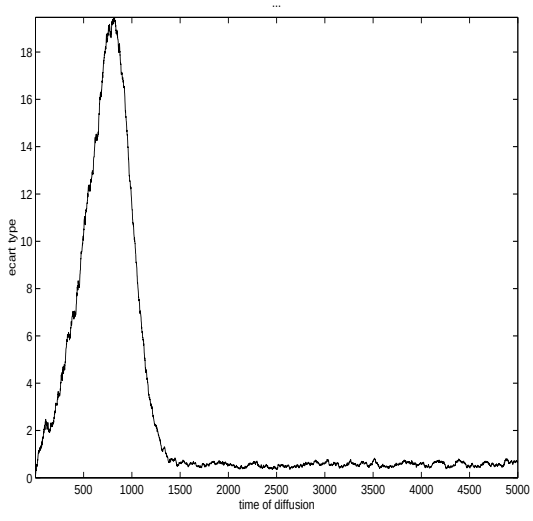
Deuxième colonne : $a = 2,5$, $b = 0,2$.



Courbe de diffusion.



Évolution de la cohérence.



Variation de l'écart-type.

5.2.2 Étude des paramètres favorables au modèle de production «libre».

Différentes distributions des préférences.

Afin de ne pas multiplier les exemples, nous allons présenter deux types de distributions des préférences en plus de la distribution uniforme qui nous sert d'étalon : une distribution normale, centrée en zéro et une distribution bimodale. La première distribution illustre une concurrence où la majorité de la population est indifférente entre les deux technologies, qui pour elle, offrent les mêmes caractéristiques. La seconde représente une population divisée en deux, chaque moitié étant en moyenne très favorable à l'une des technologies. Notons que les trois distributions ont une moyenne nulle, qu'elles sont donc identiques pour un producteur qui ne s'intéresserait qu'à l'agent représentatif de cette population.

Notre but est d'étudier la diffusion et les vitesses relatives de diffusion suivant les valeurs des paramètres a et b, suivant les distributions de préférence. a et b dépendent du produit, de la technologie étudiée, mais aussi de la stratégie des producteurs en ce qui concerne la redistribution ou la «capture» des externalités. La distribution des préférences dépend des qualités intrinsèques des deux technologies, mais aussi de la stratégie des producteurs, qui peuvent choisir d'adopter les innovations du concurrent et donc de produire une technologie proche de la sienne ou, au contraire, de se différencier horizontalement et donc de développer un avantage concurrentiel sur une partie de la population.

Il est clair que la distribution des préférences et la façon dont sont redistribuées les externalités peut changer au cours du temps, mais que ces changements, qui sont essentiellement dus à des évolutions de la stratégie des producteurs, sont difficiles à réaliser car il faut changer les routines de fonctionnement de l'entreprise. Nous supposerons, dans cette partie, que les stratégies des producteurs sont fixes. Ce que nous souhaitons étudier ici, ce sont les structures technologiques favorisant la diffusion des logiciels libres, leur entrée sur un marché déjà dominé par un produit propriétaire, sachant les caractéristiques (a et b) de la technologie.

Nous avons travaillé sur trois cas, le premier où les externalités locales dominent ($b < 0,5$), le second où les externalités globales sont dominantes ($b > 0,5$) et le troisième où elles sont d'égale importance ($b = 0,5$). Pour chaque cas, nous avons considéré deux sous-

cas, suivant que les externalités sont fortes (a grand) ou faibles (a petit). Dans chacun de ces sous cas, les trois distributions des préférences ont été simulées. Les valeurs des temps de diffusion pour l'ensemble de ces cas ont été placées dans l'annexe C.

L'importance respective des effets locaux et des effets globaux.

On peut constater deux phénomènes généraux.

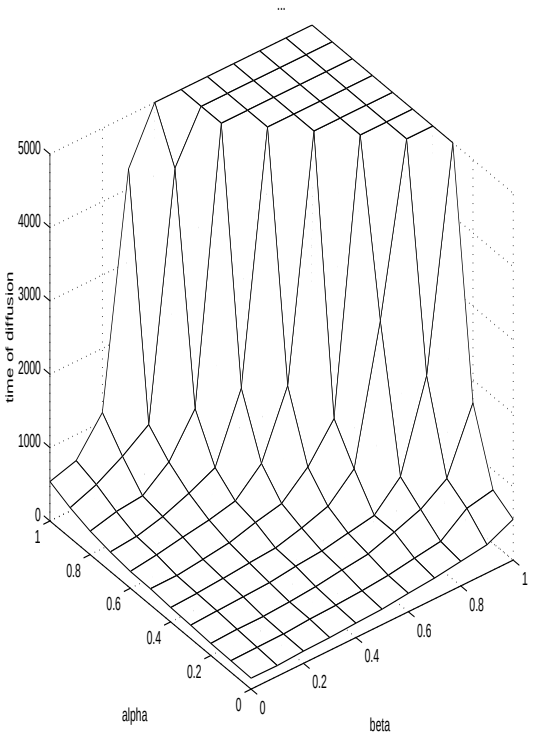
Il y a plus souvent diffusion avec des α grands qu'avec des β grands (voir figure 5.3), il est plus facile de faire basculer une population quand le lock-in technologique est dû à des externalités locales, les individus pouvant changer de proche en proche. On peut l'interpréter de la façon suivante : si le standard dominant redistribue bien les rendements croissants d'adoption globaux, s'il n'abuse pas de son pouvoir de monopole, les individus mécontents restent isolés, les agents sont globalement satisfaits et privilégient l'information globale dans leur choix technologique. Si ce résultat n'est pas plus visible dans notre modèle, c'est essentiellement dû à sa construction : les influences locales dépendent grandement de la probabilité d'avoir des voisins qui ont déjà adopté la technologie, donc de la part de marché globale de cette technologie. Pour renforcer cet impact, il faudrait s'intéresser à la construction des voisinages des agents, à leur regroupement par affinité, etc. Ce serait sans doute une des extensions les plus fructueuses de ce modèle. Mais, parce que les effets de niche sont bien pris en compte, il y a bien un impact.

D'ailleurs, et c'est le deuxième résultat, la diffusion est facilitée quand les externalités locales sont plus importantes que les externalités globales (voir figure 5.4).

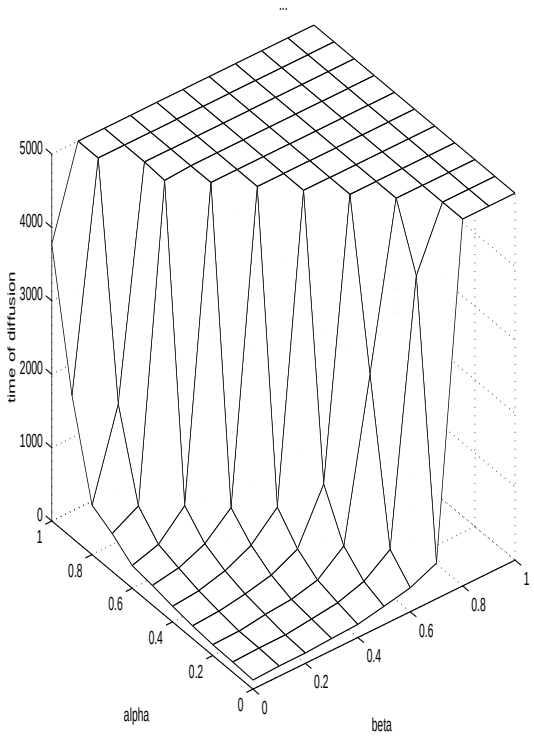
On retrouve là un résultat présenté au chapitre 2 et bien connu des économistes (voir notamment Arthur [1988a], Farrell & Saloner [1985, 1988], Postrell [1986]). C'est un résultat assez intuitif : lorsqu'une technologie n'a pas à tenir compte de la base installée, elle peut plus facilement se diffuser de proche en proche en s'appuyant sur les zélotes pour créer des niches.

On va voir qu'on peut mentionner des résultats plus intéressants pour la diffusion des logiciels libres.

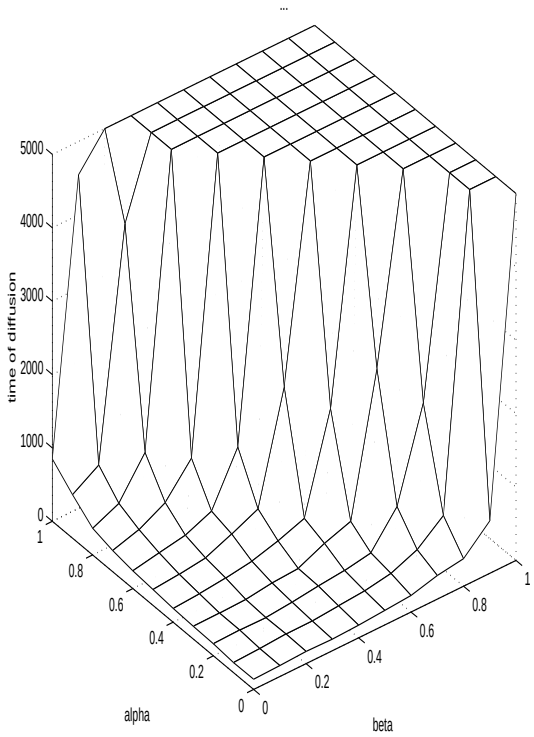
Figure 5.3 — Comparaison des influences locales et globales, $a = 2,5$ et $b = 0,5$.



Distribution uniforme.



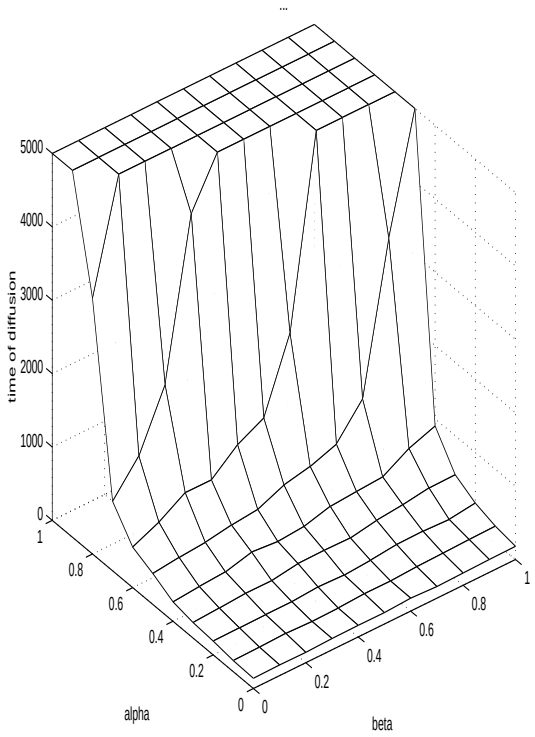
Distribution normale (centrée en 0, écart-type 0,4).



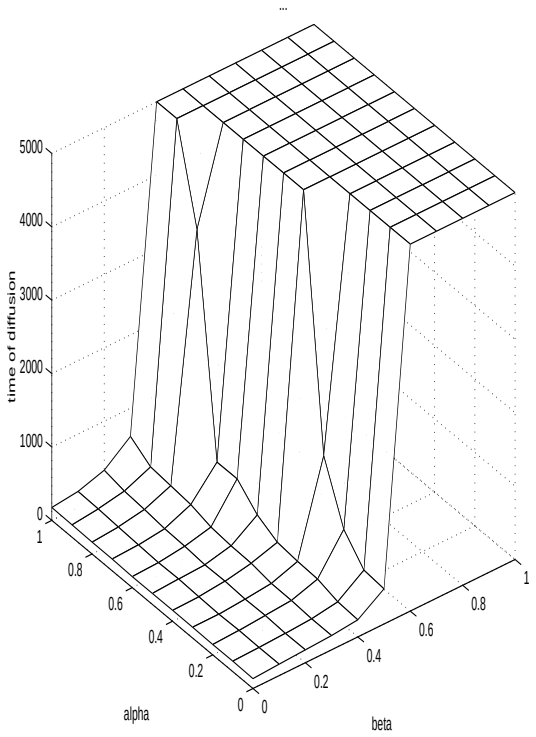
Distribution bimodale (pôles centrés en $-0,5$ et $0,5$; écarts types 0,2).

Figure 5.4 — Influence des effets globaux et locaux sur la diffusion ($a = 2,5$).

Distribution bimodale (pôles centrés en $-0,5$ et $0,5$; écarts types $0,2$).

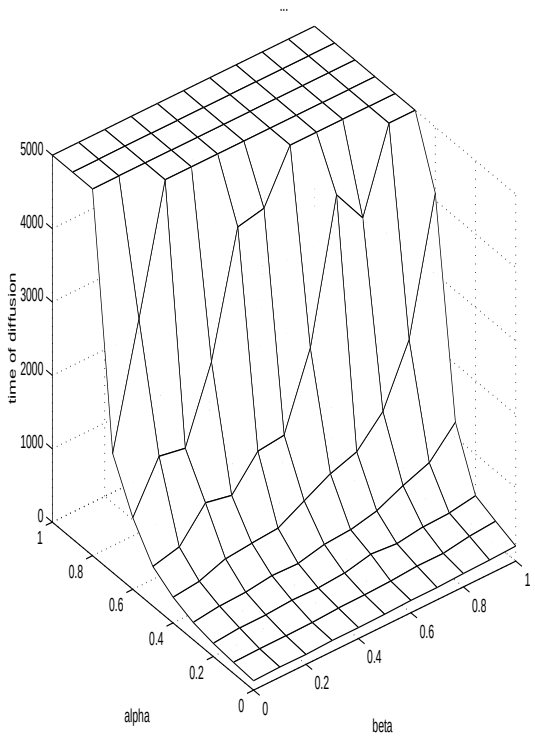


Externalités locales dominantes ($b = 0,2$).

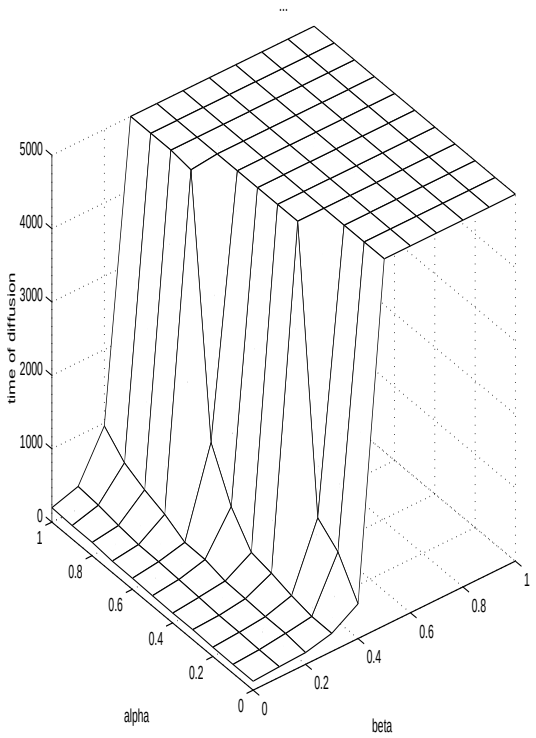


Externalités globales dominantes ($b = 0,8$).

Distribution normale (centrée en 0, écart-type 0,4).



Externalités locales dominantes ($b = 0,2$).



Externalités globales dominantes ($b = 0,8$).

Stratégie des producteurs et standardisation.

La stratégie des producteurs de logiciel libre doit être adaptée à l'environnement technologique et le facteur clef dans la diffusion est le poids des effets de standardisation (voir figure 5.5).

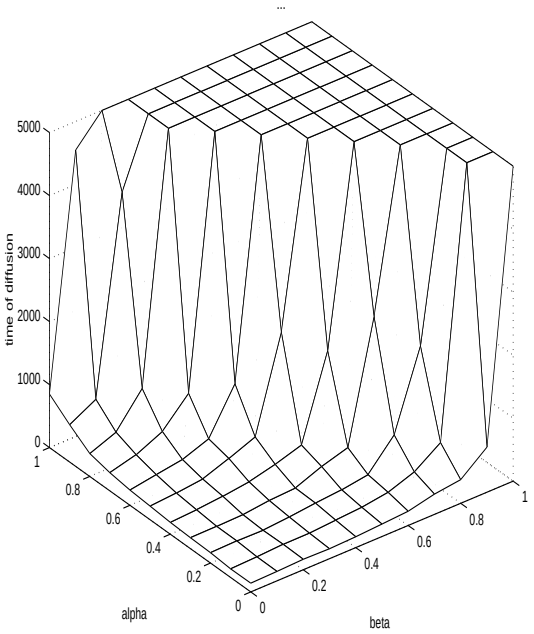
Quand les agents recherchent la standardisation (a grand), il sera plus facile de diffuser en se différenciant (distribution bimodale). À cause des effets de standardisation, la seule chance de conquérir une part de marché significative est de convaincre une partie de la population que le nouveau produit est vraiment meilleur que l'ancien. Il est nécessaire de créer des niches sur lesquelles on va s'appuyer afin de diffuser le produit.

Quand la distribution des préférences est centrée en zéro, le mieux qu'on puisse espérer est une part du marché total, souvent faible si les effets de standardisation sont grands. Ce qui signifie que, pour des logiciels comme Linux, il est plus important d'acquérir d'abord des positions fortes dans certains marchés que de développer les composants qui permettront d'adopter ce système pour en faire une machine de bureau. La diffusion de Linux se fera dans un deuxième temps, lorsqu'une première population d'adopteurs aura été convaincue par ses qualités techniques et cherchera à diffuser ce système afin, par exemple, d'homogénéiser le parc de ses machines. Cela conforte l'analyse que nous faisions dans le chapitre 3 : il est plus important pour les entreprises qui distribuent Linux de conquérir les entreprises et les marchés émergeants (systèmes embarqués, PDA) que de chercher à concurrencer directement Windows dans les ordinateurs grand public.

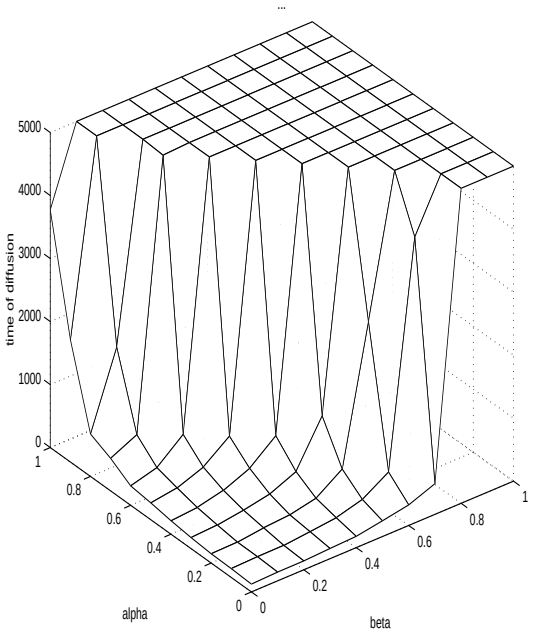
Pour les mêmes raisons, les résultats s'inversent quand la sensibilité aux externalités est faible ($a = 0,5$) et il vaut mieux développer une offre similaire à celle qui existe déjà. Sinon, cette offre n'intéressera qu'une partie de la population. Si la distribution des préférences est centrée, l'avantage concurrentiel se fera sur la façon dont sont redistribuées les externalités. Le nouvel entrant peut conquérir le marché s'il redistribue suffisamment bien ces externalités. On peut en tirer une autre leçon, cette fois-ci pour les producteurs de composants libres (comme Digital Creation, ACT ou MySQL, entreprise qui distribue un logiciel éponyme de gestion de base de données, concurrent d'Oracle). C'est parce que leur service sera plus efficace, qu'ils coordonneront mieux les retours de leurs utilisateurs qu'ils seront plus performants qu'un producteur qui s'appuiera sur un logiciel propriétaire.

Figure 5.5 — Stratégies de compatibilité suivant l'importance des effets de standardisation.

Forte standardisation ($a = 2,5$), externalités locales et globales égales ($b = 0,5$).

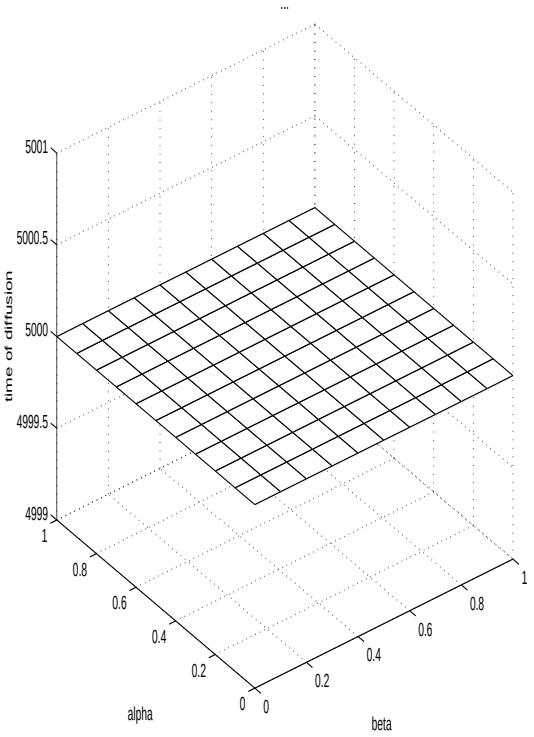


Distribution bimodale (pôles centrés en $-0,5$ et $0,5$; écarts types 0,2).

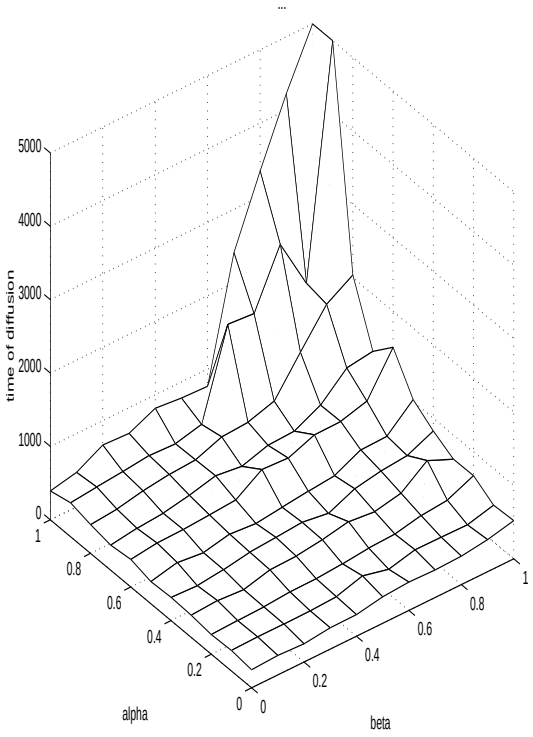


Distribution normale (centrées en 0 ; écart type 0,4).

Faible standardisation ($a = 0,5$), externalités locales et globales égales ($b = 0,5$).



Distribution bimodale (pôles centrés en $-0,5$ et $0,5$; écarts types 0,2).



Distribution normale (centrées en 0 ; écart type 0,4).

Dans ce cas, nous l'avons déjà dit, l'organisation libre ne change pas fondamentalement l'organisation de production puisqu'il y a toujours une entreprise qui, à défaut de posséder, contrôle le logiciel et son développement. Seule la relation de long terme avec les clients est transformée à cause de l'ouverture du code. L'avantage principal du libre est, ici, sa structure ouverte : ouverture des sources, mais aussi modularité et liberté de contribution, qui sont les deux caractéristiques permettant effectivement aux utilisateurs de contribuer.

On voit donc que la stratégie de positionnement est fonction du marché. Comme le faisaient remarquer Katz & Shapiro [1994], la stratégie des entreprises, leur positionnement au début du processus de concurrence en fonction du régime de concurrence auquel elles sont confrontées, est un des, si ce n'est le paramètre fondamental qui caractérise la diffusion technologique.

Après cette première étude «statique» (au sens où les stratégies des entreprises sont statiques, fixées une fois pour toute au début du processus de diffusion), on peut s'intéresser à l'évolution de ces stratégies d'entreprises au cours du processus de diffusion (et l'impact de ces actions sur la diffusion).

5.3 Évolution des stratégies commerciales des entreprises.

Par «stratégie commerciale», nous entendons la façon dont les entreprises redistribuent les externalités dues à l'adoption de leur produit par les utilisateurs. Il nous faut ajouter au modèle précédent la variable temporelle pour pouvoir étudier leur évolution au cours du processus de diffusion. Car, si nous avions écrit la fonction E de la façon suivante :

$$E = E\left[a,\, b * I(X_1,\, X_2, t),\, (1-b) * i(x_1,\, x_2, t)\right],$$

Jusqu'à maintenant, nous l'avons considérée comme indépendante du temps. C'est l'étude de l'évolution de cette fonction de seuil au cours du temps, en fonction de l'évolution des stratégies des entreprises, qui va nous intéresser ici. Nous allons reprendre la même expression pour la fonction de seuil, c'est-à-dire :

$$E = \tanh\left[a * \left[(1-b) * (x_2 - \alpha * x_1) + b * (X_2 - \beta * X_1)\right]\right],$$

mais α et β vont maintenant varier au cours du temps.

5.3.1 Évolution des stratégies commerciales.

Dans notre modèle, toutes les évolutions des stratégies des entreprises se répercutent de manière identique, par des variations de α et de β dans le temps. Si, par exemple, le prosélytisme local des défenseurs de Linux disparaît ou au moins diminue, une adoption locale de Linux va moins compter et α va augmenter, avec tout ce que cela implique sur la diffusion. La question est donc de savoir ce qui se passe quand α augmente pendant la diffusion, ou parallèlement, ce qui se passe si la créativité de la communauté Linux baisse, ou si Microsoft baisse ses prix, ou si Microsoft change son modèle économique (ou au moins sa politique de versions), bref, quand β augmente ?

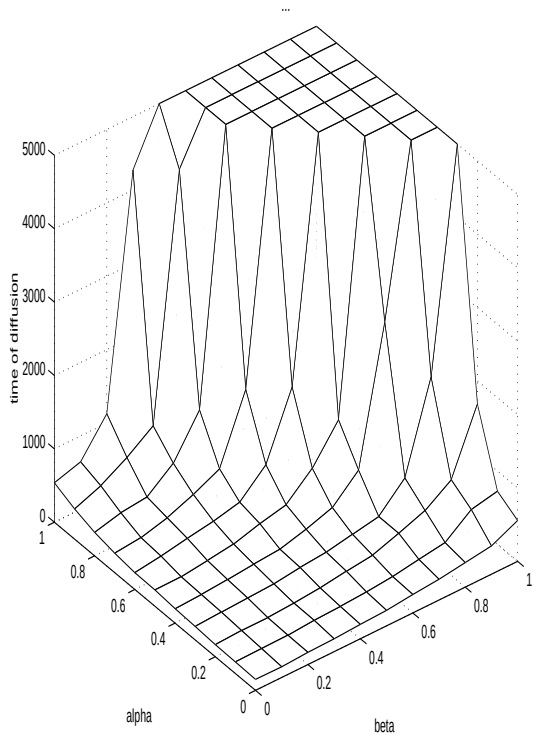
Bien que de telles évolutions soient plutôt rares, car on a à faire à des organisations construites autour de routines, dont les coûts de changement («switching costs») sont très élevés, nous étudions ici le cas d'une diffusion où β est augmenté significativement dès que le nombre des utilisateurs de Linux atteint un certain seuil, défini comme un pourcentage de la population totale des adopteurs[8]. Nous avons repris les mêmes distributions des préférences (distribution uniforme, distribution normale centrée en zéro, distribution bimodale) et nous avons étudié différentes technologies, caractérisées par les sources des externalités (locales ou globales), suivant la valeur du coefficient b) et par l'importance des effets de standardisation (valeur du coefficient a). Quand la proportion des adopteurs de Linux atteint 5 % de la population totale, β est augmenté de 0,2 : la réaction de Microsoft ou la baisse de créativité se produit donc pour un seuil très bas alors que la réaction elle-même est plutôt forte.

De l'analyse des diffusions (dont le détail est toujours présenté en annexe C), nous tirons le résultat suivant : Linux diffuse pourtant de façon quasi-similaire au cas où il n'y

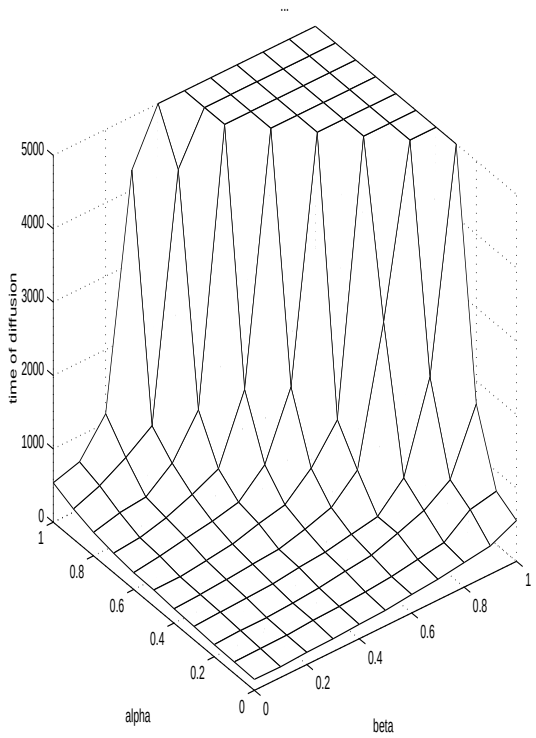
[8]Nous obtenons les même résultats pour une baisse du prosélytisme des «linuxiens» (augmentation de α).

Figure 5.6 — Temps de diffusion de Linux avec une augmentation de β de $0,2$ quand Linux atteint 5 % de part de marché.

Distribution uniforme ($a = 2,5$ et $b = 0,5$).

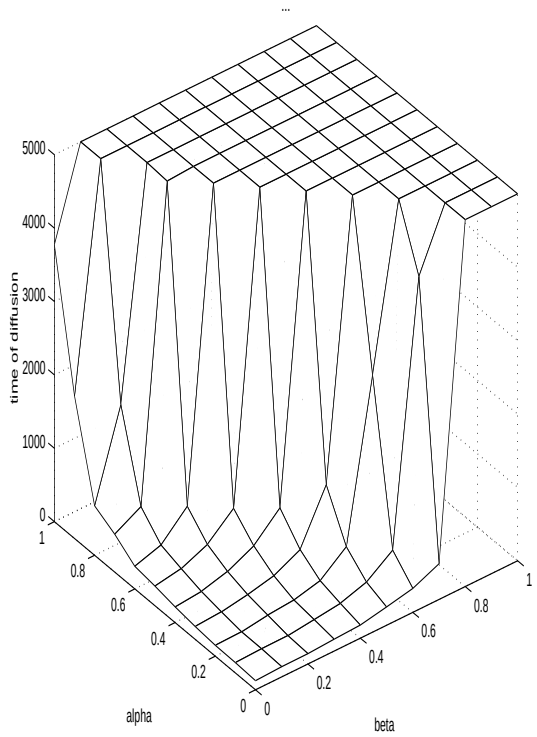


Pas de réaction.

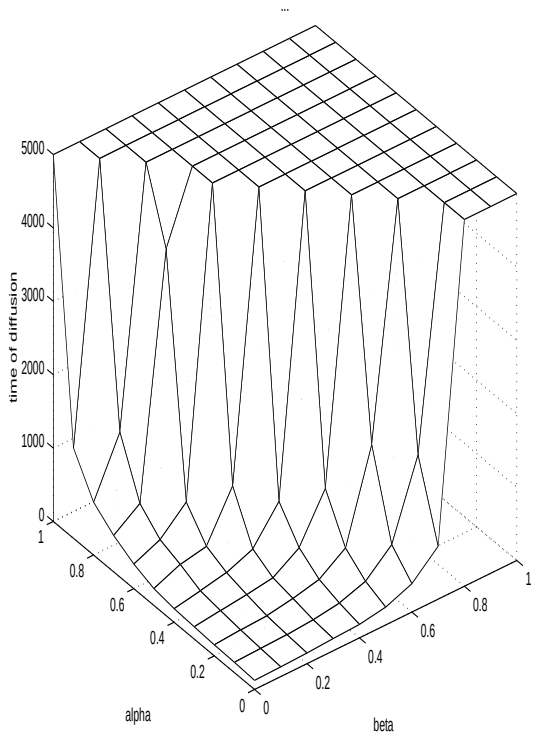


Réaction.

Distribution normale centrée en 0 (écart-type 0,4).



Pas de réaction.



Réaction.

a pas de réaction ou d'évolution dans la créativité (voir figure 5.6).
Plus précisément, augmenter β de 0,2 quand le pourcentage des «linuxiens» atteint 5 % de la population totale dans une diffusion revient à considérer la diffusion avec la variable β' où $\beta' = \beta + \delta$ avec $\delta \leq 0,1$. À cause de la non-linéarité de la dynamique - un très fort attracteur associé à un état méta-stable dans l'espace des trajectoires -, les résultats apparaissent peu sensibles aux possibles variations des paramètres pendant le processus de diffusion[9].

On retrouve ici l'effet de dépendance du chemin (David [1985]) : une fois qu'une diffusion est commencée, il est difficile de la stopper. Ou, pour dire cela différemment, il est très vite déjà trop tard. L'événement historique qu'on attribue à la phobie de Microsoft aura peut-être joué le même rôle pour Linux que celui qu'on fait jouer aux contraintes techniques dans le cas du clavier QWERTY, ou à la volonté des entreprises clientes de créer des concurrents à IBM dans le cas des PC. Il aura bientôt disparu, mais aura orienté de façon irréversible le sentier de diffusion. D'après nos résultats, et puisqu'il semble que la diffusion ait, concrètement, plus que démarré, il est en peut-être déjà trop tard pour W2000.

Pour résister à la concurrence d'un nouvel entrant, plus que leurs stratégies commerciales, les entreprises peuvent faire évoluer le contenu technologique de leurs produit, ce qui veut dire changer la courbe de distribution des préférences. Nous allons étudier cette possibilité dans la sous-section suivante.

5.3.2 Évolution des stratégies d'innovation.

L'étude de la partie précédente a souligné l'importance de la distribution initiale des préférences dans une concurrence entre technologies ou entre offres. Mais l'hybridation peut aussi être une stratégie dynamique. Pendant la diffusion, les producteurs peuvent essayer d'adapter leur offre pour mieux répondre aux besoins des utilisateurs, donc pour changer la distribution des préférences dans la population. Ils peuvent «hybrider» leur produit, leur technologie, en y incorporant des caractéristiques de la technologie concurrente. Cela

[9] Nous pensions initialement qu'on pouvait interpréter ce phénomène comme une conséquence des fortes incitations à la compatibilité (paramètre a), mais d'autres simulations avec des valeurs plus faibles de a donnent les mêmes résultats.

change la distribution des utilités, en diminuant le nombre des adopteurs qui préfèrent la technologie concurrente (puisque le producteur a incorporé dans sa technologie des caractéristiques de la technologie concurrente).

Si l'on se place au départ dans le cas de la distribution bimodale et si l'on considère que la technologie 2 est hybridée, qu'elle acquiert des caractéristiques de la technologie 1, le pic de droite va se déplacer vers la gauche, se rapprocher de zéro : les supporters de la technologie 1 vont devenir de plus en plus indifférents au fur et à mesure que la technologie 2 incorpore les caractéristiques qui la leurs faisaient préférer. Nous pouvons tester la sensibilité des trajectoires de diffusion aux stratégies d'hybridation, notamment la situation où un producteur choisit d'hybrider sa technologie plus ou moins vite alors que l'autre producteur a une stratégie «pure» de différenciation.

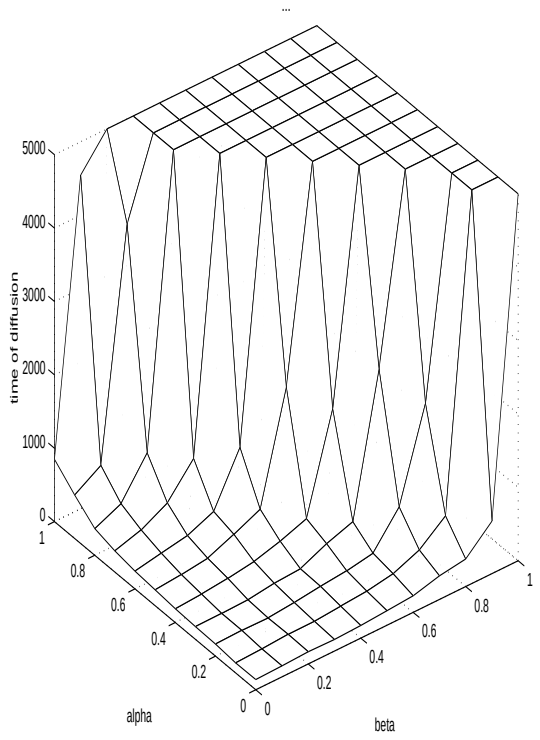
Hybridation de la technologie de l'entrant.

La première situation qu'on peut tester est celle d'un entrant qui hybriderait sa technologie pour capturer une plus grande part de la population qui préfère strictement la technologie installée. Nous avons simulé d'abord le cas d'une diffusion dans une population dont la distribution des préférences est bimodale. Pendant la diffusion, le centre du second mode se décale vers le centre de la diffusion. La conséquence de cette stratégie est que les préférences deviennent, en moyenne, favorables à la technologie entrante. Nous présentons ici (figure 5.7) le cas où, à intervalle de temps régulier (ici tous les 200 intervalles), le second mode de la bimodale est déplacé vers zéro (ici, de 5 % de la distance entre le centre du mode et zéro). Le résultat général est qu'il n'y a que peu de changements en terme de diffusion.

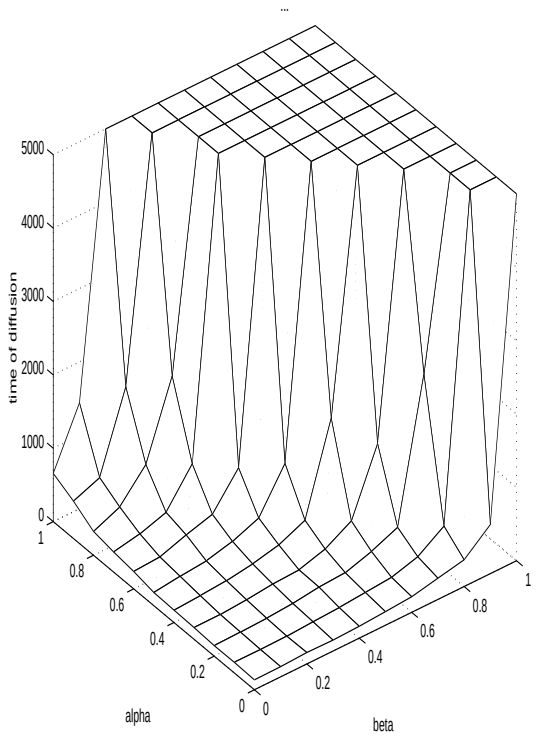
Ce résultat est assez intuitif, au vu des résultats précédents : avant d'essayer d'attirer des utilisateurs qui préfèrent l'autre technologie, il faut s'assurer que ceux qui préfèrent la vôtre l'adoptent car, de toute façon, il le feront avant les autres. Une fois qu'ils ont adopté, ils peuvent représenter une population suffisamment importante pour laisser faire les effets de réseau.

La situation est complètement différente quand c'est la technologie dominante qui essaie de s'hybrider.

Figure 5.7 — Stratégie d'hybridation du nouvel entrant contre stratégie «pure».
Distributions bimodales centrée en $-0,5$ et $0,5$, écart-type $0,2$, $a = 2,5$, $b = 0,5$.



Pas d'hybridation.



Hybridation par déplacement de 5 % tous les 200 coups du pôle de droite.

Hybridation de la technologie dominante.

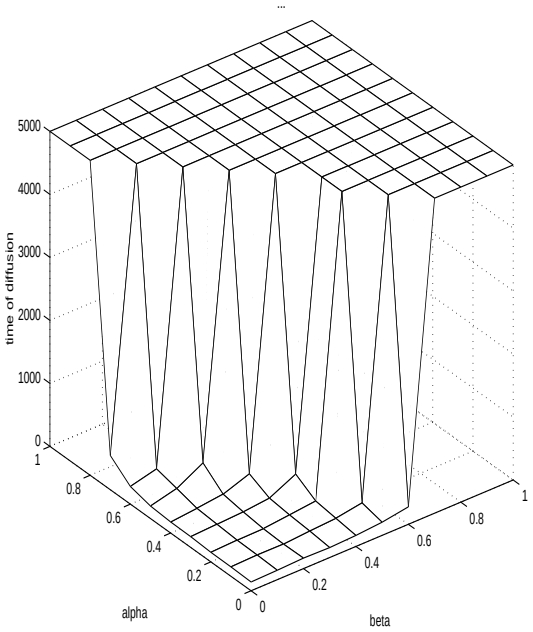
Dans ce cas, l'avantage concurrentiel du nouvel entrant diminue en même temps que diminue la population des adopteurs précoces. Cela diminue les chances que la technologie entrante puisse atteindre le seuil suffisant pour diffuser. Cette stratégie a un impact fort sur le processus de diffusion (figure 5.8) et sur la capacité d'une nouvelle technologie, d'une nouvelle organisation, à s'imposer.

Ce qu'on peut en déduire sur la concurrence dans le logiciel.

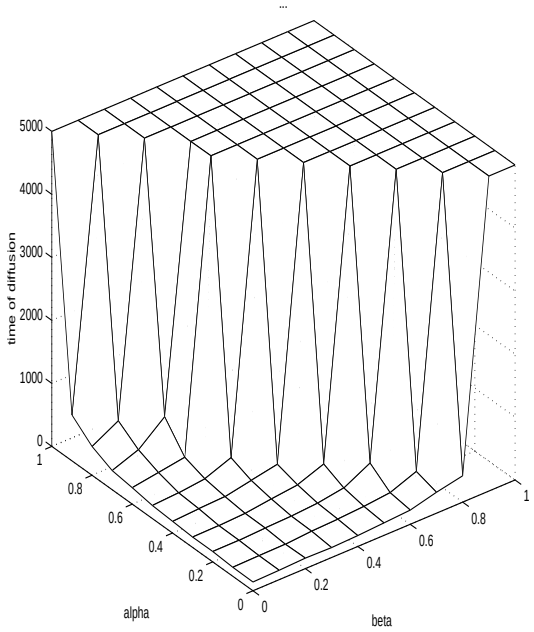
Cela nous amène à faire deux remarques sur les caractéristiques de la concurrence dans le logiciel : l'organisation de production Libre est construite pour favoriser l'innovation marginale autour d'un noyau technologique et le développement de composants

Figure 5.8 — Stratégie d'hybridation de la technologie déjà installée contre stratégie «pure».

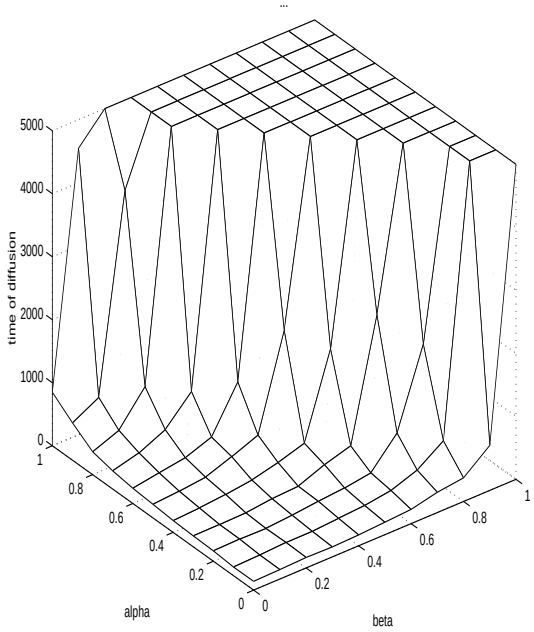
Distributions bimodales centrée en $-0,5$ et $0,5$, écart-type $0,2$, $a = 2,5$, $b = 0,5$.



Hybridation par déplacement de 5 % tous les 200 coups du pôle de gauche.



Hybridation par déplacement de 5 % tous les 1000 coups du pôle de gauche.



Pas de réaction.

qui augmentent les services rendus par la technologie. Une fois qu'un logiciel libre est en place, il sera d'autant plus difficile de construire un concurrent car les solutions techniques innovantes proposées par le concurrent, les nouvelles fonctionnalités, sont rapidement incorporées au logiciel libre.

L'autre sens est moins vrai : s'il est plus facile de copier un logiciel libre, parce que le code est disponible et parce qu'il est construit en modules, réutiliser des modules peut obliger à rendre libre tout ou partie du logiciel (c'est l'effet contaminant de la licence GPL) et surtout, l'adaptation de ces modules à un logiciel qui n'est pas structuré de la même façon, qui n'utilise pas forcément les même standards d'échange entre modules, reste coûteuse. Remettre en cause la structure du produit, flexibiliser son offre nécessite de changer l'organisation de production de l'entreprise, donc, une fois de plus, les routines, les règles internes à l'entreprise. C'est long et difficile. L'exemple de SUN montre assez bien ce fait : SUN a d'abord cherché à créer une licence spécifique pour ouvrir et distribuer le code de ses logiciels (Star Office et Solaris), licence qui lui permettait de s'approprier les contributions des utilisateurs et de continuer à percevoir des royalties sur la vente de ses logiciels. La stratégie de SUN a évolué, puisque les prochaines versions de Star Office seront, dans certains cas, proposées sous la licence GPL. Mais cette entreprise rencontre des difficultés à faire ce passage car elle n'est pas propriétaire de l'ensemble du code des logiciels (il faut alors les réécrire) et elle n'a pas encore abandonné l'idée de faire payer des royalties aux entreprises qui utiliseraient ces logiciels pour développer des produits commerciaux[10].

5.4 Conclusion.

Ce que nous enseigne ce modèle est qu'une innovation technologique mineure peut s'avérer suffisante pour remplacer un standard dominant quand elle est associée à un modèle économique supérieur. Car, pris du point de vue de la technologie, Linux n'est pas significativement supérieur à Windows, or David [1985] et d'autres auteurs ont montré

[10] Voir : http ://www.sun.com/981208/scsl/principles.html ;$sessionid$H5BZ2EQAABLSJAMTA1LU3NQ pour une présentation de la licence de SUN et des explications technico-commerciales que cette entreprise donne sur l'avantage de cette licence. Voir aussi le site de la version libre de Star Office (sous GPL), renommée «Open Office» : http ://www.sun.com/software/star/openoffice/ ;$sessionid$H5BZ2EQAABLSJAMTA1LU3NQ.

qu'une technologie, même supérieure, pouvait ne pas réussir à s'imposer face à un standard installé. C'est qu'ici la supériorité ne provient pas d'une supériorité technologique, mais d'un meilleur modèle économique. Le modèle Libre fait, d'une part, une très bonne utilisation de la créativité, qui favorise (et est favorisé par) le développement d'entreprises de support, permettant le développement de produit efficaces et qui créent une structure d'incitation et de récompense pour les développeurs. Ce modèle est, d'autre part, plus efficace pour redistribuer les rendements croissants d'adoption aux utilisateurs, ce qui favorise sa diffusion.

Des aspects non techniques s'avèrent suffisamment importants pour permettre à une technologie qui n'est pas supérieure de vaincre un standard. C'est un point très important car, au delà du fait qu'il confirme que des standards peuvent être vaincus par des technologies alternatives, il remet en avant le fait que des paramètres explicatifs plus proches des facteurs économiques que des caractéristiques techniques pures sont aussi des candidats pour expliquer des évolutions de standards. Nous avions constaté ce fait dans l'histoire de l'informatique, nous voyons qu'il peut se répéter quand les rendements croissants d'adoption sont essentiellement dus à des externalités de réseau, locales et globales, ce qui est nouveau dans cette industrie.

Ce qui caractérise les trajectoires technologiques est alors plus proche de la persistance (Foray [1997]) que d'une dépendance pure et irréversible au chemin : la diffusion technologique crée de façon endogène des effets de seuil. Les technologies suivantes doivent être «suffisamment meilleures» pour surpasser les standards existants, c'est-à-dire surmonter un seuil créé précédemment de façon endogène. Comme Dalle [1998] le défend, ce phénomène offre une explication très solide à la distinction classique, mais essentiellement empirique, entre innovations majeures et mineures : les innovations majeures sont celles qui sont suffisamment meilleures pour se diffuser au delà de ce seuil non linéaire, alors que les innovations mineures sont en dessous de ce seuil et ne se diffuseront jamais. Par conséquent, les innovations majeures, qui peuvent être dues à des causes non techniques comme c'est le cas pour le logiciel libre et son nouveau modèle économique, peuvent ne pas être compatibles avec le standard dominant. C'est socialement optimum puisque cette nouvelle technologie est justement suffisamment meilleure pour compenser les coûts élevés

du changement collectif. Les innovations mineures, au contraire, doivent rationnellement se rendre compatibles avec le standard dominant si elles veulent avoir une chance de diffuser : l'existence de seuils crée de façon endogène des incitations à la compatibilité pour celles-ci. Par conséquent, le standard existant est amélioré continuellement au lieu d'être sans cesse concurrencé par des alternatives peu enthousiasmantes, ce qui est, là aussi, socialement optimal.

Finalement, la concurrence entre technologies et les effets de lock-in sont très sensibles à la stratégie des producteurs vis à vis des externalités et des investissements. S'ils assurent une redistribution correcte de ces externalités, il sera difficile pour un nouvel entrant de créer des niches suffisamment importantes pour surmonter l'effet de masse global favorable à la technologie dominante. Mieux, si le standard est suffisamment flexible pour intégrer les innovations proposées par l'entrant, il est presque impossible pour celui-ci de s'imposer. C'est évidemment un avantage pour le logiciel libre, organisation qui favorise l'imitation. Par contre cela peut poser un problème d'incitation initiale à innover, si cette innovation est trop facilement imitée.

L'avantage du système de protection intellectuelle par le droit d'auteur est qu'il permet aux producteurs qui innovent de choisir le système qui leur convient le mieux. C'est ce qui fait sa force dans le cas de la production de ce bien si particulier qu'est le logiciel. C'est pourquoi nous pensons qu'il est important de continuer à étudier le phénomène du logiciel libre, la façon dont il encourage l'innovation avant de changer les règles institutionnelles qui protègent le logiciel, en introduisant un protection par le brevet, par exemple.

Conclusion générale

L'OBJECTIF de ce travail était de proposer une analyse économique de la production et de la diffusion des logiciels libres. Bien que ce système de production semble aller à contre courant des modes traditionnels, nous avons défendu l'idée que c'était une organisation économique efficace et pérenne, que nous avons appelée l'«organisation Libre» ou, plus simplement, le «Libre». Pour appuyer cette idée, nous avons d'abord étudié la façon dont il fonctionnait, avant de comparer ses mérites à ceux du système dominant actuel (l'organisation «Propriétaire»), et de nous intéresser à sa diffusion.

Après avoir rappelé brièvement les éléments qui nous ont conduit à cette analyse, nous poserons la question de la pertinence d'une action de la puissance publique pour soutenir ce nouveau modèle de production de la connaissance et de sa diffusion à d'autres domaines industriels.

Notre travail a montré que la production de logiciels libres a été initiée par des utilisateurs «designers» de l'outil informatique, capables de concevoir et de développer des logiciels, parce qu'ils n'étaient pas satisfaits de la qualité de ceux produits par les entreprises. Pour assurer cette production, ils ont mis en place une organisation très structurée, tant sur le plan de l'architecture du logiciel, que de l'organisation de sa production : autour d'un coeur, un noyau logiciel, contrôlé par un ensemble restreint de développeurs, d'autres groupes de développement construisent des «modules» périphériques. Le caractère ouvert et l'unicité du noyau garantissent la cohérence du logiciel, l'interropérabilité des composants et préviennent l'apparition de versions incompatibles. En même temps, la possibilité de construire des modules permet d'adapter le fonctionnement du logiciel aux besoins spécifiques de chaque utilisateur. Un des principaux avantages de cette organisation est qu'elle assure la normalisation des interfaces et garantit que cette normalisation restera publique. Mais c'est aussi une organisation de normalisation de qualité car le code des logiciels peut être inspecté (et, si nécessaire, amendé) avant que d'être utilisé ou au cours de l'utilisation, si une erreur est découverte.

Et, parce que tous les utilisateurs peuvent non seulement modifier un logiciel libre, mais aussi le redistribuer librement, les entreprises trouvent paradoxalement, un intérêt à contribuer au développement de ces logiciels libres : on pourrait penser que la libre dif-

fusion est un frein à l'engagement des entreprises utilisatrices (leurs concurrents peuvent profiter gratuitement de leurs investissements pour développer le logiciel) et productrices (qui peuvent plus difficilement vendre les logiciels qu'elles produisent). Ce paradoxe est levé si l'on considère qu'un part importante des coûts des entreprises utilisatrices (et des revenus des entreprises productrices) provient d'activités de services : adaptation des logiciels aux besoins propres des utilisateurs et surtout maintenance de ces logiciels. Le Libre, parce qu'il permet de reverser les adaptations réalisées au «pot commun», est en fait un système dans lequel les entreprises peuvent externaliser une partie de leurs coûts de maintenance, sans que le désavantage concurrentiel soit très important (car ces contributions sont souvent marginales). Les producteurs y trouvent aussi un avantage car l'efficacité de l'industrie du service informatique, bien faible actuellement, s'en trouvent améliorée : le Libre abaisse les asymétries d'information sur l'efficacité et l'engagement des producteurs de services[11]. Face au problème de la sélection adverse, parce que le logiciel est ouvert et qu'on peut inspecter alors le travail des entreprises, il construit un système d'assurance qualité; face au hasard moral, parce que le logiciel est librement utilisable, et que l'on peut donc changer plus facilement de producteur de logiciel, il rend l'engagement des producteurs sur la qualité de leur service plus crédible. Le Libre assure aussi la rentabilité de ces producteurs car le logiciel est une connaissance codifiée cumulative. Ce qui veut dire qu'il faut apprendre le code pour accéder à la connaissance, et qu'il faut suivre l'évolution de ce code pour pouvoir, à tout moment, adapter le logiciel et garantir la stabilité de ces adaptations. Cette nécessaire et perpétuelle accumulation de connaissances est plus facilement réalisable dans les entreprises de service, qui peuvent entretenir des développeurs aux compétences variées et amortir le coût des développeurs sur l'ensemble des projets qui leur sont confiés. Donc, entre faire et faire-faire le suivi des logiciels qu'elles utilisent, les entreprises utilisatrices ont, le plus souvent, intérêt à faire faire. Toujours à cause de ce que nous qualifierons d'«économie d'échelle» dans la compétence (et notamment de ce que les développeurs capables de proposer des modifications des logiciels libres acceptables par le noyau sont peu nombreux) et des effets d'apprentissage, ces entreprises

[11] Les problèmes qui se posent à l'industrie du service informatique sont similaires à ceux que rencontrent les vendeurs et les acheteurs de voitures d'occasion, des «lemons», rendus célèbres par Akerlof [1970] : parce que la qualité de l'offre est difficilement observable, les acheteurs ont tendance à sous payer les offres de bonne qualité. Cela entraîne un retrait de ces offres, qui ne sont pas rentables, donc une baisse de la qualité moyenne du marché, ce qui diminue encore la disposition à payer des utilisateurs, etc.

de services seront peu nombreuses, ce qui limite la concurrence entre elles et assure leur rentabilité. L'ouverture favorise enfin la coordination des différents producteurs qui interviennent dans le processus qui fait correspondre à un besoin (ce que nous avons appelé une «caractéristique d'utilisation»), une réponse en terme de produit informatique : les constructeurs de composants, qui en connaissent le fonctionnement intime, sont capables de le faire évoluer ; les constructeurs d'architecture (les «architectes»), qui connaissent les caractéristiques fonctionnelles des composants, sont capables de les inter-connecter afin de construire des technologies d'utilisation ; enfin les «traducteurs», qui connaissent les besoins de leurs clients, savent les traduire en terme de technologies d'utilisation.

En-delà de l'amélioration des relations de services, cette organisation est apparue plus efficace que l'organisation actuelle (que nous avons appelée le «Propriétaire») pour diffuser les innovations des utilisateurs et des chercheurs, pour inciter les producteurs et les utilisateurs à proposer des innovations incrémentales et pour organiser le processus de standardisation. En résumé, si l'organisation Propriétaire reste la plus efficace (au sens de Pareto) pour inciter les producteurs de logiciels à créer de nouveaux logiciels (et notamment des logiciels pour lesquels les revenus de service sont faibles, comme les jeux), cette nouvelle organisation peut devenir l'organisation dominante de la production de logiciels, qui s'oriente aujourd'hui vers la production de composants standards adaptables aux besoins de chaque entreprise (ce que Horn [2000b] a appelé le «sur-mesure de masse»). En effet elle est meilleure (toujours au sens de Pareto) pour produire les composants techniques, les composants en forte évolution et les composants supportant le plus d'effets de réseau ou d'effet d'interrelation technologiques, c'est-à-dire la plupart des logiciels utilisés dans les entreprises (et notamment le système d'exploitation, les logiciels qui font fonctionner les réseaux informatiques et plus généralement tous les outils de traitement de l'information).

Mais, et c'est le principal frein à la diffusion du libre, le système Propriétaire et ses produits sont les standards actuels. Le changement de standard est particulièrement difficile en informatique à cause des rendements croissants d'adoption (particulièrement des effets d'apprentissage, d'interrelations technologiques et des externalités de réseau). Plus précisément, si sur le marché des outils, des technologies de base de l'informatique,

le modèle Libre est une évolution de l'organisation actuelle et devrait se réaliser assez facilement, la diffusion sur le marché des applications «grand-public», des outils de traitement de l'information, sera plus longue, et passe par l'adoption de ces logiciels par les grandes entreprises (seules capables de supporter les coûts du changement de standard et d'adaptation des logiciels aux besoins des utilisateurs «naïfs»). Il apparaît qu'une fois de plus, c'est la diffusion (ou non) des systèmes d'exploitation libres qui permettra (ou non) la diffusion de l'organisation Libre et donc la réorganisation de l'industrie informatique, autour d'un pôle libre (qui remplacerait les différentes offres Unix qui existent actuellement), capable de concurrencer et peut-être, à long terme, de remplacer l'offre de logiciels propriétaires construite aujourd'hui autour des systèmes d'exploitation de Microsoft.

La puissance publique jouera certainement un rôle important dans la diffusion des logiciels libres et donc du Libre, par son poids en tant qu'utilisateur d'outils informatique, mais aussi en tant que financeur de la recherche publique et en tant que régulateur de la concurrence.

Tout d'abord, comme tout grand utilisateur de l'informatique, l'État doit se poser la question de l'adoption de logiciels libres et de l'initialisation de projets en ce sens, pour garantir son indépendance vis-à-vis des fournisseurs, pour améliorer la sécurité des logiciels utilisés (particulièrement dans les activités sensibles comme les activités de défense) et l'accès par tous aux documents publics, car en utilisant des logiciels libres, on peut s'assurer qu'ils respectent les normes d'échanges. Il n'y a là aucune spécificité, ces besoins et ces exigences sont celles de toutes grande organisation en interaction permanente avec des agents extérieurs. Seule la taille de l'État donne à cet engagement un poids particulier.

De plus, parce que l'État finance aussi la recherche publique, et qu'elle est une des trois sources d'innovation identifiées, il a aussi un rôle à jouer dans la façon dont sont diffusés les logiciels issus de cette recherche. Nous avons aussi vu l'importance de l'Université comme lieu de formation donc de transferts de technologies entre des chercheurs et praticiens capables de les intégrer dans des produits et composants technologiques. Cela n'est pas propre au logiciel, mais il nous semble que la production publique de logiciel,

lorsqu'elle concerne des outils techniques, des composants soumis à des rendements croissants importants, des composants évolutifs, doit garantir l'accessibilité de ces composants, leur normalisation. Et le Libre paraît le plus adapté dans ce cas.

Ce qui nous amène à réfléchir au troisième rôle de l'État, qui est de construire et de garantir le système institutionnel qui organise la concurrence. Même si ce modèle économique est encore en construction, il propose une façon originale et alternative de résoudre les dilemmes posés par l'économie de la créativité. Ses avantages et ses inconvénients doivent évidemment être comparés à ceux d'autres modèles sur une période longue. Or la souplesse du droit d'auteur (ou de copyright) permet la coexistence de ces deux stratégies. Ce système a été construit pour donner un cadre à la production marchande de logiciel, qui prenait le pas sur la production scientifique. Et le modèle économique Libre nous semble d'autant plus intéressant que c'est une évolution des modèles existants, conduite par les besoins d'utilisateurs insatisfaits du fonctionnement des institutions organisant l'économie de la créativité. C'est pourquoi, avant de vouloir changer l'organisation institutionnelle de l'informatique, en rajoutant une nouvelle couche de protection avec le brevet, il nous semble important de commencer par essayer d'encourager l'évolution que propose l'organisation Libre.

Ce travail peut apporter sa contribution à l'industrie informatique, elle l'est aussi pour l'ensemble de l'organisation économique car le Libre propose un système original de gestion de l'innovation et de création des connaissances. Comme il n'est pas possible de tracer une frontière entre ce qui est une connaissance «pure» (qui appartiendrait à tous et qui ne devrait pas être protégée) et ce qui est une connaissance appliquée (dont il faudrait encourager la création en permettant sa protection)[12], la décision d'accorder un droit propriété intellectuelle sur une connaissance doit se fonder sur des critères autres que les caractéristiques de celle-ci. Ces critères peuvent relever de l'éthique (la brevetabilité du vivant) ou de l'économique (comment construire le système d'incitation à l'innovation le plus efficace ?). Ce que le Libre montre, c'est que dans certains secteurs économiques, diminuer

[12]La découverte des nombres premiers s'est accélérée ces dernières années, essentiellement parce que ces nombres sont utiles dans les algorithmes de cryptographie. Faut-il alors considérer que ce qui était connaissance pure est devenue connaissance appliquée ?

fortement le niveau de protection intellectuelle et organiser la mutualisation des connaissances peut augmenter le niveau de production d'innovations et l'efficacité générale du système de production. Si cette mutualisation n'est pas un fait nouveau, c'est la première fois qu'un tel système est institutionnalisé au niveau d'une industrie dans son ensemble et non pas simplement d'un groupe de producteurs. Le fait qu'il fasse évoluer les institutions gouvernant la production de connaissance ne doit pas être vu comme quelque chose d'étrange. À l'origine, les lettres patentes ont été créées pour importer des technologies qui existaient déjà ailleurs (David [1993]) et les scientifiques avaient initialement comme tâche d'accroître la réputation des princes, ce qui nécessitait de rendre publique les découvertes (David [1995]). Si nous oublions un instant que nous sommes nés dans un monde où les brevets et la communauté scientifique existaient depuis longtemps, peut-être que le phénomène Libre nous apparaîtrait alors complètement naturel. Au moment où l'économie est de plus en plus fondée sur la connaissance, la question de leur mutualisation dans un contexte marchand a toute les chances de se voir posée dans un nombre croissant de domaines et d'activités, à commencer par les biotechnologies. Il ne faudrait donc pas sous-estimer ce que l'exemple du Libre peut apporter à ce que David [1995] a appelé l'«économie du savoir» et Romer [1993] l'«économie des idées».

Annexes

Annexe A.

Production et diffusion des logiciels libres ; quelques chiffres.

Il y a plusieurs manières d'évaluer l'importance du phénomène Libre : on peut s'intéresser à la dynamique de production, en évaluant la quantité et la diversité des logiciels produits, le nombre de développeurs impliqués ou on peut s'intéresser à l'impact sur la demande en mesurant l'utilisation de tel ou tel logiciel libre. Ces deux mesures se complètent bien évidemment. Ce sont les deux mesures de l'efficacité d'un système de production : est-ce que le Libre est efficace pour produire du logiciel et est-ce que ces logiciels sont en adéquation avec les besoins des utilisateurs. Nous allons commencer par l'étude de la production, avant celle de la diffusion des logiciels produits.

La production de logiciels libres.

Il existe de trop nombreux projets libres[13] pour que nous puissions les analyser en détail, ce qui n'aurait d'ailleurs pas grand intérêt. Mais on peut essayer d'évaluer l'offre en terme de caractéristiques d'utilisation, puis en terme de volume de logiciel produit. Enfin, on peut s'intéresser à la force de production, c'est-à-dire à nombre de producteurs impliqués dans ces projets.

L'offre de logiciels libres.

L'objectif de Richard Stallman, qui a initié le mouvement «Libre» en 1983, était de produire un «système d'exploitation» entièrement libre, c'est-à-dire un ensemble de logiciels comprenant le noyau[14], mais aussi des logiciels qui permettent d'utiliser ce système (interfaces graphiques, etc.) et de développer des programmes avec (compilateurs/interpréteurs, éditeur de texte, logiciel de courrier, bref tous les outils utilisables par un individu qui programme). Avec le développement du noyau Linux, dont la première version «stable[15]» date de 1994, on peut dire que cet objectif a été réalisé.

Depuis, outre l'amélioration des «briques de base» (le développement d'interfaces gra-

[13] Le projet Orbiten [2000] en a analysé 3.149 ; si l'on compte le nombre de projets suivis dans «Sourceforge», on dépasse allègrement les 10 000 projets (sans tenir compte des doublons).

[14] Le «cœur» du système d'exploitation, la partie logiciel qui est chargée de contrôler les différents composants de la machine et de les faire exécuter les calculs demandés par les programmes informatiques. Il doit être adaptée à chaque type de machine.

[15] C'est à dire une version utilisable, dont les principales erreurs (les «bogues») ont été corrigées et dont on estime qu'elle fonctionne correctement.

phiques, l'amélioration de la stabilité et l'extension des fonctionnalités du noyau), les développeurs se sont attaqués au développement de logiciels d'application tels que les suites bureautiques, les logiciels de traitement d'images (Gimp), les jeux (Free Civ), les logiciels multi-média, et à l'amélioration de la compatibilité de ces logiciels avec les offres existantes (on peut, aujourd'hui, faire fonctionner Windows ou les logiciels développés pour Windows sur GNU/Linux)[16]. De plus, de nombreux logiciels propriétaires comme Netscape, développés pour fonctionner avec des systèmes d'exploitation ayant des caractéristiques proches des systèmes libres, fonctionnent aussi avec ces systèmes.

On peut dire qu'aujourd'hui l'offre de logiciels libres se rapproche en qualité et en diversité de l'offre des grands standards propriétaires, Unix ou Windows. Passons au volume de logiciels produits et l'évolution de cette production dans le temps.

Le volume de logiciels produits.

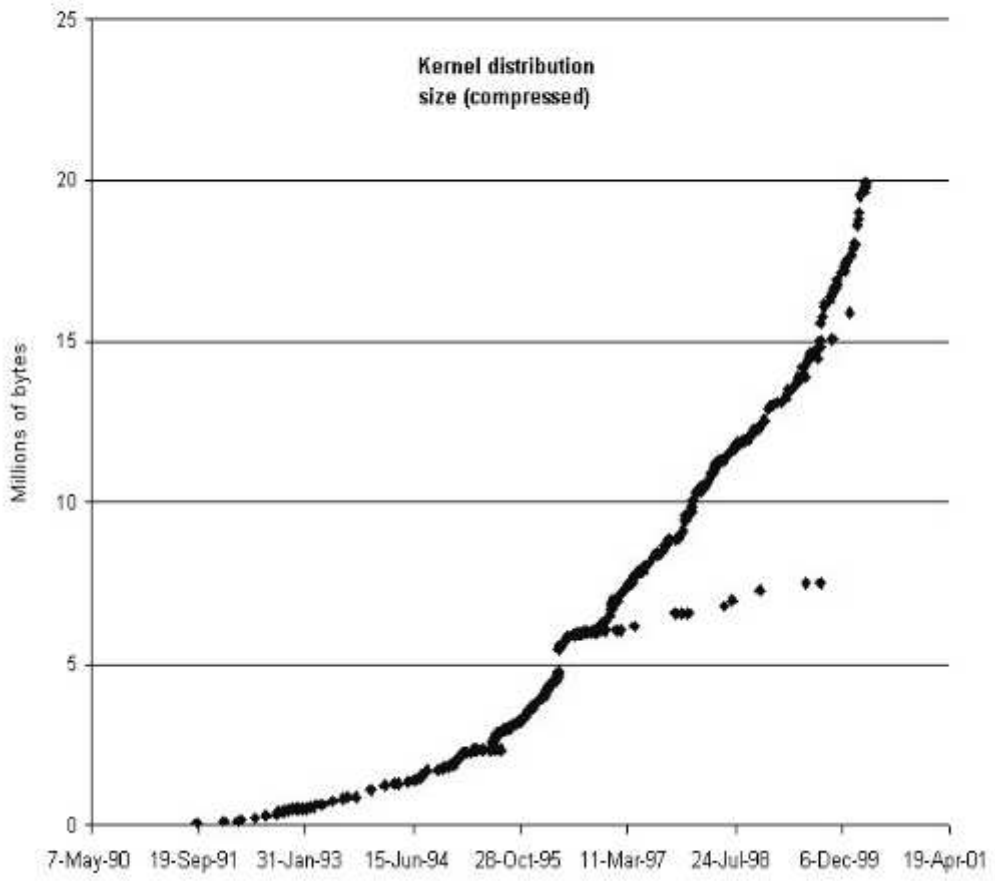
Le projet «Orbiten» (Orbiten [2000]) a analysé un volume de code représentant 1,04 Giga-octets de données, alors que 50 %, seulement des projets présents sur Freshmeat ont pu être analysés[17]. Cette production est toujours en croissance, et en croissance forte pour certains logiciels phares : la taille du noyau Linux croît exponentiellement (voir la figure) ; le projet Gnome (http ://www.gnome.org/) commencé en 1999, a pour objectif de construire un bureau «virtuel» comparable aux bureaux de Lotus ou de Microsoft (Microsoft Office). Aujourd'hui existent un système d'interface graphique, une suite bureautique et des outils pour développer ces logiciels.

De même qu'il est difficile d'avoir des chiffres précis des parts de marché des logiciels libres, il est assez difficile de savoir le nombre d'individus qui contribuent à la production d'un logiciel libre (essentiellement à cause de très grand nombre de projets). Le projet Orbiten [2000] a aussi essayé d'évaluer ce nombre. Bien que cette évaluation ne soit pas exempte de toute critique (dans les projets de la FSF, c'est souvent le nom de cette organisation qui apparaît, plutôt que le nom des contributeurs, leur nombre total est, de ce fait très sous-évalué), cela donne un ordre de grandeur de la taille de l'«organisation libre».

[16] L'entreprise de conseil Jipo (http ://www.jipo.com/) a réalisé, pour le compte de l'OTV (Observatoire des Télécommunications dans la Ville, http ://www.telecomville.org) une étude sur les offres existantes en logiciel libre, étude disponible à l'adresse : http ://www.telecomville.org/obs/inf716.html .

[17] À titre de comparaison, la taille de Windows ME et de la suite Works est inférieure à 500 Méga-octets.

Figure 5.9 — Évolution de la taille du noyau Linux.



source : Tuomi [2001].
La ligne pleine correspond à la version du noyau en développement et les lignes pointillées aux versions «stables», c'est-à-dire aux versions pour lesquelles les choix techniques sont figés.

Plus de 12 000 contributeurs différents sont recensés, ce qui est une force de développement considérable : Microsoft, qui consacre moins de 40 % de son chiffre d'affaires à la production de logiciels (Dréan [1996], p. 369) est une entreprise de 30 000 personnes.

La production du logiciel libre est donc dynamique, importante en volume et variée en contenu. Mais ce n'est pas seulement un succès du côté de la production. Dans certains domaines, la part de marché du logiciel libre est très importante.

La diffusion des logiciels libres.

Les succès libres.

Les succès les plus visibles des logiciels libres se trouvent dans les applications développées autour d'Internet.

Le logiciel Sendmail achemine environ 80 % du trafic mail (Lerner & Tirole [2000])[18], le logiciel serveur Apache fait fonctionner plus de 60 % des sites Web mondiaux et 30% des 30 plus gros serveurs du Web[19]. En avril 1999, les systèmes d'exploitation libres faisaient fonctionner près de 45 % des serveurs Internet[20].

D'après IDC, la part de marché de Linux comme système d'exploitation pour serveur est passé de 6,8 % en 1997 à 17,2 % en 1998 (+ 212 %), à comparer à celle des systèmes Unix classiques (17,4 % ; + 4%) et à celle de NT (36 % ; + 80 %) ; Aujourd'hui, GNU/Linux équipe environ 4 % des machines utilisées dans le monde (à rapporter aux 5 % des systèmes d'Apple). Pour IDG, GNU/Linux devrait passer, au Japon, d'environ 4 % du marché des serveurs, à 12 % en 2000. C'est le système d'exploitation qui progresse le plus vite[21]. Certains s'inquiètent déjà du «retard» français, car GNU/Linux n'est installé, en France, que sur 20 % des serveurs[22].

Il est aussi remarquable que, à part Java et Visual Basic, les langages de programmation

[18] Ce qui ne veut pas dire que Sendmail possède 80 % de ce marché (un même mail peut passer par plusieurs serveurs) mais que si on retirait tous les serveurs utilisant Sendmail, 80 % des mails ne seraient plus acheminés.

[19] Apache est un logiciel de serveur HTTP, qui permet d'envoyer les pages webs demandées par les navigateurs des postes clients.

Voir http ://www.netcraft.com/survey/ sur la part de marché d'Apache et voir http ://www.bsdtoday.com/2000/July/Features239.html sur sa part de marché dans les gros serveurs.

[20] Soient les serveurs FTP, WWW et les serveurs de news. Source : http ://www.leb.net/hzo/ioscount/data/r.9904.txt ; on retrouve dans les gros serveurs, la même tendance que pour Apache : la préférence pour les solutions propriétaires, puisque Solaris et Windows NT / 2000 sont les solutions les plus populaires.

[21] http ://www.theregister.co.uk/content/1/12065.html pour les chiffres du Japon.

«According to the latest release of IDC's Worldwide Quaterly Server Tracker, Linux server shipments increased 166 % to 72,422 units in Q499 from Q498, representing the fastest-growing operating environment in the server market» (CNET letter, 10/04/00).

«Through 2003, total Linux commercial shipments will grow faster than the total shipments of all other International Data Corporation (IDC) covered client or server operating environments. IDC estimates Linux commercial shipments will increase at a compound annual growth rate (CAGR) of 25 % from 1999 through 2003, compared with a 10 % CAGR for all other client operating environments combined and a 12 % CAGR for all other server operating environments combined» (IDC, http ://www.idc.com/Data/Software/content/SW033199PR.htm).

[22] Informatique magazine, 7 mai 1999, p. 80.

les plus utilisés soient tous des langages «libres» (soit le compilateur, soit l'interpréteur[23] est libre) : Ada 95 (compilateur GNAT), CAML sont, depuis leur création, des logiciels libres et il existe des compilateurs libres pour C/C++ (compilateur GCC), LISP, COBOL et Fortran. Le langage qui a le plus de succès pour créer des sites Web dynamiques est PHP, qui est aussi un logiciel libre, comme le sont Python ou Perl[24], d'autres langages utilisés (entre autre) dans les serveurs Web.

Nous ne pouvons que constater que le phénomène est important, qu'il semble promis à le devenir encore plus. Plus remarquable, sans doute, est le fait que c'est un phénomène de plus en plus «marchand», en ce sens que de plus en plus d'entreprises s'y investissent, et de façon importante.

L'implication grandissante des entreprises.

La «marchandisation» du Libre revêt deux aspects : d'une part, des producteurs traditionnels de logiciels s'investissent dans la production de logiciels libres, d'autre part, des entreprises sont créées à partir des logiciels libres développés.

IBM a, par exemple, annoncé un investissement de 1 milliard de dollars en 2001 pour développer les offres basées sur GNU/Linux et une partie de cet investissement sera consacrée à l'adaptation de ce logiciel aux machines et aux logiciels de cette entreprise[25]. HP-Compaq a annoncé que GNU/Linux serait un des trois systèmes qu'elle proposera avec ses machines (avec Windows et un système propre). SUN est le deuxième producteur de logiciels libres, après la Free Software Fundation, l'organisme fondé par Richard Stallman pour produire et coordonner la production libre.

Pour la plupart des logiciels libres, il existe aujourd'hui au moins une entreprise qui participe à sa production et qui vend des services en s'appuyant principalement sur ce logiciel (voir le tableau).

Loin de s'essouffler, il semble bien que le Libre se diffuse, intéresse de plus en plus de producteurs et contribue à créer de l'activité marchande.

[23] Logiciel qui traduit le code source d'un programme en langage machine au moment où ce programme est exécuté.

[24] On estime la population de développeurs utilisant Perl à environ 1 million (Lerner & Tirole [2000].)

[25] http ://news.cnet.com/news/0-1003-200-4111945.html .

Tableau 5.1 — Logiciels libres et entreprises dédiées.

Nom du logiciel	**Type de licence**	**Créateur**	**Année de création**	**Nom de l'entreprise**
Apache	BSD	Groupe de Webmaîtres utilisant le logiciel du NCSA	1995 (version stable début 1996)	Covalent technologies inc. (plusieurs membres de la fondation Apache travaillent dans cette entreprise.
Sendmail		Eric Allman	1970	Sendmail inc. (fondée par Allman)
Linux	GPL	Linus Torvalds	1991 (version stable en 1994)	Beaucoup d'entreprises (RedHat, Mandrake, Alcôve, etc.)
Gnome	GPL	Miguel de Icaza	1998	Helixcode (fondé par de Icaza)
Perl	Perl	Lary Wall	1987	
PHP				Zend
TCL		John Ousterhout	vers 1990	Scriptics Corporation (fondée par J. Ousterhout)
GCC	GPL	FSF		Cygnus (rachetée par RedHat)
MySQL		MySQL AB		MySQL AB
Zope	GPL	Digital Creation		Digital Creation
GNAT (Ada 95)	proche de la GPL	Groupe de chercheurs à l'université de New York, financés par le DoD	1995	Ada Core Technology (fondée par des membres du groupe d'universitaires qui ont développé le compilateur.et ACT Europe

Annexe B.

Le détail des réponses au questionnaire.

Le questionnaire se trouve à l'adresse :

http ://www-eco.enst-bretagne.fr/cgi-bin/nj.pl

Le détail des réponses.

	Code APE	Service	Fonction	Âge de l'entre–prise.	Activité	Autre activité	Clientèle principale	Autre clientèle principale
1	722Z	Direction Générale	P.D.G., DG, gérant	Moins d'un an	logiciel		P.M.E.	
2	722Z	Direction Générale	P.D.G. , Dg , Gérant	Moins de 10 ans	logiciel		Grands comptes	
3	516G	Marketing , Commercial	Directeur	Moins de 5 ans	service info		Grands comptes	
4	722Z	Marketing, commercial	Directeur	Moins d'un an	logiciel		Grands comptes	
5	72 2Z	Direction Technique	Directeur	Moins de 5 ans	service info		Autre	secteur IT
6	722Z	Direction Générale	P.D.G., DG, gérant	Moins de 3 ans	logiciel		Grands comptes	
7	???	Direction Technique	Directeur	Moins de 5 ans	service info		P.M.E.	
8	????	R&D	Ingénieur, chercheur, cadre	Plus de 10 ans	logiciel		Grands comptes	
9	721Z	Direction Générale	P.D.G., DG, gérant	Moins de 3 ans	service info	dév. logiciels spécifiques	P.M.E.	
10	???	Production Exploitation	Ingénieur, chercheur, cadre	Moins de 5 ans	Autre	Web agency (dvt de sites web + base de données, programmation avancée, jeux en ligne)	P.M.E.	
11	???	Production Exploitation	Ingénieur, chercheur, cadre	Moins de 10 ans	service info		Grands comptes	
12	725Z	Direction Technique	Ingénieur, chercheur, cadre	Plus de 10 ans	logiciel		Grands comptes	
13	722z	Installation	Ingénieur, chercheur, cadre	Moins de 3 ans	service info		P.M.E.	
14	516G	R&D	Chef de Service	Moins de 10 ans	service info		P.M.E.	
15	516J	R&D	Ingénieur , Chercheur , Cadre	Plus de 10 ans	Autre	Société de services en télécoms par satellite	P.M.E.	
16	???	Marketing, commercial	Ingénieur, chercheur, cadre	Plus de 10 ans	constructeur		Grands comptes	
17	722z	Direction Générale	P.D.G., DG, gérant	Moins de 10 ans	Autre	web agency, hébergement	P.M.E.	
18	742C	R&D	Chef de Service	Moins d'un an	Autre		Autre	

	Code APE	Service	Fonction	Âge de l'entre–prise.	Activité	Autre activité	Clientèle principale	Autre clientèle principale
19	????	Direction Technique	Ingénieur, chercheur, cadre	Plus de 10 ans	conseil		Grands comptes	
20	722Z	Direction Générale	P.D.G., DG, gérant	Moins de 10 ans	Logiciel	grossiste	P.M.E.	
21	742C	Direction Générale	P.D.G., DG, gérant	Moins de 10 ans	service info		Grands comptes	
22	622Z	Production Exploitation	Ingénieur, chercheur, cadre	Moins de 5 ans	conseil		T.P.E. et indépendants	
23	dev. logiciel	R&D	P.D.G., DG, gérant	Moins d'un an	service info		Grands comptes	
24	722Z	R&D	P.D.G., DG, gérant	Moins de 3 ans	logiciel		Grands comptes	
25	524Z	Installation	P.D.G., DG, gérant	Moins de 3 ans	service info		Grands comptes	
26	721Z	Production Exploitation	Directeur	Moins de 10 ans	Autre	Formation	Grands comptes	
27	721Z	Autre	P.D.G., DG, gérant	Moins de 5 ans	conseil		Grands comptes	
28	725Z	Direction Générale	P.D.G., DG, gérant	Moins de 5 ans	service info		P.M.E.	
29	722z	Direction Générale	P.D.G., DG, gérant	Plus de 10 ans	logiciel	distbuteur de logiciel de base / prestataire de migrations	P.M.E.	
30	721Z	Direction Générale	P.D.G.	Moins de 5 ans	service info		Grands comptes	
31	333Z	R&D	Ingénieur, chercheur, cadre	Plus de 10 ans	Autre	système d'informatique industrielle	Grands comptes	
32	516G	Direction Générale	P.D.G., DG, gérant	Moins de 3 ans	service info		Autre	3 tiers
33	722Z	Direction Technique	Directeur	Moins d'un an	logiciel		Grands comptes	
34	722Z	Direction Générale	P.D.G., DG, gérant	Moins de 10 ans	service info		P.M.E.	
35	723Z	Direction Générale	P.D.G. , Dg , Gérant	Moins de 10 ans	service info		Grands comptes	
36	???	Direction Générale	P.D.G. , Dg , Gérant	Moins d'un an	logiciel		Grands comptes	

	Nombre salariés début 2001	Nombre salariés fin 2001	CA en 2000 (en ME)	Croissance du CA (2001)	Bénéf. 2000 (en ME)	Bénéf. 2001 (en ME)	Utilisation Linux en interne	Utilisation commerciale Linux	Distrib. Commerciales utilisées
1	5- 10	11- 25		26 à 50			Une seule distribution	base de l'offre	Une seule distribution
2	11- 25	11- 25	0		0	0	Une seule distribution	base de l'offre	Une seule distribution
3	26- 50	51- 100	34	26 à 50	0	0	Plusieurs distributions	base de l'offre	Plusieurs distributions
4	11- 25	26- 50	0,15	Plus de 50	N/A	N/A	Plusieurs distributions	base de l'offre	Plusieurs distributions
5	51- 100	101- 200	NC		NC	NC	Plusieurs distributions	base de l'offre	Plusieurs distributions
6	11- 25						Plusieurs distributions	base de l'offre	Plusieurs distributions
7	11- 25	26- 50	3	11 à 25			Une seule distribution	base de l'offre	Une seule distribution
8	51- 100	51- 100	15	26 à 50	2	4	Une seule distribution	Tps_en_tps	Une seule distribution
9	2- 4	5- 10	0,21	26 à 50	0	0	Une seule distribution	base de l'offre	Une seule distribution
10	11- 25	26- 50	???	???	???	???	Une seule distribution	base de l'offre	Une seule distribution
11	11- 25	11- 25	1	?	?	?	Plusieurs distributions	base de l'offre	Plusieurs distributions
12	5- 10	5- 10	0,4	11 à 25	*	*	Une seule distribution	Tps_en_tps	Une seule distribution
13	5- 10	11- 25	?	?	?	?	PasLinux	Jamais	PasLinux
14	5- 10	5- 10	1,5	4 à 10	0,15	0,3	Plusieurs distributions	base de l'offre	Plusieurs distributions
15	51- 100	51- 100	50	5 à 10	0	0	Une seule distribution	base de l'offre	Une seule distribution
16	+501	+501	???	???	???	???	Plusieurs distributions	Tps_en_tps	Plusieurs distributions
17	2- 4	2- 4		11à 25			Une seule distribution	Souvent	Une seule distribution
18	1	1	0,5	26 à 50	n/a	1,5	PasLinux	Nsp	PasLinux

	Nombre salariés début 2001	Nombre salariés fin 2001	CA en 2000 (en ME)	Croissance du CA (2001)	Bénéf. 2000 (en ME)	Bénéf. 2001 (en ME)	Utilisation Linux en interne	Utilisation commerciale Linux	Distrib. Commerciales utilisées
19	101- 200	201- 500	11	26 à 50	1		Une seule distribution	base de l'offre	Une seule distribution
20	2- 4	2- 4					Plusieurs distributions	base de l'offre	Plusieurs distributions
21	1	1	0,1	11 à 25	0	0	Une seule distribution	base de l'offre	Une seule distribution
22	2- 4	1			0	0	Plusieurs distributions	base de l'offre	Une seule distribution
23	1	5- 10	0	0	0	0	Une seule distribution	base de l'offre	Une seule distribution
24	1	2- 4	0,02		0		Une seule distribution	base de l'offre	Une seule distribution
25	1	2- 4	*	*	*	*	Plusieurs distributions	base de l'offre	Plusieurs distributions
26	5- 10	11- 25					Plusieurs distributions	base de l'offre	Une seule distribution
27	1	1	0,1	0	0,05	0,05	Plusieurs distributions	Souvent	Plusieurs distributions
28	1	2- 4					Une seule distribution	base de l'offre	Une seule distribution
29	11- 25	11- 25	1,5	11 à 25	0,07	0,1	Plusieurs distributions	base de l'offre	Plusieurs distributions
30	2- 4	2- 4	0,2	26 à 50	0,02	0,04	Plusieurs distributions	base de l'offre	Plusieurs distributions
31	51- 100	51- 100			7,6	8	Plusieurs distributions	Tps_en_tps	Une seule distribution
32	11- 25	11- 25	0,5	Plus de 50	0,06	1,25	Plusieurs distributions	base de l'offre	Plusieurs distributions
33	51- 100	51- 100	16	5 à 10			Plusieurs distributions	base de l'offre	Plusieurs distributions
34	5- 10	5- 10	0,3	11 à 25	0	0	Plusieurs distributions	base de l'offre	Une seule distribution
35	2- 4		0		0	0			Une seule distribution
36	11- 25	26- 50	0	Plus de 50	0	0			Plusieurs distributions

	Utilisation de Perl	Utilisation de Python	Utilisation de GCC	Utilisation de TCL	Utilisation de Java	Utilisation de PHP	Utilisation d'Ada 95
1	pas utilisé	pas utilisé	standard	pas utilisé	standard	standard	pas utilisé
2	standard	Nsp	standard	Nsp	Nsp	Nsp	standard
3	pas utilisé	standard	standard	pas utilisé	autorisé	utilisé	pas utilisé
4	standard	utilisé	utilisé	inconnu	standard	pas utilisé	pas utilisé
5	standard	standard	standard	autorisé	autorisé	standard	Nsp
6	recomandé	pas utilisé	standard	Nsp	pas utilisé	standard	Nsp
7	standard	pas utilisé	standard	pas utilisé	autorisé	standard	pas utilisé
8	autorisé	recomandé	recomandé	pas utilisé	autorisé	autorisé	pas utilisé
9	standard	Nsp	standard	Nsp	standard	standard	inconnu
10	standard	recomandé	standard	pas utilisé	recomandé	standard	pas utilisé
11	standard	autorisé	standard	recomandé	recomandé	standard	inconnu
12	utilisé	pas utilisé	autorisé	pas utilisé	autorisé	standard	pas utilisé
13	pas utilisé	pas utilisé	standard	pas utilisé	pas utilisé	standard	pas utilisé
14	utilisé	pas utilisé	standard	Nsp	standard	standard	pas utilisé
15	recomandé	pas utilisé	recomandé	pas utilisé	pas utilisé	standard	pas utilisé
16	autorisé	Nsp	autorisé	Nsp	recomandé	Nsp	Nsp
17	standard	pas utilisé	standard	pas utilisé	pas utilisé	standard	pas utilisé
18	Nsp	Nsp	Nsp	Nsp	Nsp	Nsp	Nsp

	Utilisation de Perl	Utilisation de Python	Utilisation de GCC	Utilisation de TCL	Utilisation de Java	Utilisation de PHP	Utilisation d'Ada 95
19	recomandé	Nsp	standard	Nsp	standard	standard	Nsp
20	standard	recomandé	standard	recomandé	autorisé	recomandé	autorisé
21	standard	autorisé	standard	standard	autorisé	autorisé	autorisé
22	autorisé	pas utilisé	standard	Nsp	Nsp	standard	Nsp
23	pas utilisé	standard	recomandé	pas utilisé	pas utilisé	pas utilisé	pas utilisé
24	standard	pas utilisé	standard	utilisé	utilisé	standard	pas utilisé
25	standard	autorisé	standard	autorisé	autorisé	standard	pas utilisé
26	autorisé	autorisé	standard	recomandé	standard	utilisé	pas utilisé
27	recomandé	pas utilisé	standard	pas utilisé	standard	pas utilisé	pas utilisé
28	pas utilisé	standard	standard	pas utilisé	autorisé	standard	pas utilisé
29	utilisé	Nsp	standard	Nsp	standard	utilisé	Nsp
30	recomandé	pas utilisé	autorisé	inconnu	standard	autorisé	pas utilisé
31	pas utilisé	pas utilisé	utilisé	utilisé	utilisé	pas utilisé	pas utilisé
32	standard	standard	standard	standard	recomandé	recomandé	pas utilisé
33	recomandé	recomandé	standard	pas utilisé	standard	standard	pas utilisé
34	standard	standard	standard	pas utilisé	standard	standard	pas utilisé
35	standard	inconnu	Nsp	Nsp	standard	standard	Nsp
36	standard	standard	standard	pas utilisé	recomandé	standard	pas utilisé

	Utilisation commerciale d'Apache	Sendmail	Unix de type BSD	Samba	Bind	Zope	Autres logiciels libres utilisés.
1	base de l'offre	Souvent	Nsp	base de l'offre	inconnu	inconnu	Postgres
2	base de l'offre	base de l'offre	Nsp	base de l'offre	Nsp	Nsp	
3	base de l'offre	Jamais	Tps_en_tps	base de l'offre	base de l'offre	Souvent	Postfix
4	base de l'offre	Jamais	base de l'offre	base de l'offre	Jamais	Jamais	Qmail
5	base de l'offre	Tps_en_tps	Souvent	Souvent	base de l'offre	Souvent	IMP, Sympa, Squid, PostgreSQL, Perl, Openldap
6	base de l'offre	Souvent	Tps_en_tps	Nsp	Nsp	Nsp	
7	base de l'offre	Tps_en_tps	Jamais	Tps_en_tps	Tps_en_tps	Jamais	MySQL
8	base de l'offre	Souvent	Jamais	Souvent	Jamais	Jamais	Tomcat
9	base de l'offre	base de l'offre	Tps_en_tps	Souvent	Tps_en_tps	inconnu	
10	base de l'offre	Jamais	Jamais	Souvent	Jamais	Jamais	Perl
11	base de l'offre	Souvent	Souvent	base de l'offre	base de l'offre	Jamais	Qmail, openldap, openssl, mod_ssl, courier-imap, btt, proftpd
12	base de l'offre	Tps_en_tps	Jamais	Tps_en_tps	Tps_en_tps	Jamais	MySQL, PostgresSQL
13	base de l'offre	Jamais	base de l'offre	Souvent	Tps_en_tps	inconnu	Postfix, netatalk (serveur Mac)
14	Souvent	Souvent	Tps_en_tps	base de l'offre	Souvent	Tps_en_tps	Postgresql, Soffice/openoffice, squid, openldap
15	base de l'offre	base de l'offre	Jamais	Tps_en_tps	base de l'offre	Jamais	
16	Tps_en_tps	Souvent	Nsp	Tps_en_tps	Souvent	Nsp	
17	base de l'offre	Souvent	Souvent	Tps_en_tps	Jamais	Jamais	Qmail, mrtg, netsaint
18	Nsp	Nsp	Nsp	Nsp	Nsp	Nsp	

	Utilisation commerciale d'Apache	Sendmail	Unix de type BSD	Samba	Bind	Zope	Autres logiciels libres utilisés.
19	base de l'offre	base de l'offre	Nsp	base de l'offre	base de l'offre	Nsp	Webmin Jetspeed BB MRTG
20	base de l'offre	base de l'offre	Tps_en_tps	base de l'offre	Souvent	Nsp	
21	Souvent	Jamais	Tps_en_tps	Souvent	Jamais	Jamais	Qmail, tinydns, hypermail
22	base de l'offre	base de l'offre	Nsp	base de l'offre	base de l'offre	Nsp	NETATALK
23	base de l'offre	Jamais	Jamais	Jamais	base de l'offre	base de l'offre	
24	base de l'offre	Tps_en_tps	Jamais	Souvent	base de l'offre	Jamais	Postfix, GNUstep/GNUstepWeb, Postgresql, MySQL
25	base de l'offre	base de l'offre	Nsp	base de l'offre	base de l'offre	Jamais	Pvm, mpich, lam, postgres, hylafax
26	base de l'offre	base de l'offre	Jamais	base de l'offre	base de l'offre	Jamais	gimp blender gcc openoffice lynx ...
27	base de l'offre	base de l'offre	Jamais	Souvent	base de l'offre	Jamais	Apache Jserv, Tomcat (normal je suis core developper)
28	base de l'offre	Souvent	Jamais	Souvent	Souvent	Jamais	PHP, Python, fetchmail, MySQL, ...
29	base de l'offre	Souvent	Jamais	base de l'offre	Souvent	Jamais	Sgbd, compilateur, interfaces ODBC
30	base de l'offre	base de l'offre	Jamais	base de l'offre	base de l'offre	Jamais	
31	Jamais	Jamais	Jamais	Tps_en_tps	Jamais	Jamais	
32	base de l'offre	base de l'offre	base de l'offre	base de l'offre	base de l'offre	base de l'offre	Hylafax
33	base de l'offre	Souvent	Nsp	Nsp	Nsp	Souvent	
34	base de l'offre	base de l'offre	Jamais	Tps_en_tps	Souvent	Tps_en_tps	plein
35	base de l'offre	base de l'offre	Nsp	base de l'offre	Nsp	inconnu	MySQL
36	base de l'offre	Jamais	Jamais	base de l'offre	base de l'offre	base de l'offre	

	Serveur de fichier	Serveur Internet intranet	Serveur bases de données	Poste client	LL : vendre du support	LL : vendre service de substitu°	LL : vendre du logiciel	LL : vendre matériel	LL : vendre autre chose
1	Souvent	Souvent	Souvent	Tps_en_tps	Oui		Oui		
2	Souvent	Souvent	Souvent	Souvent	Oui		Oui		
3	Souvent	Souvent	Tps_en_tps	Tps_en_tps	Oui	Oui		Oui	
4	Souvent	Souvent	Souvent	Souvent			Oui		
5	Souvent	Souvent	Souvent	Souvent	Oui	Oui			
6	Tps_en_tps	Souvent	Souvent	Souvent	Oui		Oui		
7	Tps_en_tps	Souvent	Souvent	Tps_en_tps	Oui	Oui			
8	Souvent	Souvent	Souvent	Tps_en_tps			Oui		
9	Souvent	Tps_en_tps	Tps_en_tps	Jamais	Oui	Oui	Oui		
10	Souvent	Souvent	Souvent	Tps_en_tps	Oui	Oui			
11	Souvent	Souvent	Souvent	Tps_en_tps	Oui	Oui		Oui	connexions internet
12	Tps_en_tps	Souvent	Souvent	Jamais	Oui	Oui			
13	Souvent	Souvent	Souvent	Jamais		Oui			installation & paramétrages
14	Souvent	Souvent	Souvent	Souvent	Oui	Oui			
15	Souvent	Souvent	Tps_en_tps	Jamais				Oui	
16	Souvent	Souvent	Souvent	Souvent	Oui			Oui	
17	Souvent	Souvent	Souvent	Tps_en_tps					les briques de nos outils de travail
18	Nsp	Nsp	Nsp	Nsp	Oui	Oui		Oui	

	Serveur de fichier	Serveur Internet intranet	Serveur bases de données	Poste client	LL : vendre du support	LL : vendre service de substitu°	LL : vendre du logiciel	LL : vendre matériel	LL : vendre autre chose
19	Souvent	Souvent	Tps_en_tps	Jamais	Oui		Oui		
20	Nsp	Nsp	Nsp	Nsp			Oui		
21	Souvent	Souvent	Tps_en_tps	Souvent	Oui	Oui	Oui		
22	Souvent	Souvent	Souvent	Tps_en_tps	Oui				
23	Jamais	Souvent	Jamais	Jamais		Oui			
24	Souvent	Souvent	Souvent	Souvent	Oui	Oui			
25	Souvent	Souvent	Souvent	Tps_en_tps	Oui			Oui	formation, en
26	Souvent	Souvent	Souvent	Souvent					utilisation interne. Tous nos postes sous Linux.
27	Souvent	Souvent	Tps_en_tps	Tps_en_tps	Oui				
28	Souvent	Souvent	Souvent	Tps_en_tps	Oui	Oui	Oui		
29	Souvent	Souvent	Souvent	Tps_en_tps	Oui		Oui	Oui	
30	Souvent	Souvent	Souvent	Jamais	Oui	Oui	Oui		
31	Tps_en_tps	Tps_en_tps	Tps_en_tps	Tps_en_tps	Oui		Oui	Oui	
32	Souvent	Souvent	Souvent	Jamais	Oui	Oui	Oui		
33	Souvent	Souvent	Souvent	Souvent	Oui		Oui		
34	Souvent	Souvent	Souvent	Tps_en_tps	Oui	Oui	Oui		
35	Souvent	Souvent	Souvent	Jamais	Oui	Oui			moins cher et plus efficace !
36	Souvent	Souvent	Souvent	Souvent	Oui	Oui	Oui	Oui	

	Tps laissé libre aux dvpeurs.	Critères mise en libre logiciel	LL : rendre un meilleur service.	LL : oblige à un meilleur service	LL : dvper sol. tech. innovantes	LL adapté dvt logiciels standards
1	entre 10 et 30%	Libre, fiable, support	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
2	entre 10 et 30%		Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Nsp
3	entre 10 et 30%	Evoluti−vité, Support, Marché, Notoriété	Plutôt d'accord	Tout à fait d'accord	Plutôt d'accord	Plutôt pas d'accord
4	entre 10 et 30%	qualité du logiciel, capacité à gérer son développement par la communauté	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
5	entre 10 et 30%		Plutôt d'accord	Tout à fait d'accord	Plutôt d'accord	Plutôt pas d'accord
6	entre 10 et 30%		Plutôt pas d'accord	Plutôt d'accord	Plutôt d'accord	Plutôt pas d'accord
7	moins de 5%		Plutôt pas d'accord	Plutôt d'accord	Plutôt d'accord	Tout à fait d'accord
8	Il n'y en a pas		Tout à fait d'accord	Plutôt d'accord	Tout à fait d'accord	Plutôt d'accord
9	Il n'y en a pas		Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord
10	Il n'y en a pas	Volonté obtenir logiciels de très bonne qualité + constante amélioration/évolution. Partage connaissances. Émulation entre entreprises vs concurrence bloquante.	Tout à fait d'accord	Plutôt d'accord	Tout à fait d'accord	Plutôt pas d'accord
11	Il n'y en a pas		Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Tout à fait d'accord
12	entre 5 et 10%		Tout à fait d'accord	Plutôt pas d'accord	Nsp	Plutôt d'accord
13	moins de 5%		Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Nsp
14	entre 10 et 30%	Couts, capacité d'adaptation, qualité standard	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
15	Nsp		Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord
16	plus de 30%	les copyright portant sur le soft original	Tout à fait d'accord	Plutôt d'accord	Tout à fait d'accord	Plutôt pas d'accord
17	entre 10 et 30%	l'utilité maximal	Plutôt d'accord	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord
18	Nsp	développeurs (pour de développement collaboratif) et d'utilisateurs (pour l'utilisation).	Nsp	Nsp	Nsp	Nsp

	Tps laissé libre aux dvpeurs.	Critères mise en libre logiciel	un meilleur service.	LL : oblige à un meilleur service	LL : dvper sol. tech. innovantes	dvt logiciels standards
19	Nsp		Plutôt pas d'accord	Plutôt pas d'accord	Tout à fait d'accord	Plutôt d'accord
20	Il n'y en a pas		Nsp	Nsp	Plutôt pas d'accord	Plutôt d'accord
21	entre 10 et 30%		Nsp	Plutôt d'accord	Plutôt d'accord	Pas d'accord du tout
22	Il n'y en a pas	Stabilité, pertinence	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
23	plus de 30%	avantage ou non aux concur−rents,	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord
24	entre 10 et 30%	avantages offerts par une adoption du logiciel par d'autres personnes/sociétés, retour sur investissement	Plutôt d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord
25	Il n'y en a pas	taille du marché, psychologie des clients vis−à−vis du modèle économique	Plutôt d'accord	Pas d'accord du tout	Tout à fait d'accord	Plutôt d'accord
26	entre 5 et 10%		Nsp	Tout à fait d'accord	Plutôt d'accord	Plutôt pas d'accord
27	entre 5 et 10%	type de licence	Nsp	Plutôt d'accord	Plutôt d'accord	Plutôt pas d'accord
28	Il n'y en a pas	consolidation du produit (déboggage en phase de test et stabilité en phase d'exploitation)	Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Plutôt d'accord
29	moins de 5%	Fiabilité, références, ouverture, outils standard	Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Pas d'accord du tout
30	Nsp	assez grande le fait qu'il soit libre ne	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord
31	moins de 5%	représente pas un manque de rapport.	Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Plutôt pas d'accord
32	entre 10 et 30%	développement communautaire plus rapide et moins cher	Plutôt d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord
33	moins de 5%	survie economique de la société	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord
34	Il n'y en a pas	intérêt pour la communauté, développement collaboratif possible.	Plutôt pas d'accord	Plutôt pas d'accord	Plutôt d'accord	Plutôt d'accord
35	entre 10 et 30%	qualité, sécurité, facilité d'utilisation et d'installation	Plutôt d'accord	Plutôt d'accord	Tout à fait d'accord	Tout à fait d'accord
36	entre 10 et 30%	utilité	Tout à fait d'accord	Plutôt pas d'accord	Tout à fait d'accord	Tout à fait d'accord

	LL n'évolue plus : peu d'op–portunités.	LL : de–mandé par client	Potentiel commerc. LL ds embarqué	Potentil instal. logiciels	Forma–tion	Vente logiciel s libres	Type Collab	Support	Integra–tion de logiciels
1	Nsp	Tout à fait d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Pas de potentiel	Fort potentiel	Fort potentiel
2	Tout à fait d'accord	Nsp	Fort potentiel	Nsp	Fort potentiel	Nsp	Faible potentiel	Fort potentiel	Fort potentiel
3	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Faible potentiel	Nsp	Pas de potentiel	Faible potentiel	Fort potentiel	Fort potentiel
4	Tout à fait d'accord	Plutôt d'accord	Fort potentiel	Faible potentiel	Fort potentiel	Nsp	Pas de potentiel	Fort potentiel	Fort potentiel
5	Pas d'accord du tout	Plutôt d'accord	Fort potentiel	Faible potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
6	Nsp	Plutôt pas d'accord	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Faible potentiel	Fort potentiel	Fort potentiel
7	Plutôt pas d'accord	Pas d'accord du tout	Fort potentiel	Faible potentiel	Faible potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
8	Nsp	Plutôt pas d'accord	Nsp	Pas de potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
9	Plutôt d'accord	Pas d'accord du tout	Fort potentiel	Fort potentiel	Faible potentiel	Pas de potentiel	Fort potentiel	Pas de potentiel	Fort potentiel
10	Plutôt pas d'accord	Plutôt d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
11	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Faible potentiel	Fort potentiel	Faible potentiel	Fort potentiel	Fort potentiel	Fort potentiel
12	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Nsp	Pas de potentiel	Fort potentiel	Faible potentiel	Fort potentiel	Fort potentiel
13	Pas d'accord du tout	Plutôt pas d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Pas de potentiel	Nsp	Faible potentiel	Fort potentiel
14	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
15	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
16	Pas d'accord du tout	Plutôt d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Pas de potentiel	Faible potentiel	Fort potentiel	Fort potentiel
17	Nsp	Nsp	Fort potentiel	Fort potentiel	Fort potentiel	Pas de potentiel	Nsp	Faible potentiel	Fort potentiel
18	Nsp	Nsp	Nsp	Nsp	Nsp	Nsp	Nsp	Nsp	Nsp

	LL n'évolue plus : peu d'opportunités.	LL : demandé par client	Potentiel commerc. LL ds embarqué	Potentil instal. logiciels	Formation	Vente logiciels libres	Type Collab	Support	Integration de logiciels
19	Plutôt pas d'accord	Tout à fait d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Nsp	Fort potentiel	Faible potentiel
20	Plutôt d'accord	Plutôt d'accord	Fort potentiel	Nsp	Fort potentiel	Pas de potentiel	Nsp	Fort potentiel	Fort potentiel
21	Plutôt pas d'accord	Tout à fait d'accord	Nsp	Nsp	Fort potentiel	Nsp	Nsp	Nsp	Nsp
22	Plutôt pas d'accord	Plutôt d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Pas de potentiel	Faible potentiel	Fort potentiel	Fort potentiel
23	Plutôt pas d'accord	Tout à fait d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
24	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Fort potentiel	Fort potentiel	Fort potentiel
25	Pas d'accord du tout	Tout à fait d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Pas de potentiel	Pas de potentiel	Faible potentiel	Fort potentiel
26	Pas d'accord du tout	Plutôt d'accord	Fort potentiel	Pas de potentiel	Fort potentiel	Faible potentiel	Nsp	Faible potentiel	Fort potentiel
27	Plutôt pas d'accord	Plutôt pas d'accord	Faible potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Fort potentiel	Fort potentiel	Fort potentiel
28	Pas d'accord du tout	Pas d'accord du tout	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel
29	Plutôt pas d'accord	Plutôt d'accord	Faible potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel
30	Plutôt pas d'accord	Pas d'accord du tout	Nsp	Fort potentiel	Fort potentiel	Pas de potentiel	Faible potentiel	Fort potentiel	Fort potentiel
31	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Nsp	Fort potentiel	Fort potentiel
32	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Nsp	Fort potentiel	Fort potentiel
33	Plutôt pas d'accord	Plutôt pas d'accord	Fort potentiel	Pas de potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel
34	Plutôt d'accord	Pas d'accord du tout	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Fort potentiel	Faible potentiel	Fort potentiel
35	Plutôt d'accord	Plutôt d'accord	Fort potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Faible potentiel	Faible potentiel	Nsp
36	Plutôt d'accord	Tout à fait d'accord	Fort potentiel	Fort potentiel	Faible potentiel	Faible potentiel	Fort potentiel	Fort potentiel	Fort potentiel

	Vente LL & proprié-taire	Dvt à facon	Autres opportunites commerciales	Contribu-tion car c'est l'usage	Seul moyen suivre évolutions du logiciel	Pour faire valider et tester améliorations
1	Fort potentiel	Fort potentiel		Tout à fait d'accord	Plutôt d'accord	Plutôt pas d'accord
2	Nsp	Nsp		Tout à fait d'accord	Tout à fait d'accord	Nsp
3	Fort potentiel	Fort potentiel		Tout à fait d'accord	Plutôt d'accord	Plutôt d'accord
4	Fort potentiel	Fort potentiel		Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord
5	Faible potentiel	Fort potentiel		Plutôt d'accord	Plutôt pas d'accord	Plutôt d'accord
6	Fort potentiel	Fort potentiel		Plutôt pas d'accord	Tout à fait d'accord	Plutôt d'accord
7	Fort potentiel	Fort potentiel		Plutôt d'accord	Plutôt pas d'accord	Plutôt d'accord
8	Faible potentiel	Fort potentiel		Plutôt d'accord	Plutôt pas d'accord	Plutôt d'accord
9	Fort potentiel	Fort potentiel		Plutôt pas d'accord	Tout à fait d'accord	Plutôt d'accord
10	Fort potentiel	Fort potentiel		Nsp	Nsp	Nsp
11	Fort potentiel	Fort potentiel		Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
12	Fort potentiel	Fort potentiel		Tout à fait d'accord	Plutôt d'accord	Nsp
13	Faible potentiel	Fort potentiel		Nsp	Nsp	Nsp
14	Fort potentiel	Fort potentiel		Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
15	Faible potentiel	Fort potentiel		Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
16	Faible potentiel	Fort potentiel		Tout à fait d'accord	Plutôt d'accord	Plutôt d'accord
17	Fort potentiel	Fort potentiel		Tout à fait d'accord	Nsp	Nsp
18	Nsp	Nsp	SSII. Pas de limitations concernant les potentiels commerciaux (mais ne travaillons qu'avec du libre). Dvts essentiellement en GPL ; Tout logiciel trop spécialisé dans métiers niches	Nsp	Nsp	Nsp

	Vente LL & proprié-taire	Dvt à facon	Autres opportunites commerciales	Contribu-tion car c'est l'usage	Seul moyen suivre évolutions du logiciel	Pour faire valider et tester améliorations
19	Faible potentiel	Fort potentiel		Tout à fait d'accord	Plutôt pas d'accord	Plutôt d'accord
20	Fort potentiel	Fort potentiel		Plutôt d'accord	Nsp	Plutôt d'accord
21	Nsp	Nsp		Pas d'accord du tout	Plutôt d'accord	Tout à fait d'accord
22	Fort potentiel	Fort potentiel		Nsp	Nsp	Nsp
23	Faible potentiel	Faible potentiel		Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord
24	Fort potentiel	Fort potentiel		Tout à fait d'accord	Plutôt d'accord	Tout à fait d'accord
25	Fort potentiel	Fort potentiel		Pas d'accord du tout	Pas d'accord du tout	Plutôt d'accord
26	Nsp	Fort potentiel	R&D	Pas d'accord du tout	Plutôt d'accord	Plutôt d'accord
27	Fort potentiel	Fort potentiel		Plutôt d'accord	Tout à fait d'accord	Tout à fait d'accord
28	Fort potentiel	Fort potentiel		Nsp	Nsp	Nsp
29	Fort potentiel	Fort potentiel	Progiciel de Gestion en COBOL (voir email)	Plutôt pas d'accord	Plutôt d'accord	Tout à fait d'accord
30	Faible potentiel	Fort potentiel		Tout à fait d'accord	Plutôt pas d'accord	Plutôt pas d'accord
31	Fort potentiel	Fort potentiel		Plutôt pas d'accord	Plutôt d'accord	Plutôt pas d'accord
32	Fort potentiel	Fort potentiel		Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord
33	Fort potentiel	Fort potentiel		Plutôt d'accord	Plutôt d'accord	Tout à fait d'accord
34	Fort potentiel	Fort potentiel		Nsp	Nsp	Nsp
35	Fort potentiel	Fort potentiel		Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord
36	Fort potentiel	Fort potentiel		Pas d'accord du tout	Pas d'accord du tout	Plutôt d'accord

	Permet de se faire connaitre des développeurs	Permet de se faire connaitre des clients	Garantie de compétence	Demande des clients	Autres raisons
1	Plutôt d'accord	Plutôt d'accord	Tout à fait d'accord	Pas d'accord du tout	Aider la communauté
2	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Nsp	
3	Plutôt d'accord	Tout à fait d'accord	Tout à fait d'accord	Pas d'accord du tout	
4	Pas d'accord du tout	Pas d'accord du tout	Plutôt d'accord	Pas d'accord du tout	
5	Plutôt d'accord	Plutôt pas d'accord	Tout à fait d'accord	Plutôt d'accord	
6	Plutôt pas d'accord	Plutôt d'accord	Pas d'accord du tout	Pas d'accord du tout	
7	Plutôt pas d'accord	Plutôt pas d'accord	Plutôt d'accord	Plutôt pas d'accord	
8	Plutôt pas d'accord	Plutôt pas d'accord	Pas d'accord du tout	Pas d'accord du tout	
9	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord	Pas d'accord du tout	
10	Nsp	Nsp	Nsp	Nsp	Il n'y a pas officiellement de contribution à des logiciels libres. Par contre les développeurs y contribuent dans leur temps libre.
11	Plutôt d'accord	Tout à fait d'accord	Plutôt d'accord	Plutôt pas d'accord	
12	Pas d'accord du tout	Pas d'accord du tout	Plutôt d'accord	Nsp	
13	Nsp	Nsp	Nsp	Nsp	
14	Plutôt d'accord	Plutôt pas d'accord	Plutôt d'accord	Nsp	
15	Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Plutôt pas d'accord	
16	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord	C'est une évolution irréversible du développement logiciel
17	Nsp	Plutôt d'accord	Plutôt d'accord	Nsp	
18	Nsp	Nsp	Nsp	Nsp	

	faire connaitre des développeurs	Permet de se faire connaitre des clients	Garantie de compétence	Demande des clients	Autres raisons
19	Nsp	Plutôt pas d'accord	Plutôt pas d'accord	Tout à fait d'accord	
20	Plutôt d'accord	Plutôt d'accord	Nsp	Plutôt d'accord	
21	Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Plutôt pas d'accord	
22	Nsp	Nsp	Nsp	Nsp	
23	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	
24	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord	
25	Plutôt pas d'accord	Plutôt d'accord	Tout à fait d'accord	Tout à fait d'accord	
26	Plutôt d'accord	Plutôt d'accord	Plutôt d'accord	Pas d'accord du tout	c'est un gage de sécurité. c'est un levier de progrès
27	Tout à fait d'accord	Tout à fait d'accord	Tout à fait d'accord	Plutôt pas d'accord	
28	Nsp	Nsp	Nsp	Nsp	
29	Tout à fait d'accord	Tout à fait d'accord	Plutôt d'accord	Plutôt d'accord	
30	Plutôt pas d'accord	Tout à fait d'accord	Pas d'accord du tout	Nsp	
31	Plutôt pas d'accord	Plutôt d'accord	Plutôt d'accord	Plutôt pas d'accord	
32	Plutôt d'accord	Plutôt pas d'accord	Plutôt d'accord	Plutôt pas d'accord	
33	Plutôt d'accord	Plutôt pas d'accord	Plutôt pas d'accord	Pas d'accord du tout	
34	Nsp	Nsp	Nsp	Nsp	
35	Plutôt pas d'accord	Pas d'accord du tout	Plutôt d'accord	Pas d'accord du tout	le fun
36	Plutôt d'accord	Plutôt d'accord	Tout à fait d'accord	Pas d'accord du tout	

Annexe C.

Le modèle : détail des résultats.

Les fichiers du progamme sont disponibles sur simple demande à :

nicolas.jullien@enst-bretagne.fr

Le détail des résultats des simulations.

Externalités locales dominantes, forte préférence pour la standardisation (a=2,5 et b=0,2).

Distribution uniforme.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	15	15	16	16	17	18	17	20	19	20	19
	0,1	16	17	17	18	20	20	20	20	21	24	22
	0,2	19	20	19	21	22	24	25	25	25	29	29
	0,3	21	22	24	27	27	28	30	30	34	38	38
	0,4	29	28	32	34	33	36	37	41	41	45	47
	0,5	35	36	41	42	47	48	53	55	63	76	84
	0,6	50	49	55	61	67	78	104	100	133	185	204
	0,7	66	75	91	94	106	138	171	275	349	500	500
	0,8	116	111	160	257	188	364	500	500	500	500	500
	0,9	230	423	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Distribution bimodale centrée en - 0.,5 et 0,5, écarts- types 0,2.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	16	16	17	17	17	17	17	19	19	18	19
	0,1	17	16	18	18	19	20	21	19	20	23	24
	0,2	18	19	19	20	22	23	23	24	25	26	28
	0,3	20	22	23	24	28	26	31	34	32	38	34
	0,4	26	27	31	32	33	36	39	42	50	48	49
	0,5	33	37	38	45	43	44	55	59	73	67	86
	0,6	41	49	55	57	69	69	87	109	144	167	396
	0,7	69	78	87	86	120	157	305	228	500	500	500
	0,8	114	149	180	213	373	500	500	500	500	500	500
	0,9	345	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Distribution normale centrée en 0, écart- type 0,4.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	13	14	14	15	15	16	17	17	18	19	21
	0,1	15	16	16	17	19	19	20	23	23	29	32
	0,2	18	22	22	24	23	26	34	31	36	37	43
	0,3	24	28	30	35	34	42	44	53	69	83	121
	0,4	34	39	52	56	59	78	98	110	148	228	410
	0,5	54	63	106	98	141	145	246	437	389	500	500
	0,6	97	164	157	257	423	431	500	500	500	500	500
	0,7	162	328	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Externalités locales dominantes, faible préférence pour la standardisation (a=0,5 et b=0,2)

Distribution uniforme.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	500	500	283	500	282	500	500	500	183	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
	0,4	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Distribution normale centrée en 0, écart- type 0,4.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	25	28	24	25	26	30	28	31	28	32	32
	0,1	27	30	31	28	30	31	29	33	34	33	35
	0,2	31	28	29	32	35	37	38	37	33	37	39
	0,3	34	33	38	37	41	38	41	41	51	43	48
	0,4	40	37	39	48	47	47	50	46	49	49	48
	0,5	36	49	46	43	50	51	50	54	54	52	55
	0,6	43	42	51	51	63	59	56	75	65	124	81
	0,7	67	76	76	59	70	83	66	94	99	81	114
	0,8	84	77	62	110	86	94	106	184	225	204	213
	0,9	75	146	102	156	282	157	402	204	281	500	500
	1	111	500	212	216	500	500	500	500	500	500	500

Distribution bimodale centrée en - 0.,5 et 0,5, écarts- types 0,2.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	500	500	500	500	500	500	500	500	500	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
	0,4	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Externalités globales dominantes, forte préférence pour la standardisation (a=2,5 et b=4/5).

Distribution uniforme.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	15	15	18	21	24	33	52	90	500	500	500
0,1	15	16	18	23	26	36	65	187	500	500	500
0,2	16	16	19	22	28	36	72	500	500	500	500
0,3	16	17	20	23	33	46	82	500	500	500	500
0,4	16	17	19	24	34	55	121	500	500	500	500
0,5	16	19	23	28	34	57	181	500	500	500	500
0,6	15	18	22	28	36	66	500	500	500	500	500
0,7	17	19	23	29	45	93	500	500	500	500	500
0,8	18	21	27	28	53	132	500	500	500	500	500
0,9	19	22	26	33	55	124	500	500	500	500	500
1	20	22	29	36	66	500	500	500	500	500	500

Distribution bimodale centrée en -0.,5 et 0,5, écarts-types 0,2.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	14	17	18	20	23	29	40	155	500	500	500
0,1	15	16	19	21	25	34	52	500	500	500	500
0,2	16	19	19	21	25	34	64	500	500	500	500
0,3	16	18	20	22	28	41	70	500	500	500	500
0,4	18	17	20	24	29	49	175	500	500	500	500
0,5	16	18	18	23	33	57	500	500	500	500	500
0,6	16	19	20	26	39	82	500	500	500	500	500
0,7	19	21	21	26	42	79	500	500	500	500	500
0,8	19	20	23	35	43	109	500	500	500	500	500
0,9	18	20	24	34	52	500	500	500	500	500	500
1	19	21	24	38	59	500	500	500	500	500	500

Distribution normale, centrée en 0, écart-type 0,4.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	12	14	16	25	47	500	500	500	500	500	500
0,1	12	15	19	25	95	500	500	500	500	500	500
0,2	13	15	19	30	119	500	500	500	500	500	500
0,3	14	16	21	38	500	500	500	500	500	500	500
0,4	14	17	22	43	500	500	500	500	500	500	500
0,5	14	19	28	83	500	500	500	500	500	500	500
0,6	15	20	29	147	500	500	500	500	500	500	500
0,7	16	21	44	496	500	500	500	500	500	500	500
0,8	18	24	54	500	500	500	500	500	500	500	500
0,9	18	28	68	500	500	500	500	500	500	500	500
1	20	31	96	500	500	500	500	500	500	500	500

Externalités globales dominantes, faible préférence pour la standardisation (a=0,5 et b=4/5).

Distribution uniforme.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	500	500	286	500	500	500	500	500	500	500	500
	0,1	387	118	500	500	500	500	500	500	500	500	500
	0,2	481	497	500	500	500	500	500	500	500	500	500
	0,3	72	126	496	500	500	500	500	500	500	500	500
	0,4	231	362	500	500	500	500	500	500	500	500	500
	0,5	452	500	500	500	500	500	500	500	500	500	500
	0,6	220	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	216	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Distribution bimodale centrée en -0.,5 et 0,5, écarts-types 0,2.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	500	500	500	500	500	500	500	500	500	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
	0,4	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Distribution normale, centrée en 0, écart-type 0,4.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
Alpha	**0**	25	26	32	29	33	41	44	65	67	80	99
	0,1	25	24	31	33	37	47	45	58	63	82	125
	0,2	22	31	32	40	32	41	51	49	66	90	140
	0,3	27	26	31	36	41	39	48	61	74	101	210
	0,4	27	32	29	46	36	44	60	64	75	119	171
	0,5	28	32	30	35	40	51	47	62	98	82	162
	0,6	27	30	29	39	44	49	61	66	93	200	278
	0,7	28	30	36	43	42	45	50	68	77	179	312
	0,8	30	28	35	34	42	67	69	93	102	151	496
	0,9	31	31	38	36	51	53	68	78	107	177	497
	1	31	37	40	43	53	55	63	101	153	243	500

Externalités globales et locales égales, forte préférence pour la standardisation (a=2,5 ou 5 et b=0,5).

Distribution uniforme.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	15	15	16	18	18	22	23	26	29	43	61
0,1	15	16	18	19	21	22	26	34	39	53	73
0,2	17	18	19	23	25	29	36	41	49	80	188
0,3	17	18	19	23	25	29	38	48	90	258	500
0,4	21	21	25	28	34	42	51	72	174	500	500
0,5	22	25	28	36	41	56	87	117	500	500	500
0,6	24	26	32	40	51	64	136	500	500	500	500
0,7	28	30	42	56	68	117	500	500	500	500	500
0,8	34	38	56	59	122	500	500	500	500	500	500
0,9	40	51	59	123	387	500	500	500	500	500	500
1	48	64	91	318	500	500	500	500	500	500	500

Distribution normale, centrée en 0, écart-type 0,4.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	13	14	14	16	18	25	33	49	500	500	500
0,1	14	15	16	20	25	36	62	417	500	500	500
0,2	15	16	21	24	33	61	278	500	500	500	360
0,3	18	20	26	34	52	123	500	500	500	500	500
0,4	22	25	35	52	86	500	500	500	500	500	500
0,5	27	33	48	80	500	500	500	500	500	500	500
0,6	31	42	77	500	500	500	500	500	500	500	500
0,7	52	72	498	500	500	500	500	500	500	500	500
0,8	67	188	500	500	500	500	500	500	500	500	500
0,9	194	500	500	500	500	500	500	500	500	500	500
1	377	500	500	500	500	500	500	500	500	500	500

Distribution uniforme, a=5 (très forte préférence pour la standardisation).

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	13	13	15	16	20	27	41	93	500	500	500
0,1	13	14	16	20	25	36	65	500	500	500	500
0,2	15	16	20	27	33	68	500	500	500	500	500
0,3	16	20	25	37	58	500	500	500	500	500	500
0,4	19	25	36	49	207	500	500	500	500	500	500
0,5	26	35	51	99	500	500	500	500	500	500	500
0,6	39	54	133	500	500	500	500	500	500	500	500
0,7	52	96	500	500	500	500	500	500	500	500	500
0,8	106	500	500	500	500	500	500	500	500	500	500
0,9	382	500	500	500	500	500	500	500	500	500	500
1	500	500	500	500	500	500	500	500	500	500	500

Distribution bimodale centrée en -0.,5 et 0,5, écarts-types 0,2.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	15	16	16	18	18	19	22	25	29	35	51
0,1	17	18	19	18	20	24	27	29	40	58	90
0,2	17	17	20	21	23	26	30	35	46	85	360
0,3	18	18	22	23	26	32	36	53	67	205	500
0,4	21	21	22	27	31	38	48	73	280	500	500
0,5	22	23	27	29	38	54	73	169	500	500	500
0,6	24	26	32	39	42	73	95	500	500	500	500
0,7	25	31	38	53	67	155	500	500	500	500	500
0,8	30	39	52	72	106	500	500	500	500	500	500
0,9	38	44	66	86	500	500	500	500	500	500	500
1	50	61	104	500	500	500	500	500	500	500	500

Externalités globales et locales égales, faible préférence pour la standardisation (a=0,5 et b=0,5).

Distribution uniforme.

Ici, il presque surement cohabitation

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	311	500	500	500	500	500	500	500	500	500	500
0,1	500	500	500	500	500	500	500	500	500	500	500
0,2	500	500	500	500	500	500	500	500	500	500	500
0,3	500	500	500	500	500	500	500	500	500	500	500
0,4	500	500	500	500	500	500	500	500	500	500	500
0,5	500	500	500	500	500	500	500	500	500	500	500
0,6	500	500	500	500	500	500	500	500	500	500	500
0,7	500	500	500	500	500	500	500	500	500	500	500
0,8	500	500	500	500	500	500	500	500	500	500	500
0,9	500	500	500	500	500	500	500	500	500	500	500
1	500	500	500	500	500	500	500	500	500	500	500

Distribution bimodale centrée en -0.,5 et 0,5, écarts-types 0,2.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	500	500	500	500	500	500	500	500	500	500	500
0,1	500	500	500	500	500	500	500	500	500	500	500
0,2	500	500	500	500	500	500	500	500	500	500	500
0,3	500	500	500	500	500	500	500	500	500	500	500
0,4	500	500	500	500	500	500	500	500	500	500	500
0,5	500	500	500	500	500	500	500	500	500	500	500
0,6	500	500	500	500	500	500	500	500	500	500	500
0,7	500	500	500	500	500	500	500	500	500	500	500
0,8	500	500	500	500	500	500	500	500	500	500	500
0,9	500	500	500	500	500	500	500	500	500	500	500
1	500	500	500	500	500	500	500	500	500	500	500

Distribution normale, centrée en 0, écart-type 0,4.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	25	27	25	29	30	36	39	38	39	44	52
0,1	28	29	29	29	31	42	33	42	42	41	50
0,2	27	28	29	32	33	35	39	41	48	52	68
0,3	30	33	30	39	40	45	52	48	56	83	67
0,4	32	30	34	40	43	46	41	57	63	67	83
0,5	30	39	35	42	39	44	48	52	69	70	102
0,6	38	42	43	46	48	72	70	84	85	95	151
0,7	34	40	42	49	58	54	65	66	84	117	122
0,8	39	43	50	55	57	71	73	84	134	181	204
0,9	46	50	54	58	56	65	184	181	258	187	500
1	39	47	67	65	82	78	77	241	335	423	500

Externalités globales et locales égales, hybridation, forte préférence pour la standardisation (a=2,5 et b=0,5).
Distribution bimodale centrée en - 0,5 et 0,5, écart- type 0,2.

Pas de déplacement : simulation «témoin ».

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	15	17	16	17	20	23	32	38	72	500
	0,1	14	15	18	18	21	23	30	44	73	500	500
	0,2	16	18	17	20	25	28	35	80	204	500	500
	0,3	17	18	19	24	26	42	55	245	500	500	500
Alpha	**0,4**	19	21	25	30	41	53	186	500	500	500	500
	0,5	22	25	30	44	61	210	500	500	500	500	500
	0,6	24	31	37	53	122	500	500	500	500	500	500
	0,7	29	40	59	102	500	500	500	500	500	500	500
	0,8	37	53	104	500	500	500	500	500	500	500	500
	0,9	59	82	393	500	500	500	500	500	500	500	500
	1	85	455	500	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 200 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	15	17	16	17	21	28	500	500	500	500
	0,1	15	16	16	19	22	29	500	500	500	500	500
	0,2	15	17	18	22	26	500	500	500	500	500	500
	0,3	17	19	21	26	46	500	500	500	500	500	500
Alpha	**0,4**	18	22	27	41	500	500	500	500	500	500	500
	0,5	20	30	53	500	500	500	500	500	500	500	500
	0,6	28	38	500	500	500	500	500	500	500	500	500
	0,7	55	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 200 coups du pôle de droite.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	14	16	15	16	19	21	29	36	67	500
	0,1	15	14	15	17	19	22	25	35	59	500	500
	0,2	14	15	16	19	24	26	37	52	244	500	500
	0,3	16	16	19	23	24	35	56	144	500	500	500
Alpha	**0,4**	17	19	22	27	35	51	173	500	500	500	500
	0,5	18	22	26	35	49	106	500	500	500	500	500
	0,6	24	26	38	49	95	500	500	500	500	500	500
	0,7	27	34	46	94	500	500	500	500	500	500	500
	0,8	33	48	88	191	500	500	500	500	500	500	500
	0,9	51	65	171	500	500	500	500	500	500	500	500
	1	66	144	500	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 1000 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	15	17	16	17	20	23	33	500	500	500
	0,1	14	15	16	19	20	23	28	43	500	500	500
	0,2	16	17	19	20	23	29	52	500	500	500	500
	0,3	16	17	19	24	29	44	500	500	500	500	500
Alpha	**0,4**	19	21	23	31	39	500	500	500	500	500	500
	0,5	21	23	28	44	500	500	500	500	500	500	500
	0,6	26	33	39	500	500	500	500	500	500	500	500
	0,7	30	46	81	500	500	500	500	500	500	500	500
	0,8	46	69	500	500	500	500	500	500	500	500	500
	0,9	73	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Externalités globales et locales égales, hybridation, faible préférence pour la standardisation (a=0,5 et b=1/2).
Distribution bimodale centrée en -0,5 et 0,5, écart-type 0,2.

Pas de déplacement : simulation «témoin ».

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	500	500	500	500	500	500	500	500	500	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
Alpha	**0,4**	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Il y a diffusion pour (alpha, beta) = (0.4, 0.6) ; (0.5, 0.5) ; (0.6 ; 0.4)
Jusqu'à environ 450, chacun gagne la moitié de la population.

Déplacement de 5 % tous les 200 coups du pôle de droite.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	28	29	30	31	34	32	37	37	35	37	37
	0,1	31	30	30	31	34	36	34	34	37	39	39
	0,2	31	34	33	33	31	34	37	39	36	41	41
	0,3	32	31	32	34	37	35	36	40	38	41	38
Alpha	**0,4**	33	33	36	35	37	36	35	41	40	39	41
	0,5	34	34	34	38	36	35	37	40	40	42	43
	0,6	32	32	37	41	40	40	40	41	46	47	49
	0,7	37	39	35	40	37	41	42	46	41	42	44
	0,8	37	37	36	37	43	39	43	42	44	53	43
	0,9	36	38	39	36	41	42	44	45	48	46	47
	1	38	36	38	42	45	45	46	45	46	51	52

Déplacement de 5 % tous les 200 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	500	500	500	500	500	500	500	500	500	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
Alpha	**0,4**	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Externalités locales dominantes, hybridation, faible préférence pour la standardisation (a=0,5 et b=1/5).
Distribution bimodale centrée en - 0.5 et 0.5, écart- type 0,2.

Pas de déplacement.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	500	500	500	500	500	500	500	500	500	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
Alpha	**0,4**	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Il y a diffusion pour (alpha, beta) = (0.5, 0.5) ; (0.6 ; 0.4) ; (0.6, 0.5)
autour de 450- 500, chacun gagne la moitié de la population
mais ne peut aller plus loin (voir les courbes).

Déplacement de 5 %tous les 200 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	500	500	500	500	500	500	500	500	500	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
	0,4	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Il y a début de diffusion (ça monte jusqu'à 300)
pour (alpha, beta) = (0.5, 0.5), puis ça retombe.

Déplacement de 5%tous les 200 coups du pôle de droite.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	30	28	30	29	33	32	33	31	30	33	31
	0,1	29	31	31	30	30	30	34	33	33	33	31
	0,2	30	30	33	35	32	35	34	35	33	35	37
	0,3	35	34	31	35	36	35	34	33	33	31	35
Alpha	**0,4**	32	33	36	34	34	35	38	36	36	37	38
	0,5	35	34	32	39	32	39	40	36	38	39	41
	0,6	36	32	39	37	36	40	37	38	40	39	42
	0,7	43	38	40	42	41	41	41	43	42	46	43
	0,8	38	41	42	43	41	44	42	45	44	46	45
	0,9	45	41	41	43	38	46	49	49	47	49	45
	1	41	42	47	51	47	44	47	49	54	53	52

Déplacement de 5 %tous les 1000 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	500	500	500	500	500	500	500	500	500	500	500
	0,1	500	500	500	500	500	500	500	500	500	500	500
	0,2	500	500	500	500	500	500	500	500	500	500	500
	0,3	500	500	500	500	500	500	500	500	500	500	500
	0,4	500	500	500	500	500	500	500	500	500	500	500
	0,5	500	500	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Il y a début de diffusion (ça monte jusqu'à 400)
pour (alpha, beta) = (0.5, 0.5), puis ça retombe.

Externalités locales dominantes, hybridation, forte préférence pour la standardisation (a=2,5 et b=1/5).
Distribution bimodale centrée en -0,5 et 0,5, écart-type 0,2.

Pas de déplacement.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	15	14	15	16	16	18	16	17	18	18
	0,1	15	16	17	16	17	18	18	20	21	22	23
	0,2	19	19	20	20	22	21	24	27	28	26	31
	0,3	21	23	26	26	31	32	31	37	41	46	47
Alpha	**0,4**	27	30	32	41	38	41	51	54	56	77	90
	0,5	42	46	50	55	59	75	83	95	139	341	500
	0,6	55	72	94	94	124	143	241	500	500	500	500
	0,7	95	139	219	434	500	500	500	500	500	500	500
	0,8	349	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 200 coups du pôle de droite.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	14	15	14	15	15	16	16	17	17	17
	0,1	15	15	16	15	17	17	17	18	19	19	22
	0,2	17	17	17	18	20	20	22	23	25	28	27
	0,3	20	21	23	23	25	28	30	31	35	37	43
Alpha	**0,4**	28	28	32	37	40	41	45	52	64	65	92
	0,5	39	43	45	47	50	53	60	87	97	136	232
	0,6	54	64	66	77	81	107	129	177	326	500	500
	0,7	83	103	125	154	182	216	500	500	500	500	500
	0,8	184	239	227	438	500	500	500	500	500	500	500
	0,9	382	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 200 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	15	15	15	15	15	16	16	16	500	17	20
	0,1	16	16	16	16	17	19	20	20	500	27	25
	0,2	19	20	20	21	24	25	29	31	500	41	56
	0,3	26	27	29	35	37	33	61	49	500	500	500
Alpha	**0,4**	35	40	47	64	228	500	500	500	500	500	500
	0,5	67	145	500	500	500	500	500	500	500	500	500
	0,6	500	500	500	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 1000 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	15	16	15	15	16	17	17	18	17	19
	0,1	15	16	16	17	17	18	18	19	20	23	25
	0,2	18	19	21	19	21	23	26	27	27	32	34
	0,3	22	24	23	26	29	31	34	34	41	50	67
Alpha	**0,4**	31	34	34	39	47	48	59	76	500	500	500
	0,5	49	49	55	65	102	500	500	500	500	500	500
	0,6	60	107	251	500	500	500	500	500	500	500	500
	0,7	500	500	500	500	500	500	500	500	500	500	500
	0,8	500	500	500	500	500	500	500	500	500	500	500
	0,9	500	500	500	500	500	500	500	500	500	500	500
	1	500	500	500	500	500	500	500	500	500	500	500

Externalités globales dominantes, hybridation, faible préférence pour la standardisation (a=0,5 et b=4/5).
Distribution bimodale centrée en -0,5 et 0,5, écart-type 0,2.

Pas de déplacement.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	500	500	500	500	500	500	500	500	500	500	500
0,1	500	500	500	500	500	500	500	500	500	500	500
0,2	500	500	500	500	500	500	500	500	500	500	500
0,3	500	500	500	500	500	500	500	500	500	500	500
0,4	500	500	500	500	500	500	500	500	500	500	500
0,5	500	500	500	500	500	500	500	500	500	500	500
0,6	500	500	500	500	500	500	500	500	500	500	500
0,7	500	500	500	500	500	500	500	500	500	500	500
0,8	500	500	500	500	500	500	500	500	500	500	500
0,9	500	500	500	500	500	500	500	500	500	500	500
1	500	500	500	500	500	500	500	500	500	500	500

Il y a diffusion pour (alpha, beta) = (0.5, 0.5) ; (0.5 ; 0.6) ; (0.5 ; 0.9) ;
Jusqu'à environ 450.
Chacun gagne la moitié de la population mais ne peut aller plus loin.

Déplacement de 5 % tous les 200 coups du pôle de droite.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	29	31	32	30	31	38	38	37	43	41	48
0,1	27	32	32	36	35	34	42	42	44	46	46
0,2	29	31	31	33	33	35	38	42	44	45	45
0,3	31	32	34	35	35	36	35	43	44	48	48
0,4	32	33	33	36	35	37	38	44	43	45	48
0,5	30	30	35	34	37	37	40	42	47	47	53
0,6	31	33	33	32	37	38	44	40	42	45	51
0,7	31	33	37	38	38	38	43	43	43	47	51
0,8	31	34	37	35	38	36	39	44	44	48	49
0,9	32	32	34	37	38	41	41	46	46	50	56
1	32	32	33	35	36	38	42	44	44	50	59

Déplacement de 5 % tous les 200 coups du pôle de gauche.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	500	500	500	500	500	500	500	500	500	500	500
0,1	500	500	500	500	500	500	500	500	500	500	500
0,2	500	500	500	500	500	500	500	500	500	500	500
0,3	500	500	500	500	500	500	500	500	500	500	500
0,4	500	500	500	500	500	500	500	500	500	500	500
0,5	500	500	500	500	500	500	500	500	500	500	500
0,6	500	500	500	500	500	500	500	500	500	500	500
0,7	500	500	500	500	500	500	500	500	500	500	500
0,8	500	500	500	500	500	500	500	500	500	500	500
0,9	500	500	500	500	500	500	500	500	500	500	500
1	500	500	500	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 1000 coups du pôle de gauche.

Alpha \ Beta	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	500	500	500	500	500	500	500	500	500	500	500
0,1	500	500	500	500	500	500	500	500	500	500	500
0,2	500	500	500	500	500	500	500	500	500	500	500
0,3	500	500	500	500	500	500	500	500	500	500	500
0,4	500	500	500	500	500	500	500	500	500	500	500
0,5	500	500	500	500	500	500	500	500	500	500	500
0,6	500	500	500	500	500	500	500	500	500	500	500
0,7	500	500	500	500	500	500	500	500	500	500	500
0,8	500	500	500	500	500	500	500	500	500	500	500
0,9	500	500	500	500	500	500	500	500	500	500	500
1	500	500	500	500	500	500	500	500	500	500	500

Il y a début de diffusion (ça monte jusqu'à 450)
pour (alpha, beta) = (0.5, 0.5), puis ça retombe.

Externalités globales dominantes, hybridation, forte préférence pour la standardisation (a=2,5 et b=4/5).
Distribution bimodale centrée en -0,5 et 0,5, écart-type 0,2.

Pas de déplacement.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	16	18	19	23	48	500	500	500	500	500
	0,1	14	15	17	20	25	50	500	500	500	500	500
	0,2	14	16	19	21	33	83	500	500	500	500	500
	0,3	15	15	18	22	34	161	500	500	500	500	500
Alpha	**0,4**	15	16	20	26	37	500	500	500	500	500	500
	0,5	15	17	19	27	52	500	500	500	500	500	500
	0,6	15	16	21	30	78	500	500	500	500	500	500
	0,7	16	18	22	37	78	500	500	500	500	500	500
	0,8	16	19	26	40	372	500	500	500	500	500	500
	0,9	17	21	28	43	500	500	500	500	500	500	500
	1	18	22	33	62	500	500	500	500	500	500	500

Déplacement de 5 % tous les 200 coups du pôle de droite.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	13	14	17	17	22	37	500	500	500	500	500
	0,1	14	15	16	20	22	45	500	500	500	500	500
	0,2	14	14	16	19	26	73	500	500	500	500	500
	0,3	14	15	17	19	35	154	500	500	500	500	500
Alpha	**0,4**	14	16	17	21	32	500	500	500	500	500	500
	0,5	15	16	18	24	45	500	500	500	500	500	500
	0,6	15	17	20	27	63	500	500	500	500	500	500
	0,7	15	18	20	30	119	500	500	500	500	500	500
	0,8	16	17	23	41	439	500	500	500	500	500	500
	0,9	16	18	25	48	500	500	500	500	500	500	500
	1	17	20	25	55	500	500	500	500	500	500	500

Déplacement de 5 % tous les 200 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	16	18	20	26	500	500	500	500	500	500
	0,1	14	16	17	21	500	500	500	500	500	500	500
	0,2	14	16	18	22	500	500	500	500	500	500	500
	0,3	15	16	18	23	500	500	500	500	500	500	500
Alpha	**0,4**	16	16	19	27	500	500	500	500	500	500	500
	0,5	15	17	21	500	500	500	500	500	500	500	500
	0,6	16	18	23	500	500	500	500	500	500	500	500
	0,7	15	18	27	500	500	500	500	500	500	500	500
	0,8	16	20	28	500	500	500	500	500	500	500	500
	0,9	18	20	41	500	500	500	500	500	500	500	500
	1	18	22	50	500	500	500	500	500	500	500	500

Déplacement de 5 % tous les 1000 coups du pôle de gauche.

		Beta										
		0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
	0	14	16	18	19	23	500	500	500	500	500	500
	0,1	15	15	16	20	26	500	500	500	500	500	500
	0,2	14	16	17	21	30	500	500	500	500	500	500
	0,3	15	17	19	23	37	500	500	500	500	500	500
Alpha	**0,4**	15	17	18	25	500	500	500	500	500	500	500
	0,5	16	17	20	28	500	500	500	500	500	500	500
	0,6	15	18	21	29	500	500	500	500	500	500	500
	0,7	16	18	22	39	500	500	500	500	500	500	500
	0,8	17	19	25	40	500	500	500	500	500	500	500
	0,9	18	20	28	500	500	500	500	500	500	500	500
	1	19	21	31	500	500	500	500	500	500	500	500

Bibliographie

[1] Abecassis C., 1997, «Les coûts de transaction : état de la théorie», Réseau n°84, pp. 11-19.

[2] Abernathy W. J., Utterback J. M., 1978, «Patterns of Industrial Innovation», Technology Review, 80(7), pp. 40-47.

[3] Akerlof G. A., 1970, «The Market for Lemons : Qualitative Uncertainty and the Market Mechanism», Quaterly Journal of Economics, 84, pp. 488-500.

[4] Allen D., 1983, «Collective invention», Journal of Economic Behavior and Organization, 4, pp. 1-24.

[5] Allen D., 1988, «Network Externalities and critical mass», Telecommunications Policy, septembre.

[6] Alstyne ., Brynjolfsson .,, 1997,

[7] Altersohn C., 1992, *De la sous-traitance au partenariat industriel*, Paris, L'Harmattan.

[8] Aoki M., 1984a, *The Cooperative Game Theory of the Firm*, Oxford, Clarendon Press.

[9] Aoki M., 1984b, *The Economic Analysis of the Japanese Firm*, Amsterdam, North Holland.

[10] Arrow K. J., 1962 a, «Economic Welfare and the Allocation of Ressources to Invention», dans Nelson (ed.), *The Rate and Direction of Inventive Activity : Economic and Social Factors*, Princeton, Princeton University Press.

[11] Arrow K. J., 1962 b, «The Economic Implications of Learning by Doing», Review of Economic Studies, 29, pp. 155-173.

[12] Arthur B., 1988a, «Competiting technologies : an overview», dans Dosi et al. (eds), *Technical Change and Economic Theory*, Pinter Publishers.

[13] Arthur B., 1988b, «Self-reinforcing mechanisms in economics», dans Anderson, Arrow, Pines (eds.), *The Economy as an evolving Complex System*, Addison Wesley Publish, New-York, pp. 9-31.

[14] Arthur B., 1989, «Competiting technologies, increasing returns and lock-in by historical events», The Economic Journal, n°99, pp. 116-131.

[15] Arthur B., Ermoliev Y., Kaniovski Y., 1987, «Path-dependent processes and the emergence of macro-structure», European Journal of Operational Research, n°30, pp. 294-303.

[16] Attia R., Davy I., Rizoulières R., 2000, «Innovative Labor and Intellectual Property Market in the Semiconductor Industry», dans Guilhon (ed.), *Technologies and Markets for Knowledge*, Kluwer Academic Publishers.

[17] Auerbach P., 1988, *Competition. The Economics of Industrial Change*, Basil Blackwell, Londres.

[18] Axelrod R., 1992, Donnant-donnant ; théorie du comportement coopératif, Odile Jacob, Paris, traduit de Axelrod, 1984, *The Evolution of Cooperation*, Basic Books, New York.

[19] Bakos Y., Brynjolfsson E, 2000, «Bundling and Competition on the Internet : Aggregation Strategies for Information Goods», Marketing Science, janvier.

[20] Balassa B., 1986, «Prices, Incentives and Economic Growth», dans Balassa (ed.), 1986, *Economic Incentives*, New York, St. Martin's Press.

[21] Bandville (de) E., Chanaron J.-J., 1991, *Vers un système automobile européen*, Economica, Paris.

[22] Bandura A. (eds.), 1995, Self-Efficacy in changing Societies, Cambridge, Cambridge University Press.

[23] Barnard C., 1938, *The Functions of the Executive*, rééd. Cambridge, Harvard University Press, 1968.

[24] Bator F. M., 1958, «the Anatomy of Market Failure», Quaterly Journal of Economics, n°72, pp. 351-379.

[25] Baudry B., 1995, *Économie des relations inte-entreprises*, La Découverte, Paris, 125 p.

[26] Baumol W., Blackman S.A.B., Wolff E.N.,1985, «Unbalanced Growth Revisited : asymptotic stagnancy and new evidence», American Economic Review, vol. 75, n°4, pp. 332-337, traduit dans «Nouvel examen du modèle de croissance déséquilibré : application au cas américain», 1986, Problèmes Économiques, n°1970, pp. 14-22.

[27] Baumol W., Blackman S.A.B., Wolff E.N.,1989, *Productivity and American Leadership : the Long View*, MIT Press, Cambridge.

[28] Behlendorf B., 1999, «Open Source as a Business Strategy», dans Di Bona & al. [1999].

[29] Benezech D., 1995, *L'apport du concept de norme technique à l'analyse de la diffusion technologique*, Thèse de doctorat de l'université de Rennes 1, mention Sciences Économiques.

[30] Benzoni L., Jutand F., 1991, «le rythme de l'innovation», Communications et Stratégies, n°2.

[31] Bernoussi K., Torres A., 1991, «Informatique, individu, métier, la nouvelle trilogie», Science et Technologie, n°33, janvier, pp. 60-63.

[32] Besen S. M., 1986, «Private Copying, Reproduction Costs, and the Supply of Intellectual Property», Information Economics and Policy 2.

[33] Besen S. M., Kirby S. N., 1989, «Private Copying, Appropriability, and Optimal Copying Royalties», Journal of Law and Economics.

[34] Besen S. M., Farrell J., 1994, «Choosing How to Compete : Strategies and Tactics in Standardization», Journal of Economic Perspectives, vol. 8, n°2, printemps, pp. 133-150.

[35] Blanchet J., 1990, *Gestion du bénévolat*, Paris, Economica, 95 p.

[36] Bloch F., Manceau D., 1996, «product preannouncement, market cannibalization and price competition», doccument de travail, HEC, mai.

[37] Blonde M.-H., Aguer D., 2000, *Les services marchands en 1999. Dynamisme des jeunes pousses ... et des autres*, INSEE Première, n°725, juillet.

[38] Bowman I., 1998, «Conceptual Architecture of the Linux Kernel», http ://plg.uwaterloo.ca/˜itbowman/CS746G/a1/

[39] Breese, 2000, «Le brevet, moteur de l'innovation», réponse à la consultation de la Commission Européenne sur la brevetabilité du logiciel en Europe.

[40] Breton P., 1990, «Une histoire de l'informatique», La Découverte, Paris 1987, réédité en collection *Point Sciences*, Le Seuil, Paris, 261 p.

[41] Brock G., 1975, *The US Computer Industry : A Study of Market Power,* Ballinger, Cambridge.

[42] Brooks F., 1974, *The Mythical Man Month : Essays on Software Engineering,* Addison-Wesley Pub Co.

[43] Brousseau É., 1993, *L'économie des contrats : technologies de l'information et coordination interentreprises*, P.U.F. (Économie et liberté), 368 p.

[44] Buchanan J. M., 1965, «An Economic Theory of Clubs», Economica, n°32, pp. 1-14.

[45] Buchanan J., Stubblebine W., 1962, «Externality», Economica, novembre, n°29, pp. 371-384.

[46] Callon M., 1991, «Réseaux technico-économiques et irréversibilités», dans Boyer, Chavance, Godard (eds.), *Les figures de l'irréversibilité en économie*, Éditions de l'École des Hautes Études en Sciences Sociales, pp. 195-230.

[47] Callon M., 1992, «Variété et irréversibilité dans les réseaux de conception et d'adoption des techniques», dans Foray & Freeman [1992], pp. 275-324.

[48] Callon M., 1994, «Is science a public good ? Fifth Mullins Lecture», Science, Tehcnology and Human Value, vol.19, n°4.

[49] Carayol N., 1999, «The Dynamics of Knowledge and Finalization of Scientific Research. the Case of engineering Sciences.», présenté au colloque EMAEE, Grenoble, juin.

[50] Carcenac T., 2001, *Pour une administration électronique citoyenne*, Rapport au Premier Ministre, avril,
http ://www.premier-ministre.gouv.fr/fr/p.cfm ?ref=22503.

[51] Chamberlain E.H., 1953, *La théorie de la concurrence monopolistique*, Paris, trad. française, PUF.

[52] Chandler A., 1962, *Strategy and Structure*, MIT Press.

[53] Chandler A., 1977, *The Visible Hand*, Harvard University Press, Cambridge, Mass.

[54] Charbit C., Zimmermann J.-B., 1998, «Systèmes d'information et entreprises : convergence ou incertitude ?», Terminal n°78, pp. 81-102.

[55] Church J., Gandal N., 1992, «Network effects,, software provision, and standardization», Journal of Industrial Economics, vol. 40, n°1, pp. 85-100.

[56] Church J., Gandal N., 1993, «Complementary network externalities and technological adoption», International Journal of Industrial Organisation, vol. 11, n°2, pp. 239-260.

[57] Clément-Fontaine M., 1999, *Étude de la GNU GPL*, mémoire de DEA, université de Montpellier, http ://www.crao.net/gpl/.

[58] Coase R. H., 1937, «The nature of the firm», Economica n°4, pp. 386-405, traduit dans Coase, 1987, «la nature de la firme», Revue française d'économie, vol. 2/1, pp. 133-163.

[59] Coase R. H., 1959, «The FCC», Journal of Law and Economics, n°2, pp. 1-40 (traduction française : «Faut-il vendre les fréquences», Réseau n°64, mars-avril 1994, pp. 139-161).

[60] Coase R. H., 1960, «The Problem of Social Cost», Journal of Law and Economics, n°3, pp. 1-44.

[61] Cohen M. D., 1983, «Conflict and Complexity : Goal Diversity and Organization Search Effectiveness», American Political Science Review, 78, pp. 435-451.

[62] Cohendet P., Creplet F., Dupouet O., 2000, «Communities of Practices and Epistemic Communities : a renewed approach of organizational learning within firms», document de travail, BETA.

[63] Constant D., Kiesler S., Sproull L., 1994, «What's Mine Is Ours, or Is It ? A Study of Attitudes about Information Sharing», Information Systems Research 5(4), pp. 400-421.

[64] Cornes R., Sandler T., 1991, «The theory of externalities, public goods, and club goods», Cambridge University Press.

[65] Cowan R., 1988, *Nuclear power reactors : a study in technology lock-in*, New York University.

[66] Cowan R., Foray D., 1997, «The economics of knowledge and the diffusion of knowledge», Industrial and corporate change, vol. 16, n°3.

[67] Crozet Y., 1997, *Analyse économique de l'État*, Armand Colin, 191 p.

[68] Cusumano M., Takeishi A., 1991, «Supplier Relations and Management : a Survey of Japanese, Japanese-Transplant, and US Auto Plants», Strategic Management Journal, vol. 12, pp. 563-588.

[69] Dalle J.-M., 1995, «Dynamiques d'adoption, coordination et diversité», Revue Économique, 46, pp. 1081-1098.

[70] Dalle J.-M., 1997, «Heterogeneity vs. externalities : a tale of possible technological landscapes», Journal of Evolutionary Economics 7, pp. 395-413.

[71] Dalle J.-M., 1998 a, «Heterogeneity and rationality in stochastic interaction models», dans Cohendet, Stahn (eds), *The economics of networks : behaviors and interactions*, Springer Verlag, Berlin, pp. 123-145.

[72] Dalle J.-M., 1998 b, «Local interaction structures, heterogeneity, and the diffusion of technological innovations», dans Orléan, Lesourne (eds), *Self-organization and evolutionary approaches : new developments*, Economica, Paris, pp. 240-261.

[73] Dalle J.-M., Foray D., 1998 a, «The innovation vs. standardization dilemma : some insights from stochastic interactions models», dans Schweitzer, Silverberg (eds), *Evolution and Self-Organization in Economics*, Duncker & Humblot, Berlin, pp. 147-182.

[74] Dalle J.-M., Foray D., 1998 b, *L'institution brevet dans une économie fondée sur la connaissance : éléments d'analyse*, rapport de recherche pour le Commissariat Général au Plan.

[75] Dalle J.-M., Foray D., 1999, «Quand les agents sont-ils négligeables (ou décisifs) ? Une approche de l'économie de l'innovation par les modèles stochastiques d'interactions», dans Callon et al. (eds), *Réseau et coordination*, Economica, Paris.

[76] Dalle J.-M., Foray D., Neymann A., 1998, *L'institution brevet dans une*

économie fondée sur la connaissance, Rapport au Commissariat au Plan, 117 p., non publié.

[77] DALLE, J.M., JULLIEN N., 2000, «NT vs. Linux, or some explorations into the economics of Free Software», dans BALLOT & WEISBUCH (eds), *Applications of Simulation to Social Sciences*, Hermès, Paris, pp. 399-416.

[78] DALLE, J.M., JULLIEN N., 2001, «'Libre' software : Turning fads into instutions ?», à paraître dans Research Policy.

[79] DANG NGUYEN G., 1995, *Économie industrielle appliquée*, Vuibert.

[80] DANG NGUYEN, G., 1999, «Économie d'Internet, enjeux et perspectives de recherche», conférence présentée au Colloque d'économie publique, Brest.

[81] DANG NGUYEN G., PETIT P., PHAN D., 1997, «La société de l'information : performances économiques et implication sociales», Communications et Stratégies, n°28, 304 p.

[82] DANG NGUYEN G., PÉNARD T., 1998, «Les accords d'interconnexion dans Internet», Communications et Stratégies, n°32 (numéro spécial Internet : la nouvelle économie d'Internet).

[83] DANG NGUYEN G., PÉNARD T., 1999, «Don et coopération dans Internet : une nouvelle organisation économique ?», Terminal, n°80-81, pp. 107-133.

[84] DANG NGUYEN G., PÉNARD T., 2000, «Les accords d'interconnexion dans les réseaux de télécommunications : des comportements stratégiques aux droits de propriétés», Revue d'économie industrielle, n°92, numéro spécial sur la théorie des contrats, pp. 297-316.

[85] DANG NGUYEN G., PÉNARD, T., 2001, «Interaction et coopération en réseau : un modèle de gratuité», à paraître dans la Revue Économique, numéro spécial économie d'Internet, septembre.

[86] DANG NGUYEN G., PHAN D., 1998, «Learning and the Diffusion of the «Digital» Paradigm in Information and Communication Technology. Toward a new economics of science», Research Policy, 23, p. 487-521.

[87] DANG NGUYEN G., PHAN D., 2000, *Économie des télécommunications et de l'internet*, Economica, 156 p.

[88] Dasgupta P., David D., 1994, «Toward a New Economics of Science», Research Policy, vol. 23.

[89] David P.A. , 1985, «Clio and the economics of QWERTY», American Economic Review (Papers and Proceedings) n°75, pp. 332-337.

[90] David P.A., 1987, «Some new standards for the economics of standardization in the information age», dans Dasgupta, Stoneman eds., *Technology policy and economic performance*, Cambridge University Press, Cambridge Mass., pp. 206-239.

[91] David P.A., 1988, *Putting the past into the future of economics*, Technical Report 533, Institute for Mathematical Studies in the Social Sciences : Stanford University.

[92] David P.A., 1993, *Intellectual property institutions and the Pandas thumb*, CEPR publication n°287, Stanford University.

[93] David P.A., 1995, «Reputation and agency in the historical emergence of the institutions of Open Science», papier presenté au «National Academy of Sciences Colloquium on the Economics of Science and Technology», Beckman Center, UC Irvine, 20-21 octobre.

[94] David P.A., 1998, «Communication norms and the collective cognitive performance of "Invisible Colleges"», in, pp. 115-163.

[95] David P.A., Bunn J., 1988, «The economics of gateway technologies and network evolution», Information Economics and Policy, n°3, pp. 165-202.

[96] David P.A., Foray D., 1994, «Percolation structures, Markov random fields and the economics of EDI standard diffusion», dans Pogorel (ed.), *Global Telecommunication Strategies and Technological Change*, Amsterdam, Elsevier.

[97] David P.A., Foray D., 1995, «Accessing and expanding the science and technology knowledge base», STI Review 16, pp. 13-68.

[98] David P.A., Foray D., Dalle J.-M., 1998, «Marshallian externalities and the emergence and spatial stability of technological enclaves», Economics of Innovation and New Technology 6, pp. 147-182.

[99] David P.A., Greenstein S., 1990, «The economcs of compatibility standards : an introduction to recent research», Economy of Innovation and New Technology, vol. 1, pp. 3-41.

[100] Davis L., North D. C., 1971, *Institutional Change in American Economic Growth*, Cambridge University Press.

[101] De Bandt J., 1995, *Services aux entreprises : informations, produits, richesses*, Economica, Paris.

[102] De Bandt J., 1996, «Coopération, accords interentreprises, concurrence», dans Ravix [1996], pp. 195-229.

[103] De Bandt J., 1998, «Les marchés de services informationnels : quelles garanties pour le client, consommateur ou partenaire ?», revue d'économie industrielle, n°86, 4^e trimestre, pp. 61-84.

[104] Delapierre M., Zimmermann J.-B., 1994, «La nature du produit informatique», Terminal, n°65, pp. 87-104.

[105] Delapierre M., Gérard-Varet L.-A., Zimmermann J.-B., 1980, «Choix publics et normalisation des réseaux informatiques», rapport BNI n°30, décembre.

[106] Delaunay J.-C., Gadray J., 1987, *Les enjeux de la société de service*, Paris, Presses de la Fondation Nationale des Sciences Politiques.

[107] Delaunay J.-C., Gadray J., 1992, *Services in Economic Thought : Three Centuries of Debate*, Dordrecht, Kluwer.

[108] Deroïan F., 1999, «Interactions locales et bifurcations endogènes au sein d'un réseau de type neuronal», document de travail, GREQAM.

[109] Deroïan F., Steyer A., Zimmermann J.-B., 1999, «Influence sociale et apprentissage dans les phénomènes de diffusion de l'innovation», communication aux journées de l'AFSE «Économie de l'Innovation», Sophia Antipolis 20-21 mai.

[110] DiBona C., Ockman S., Stone M., eds., 1999, *Open Sourcesdefault,* O'Reilly, Sebastopol, Calif.

[111] Donsimoni M.-P., Geroski P., Jacquemin A., 1984, «Concentration Indices and Market Power : Two Views», Journal of Industrial Economics, vol 32, pp. 419-434.

[112] Duffy W., Neuberger E., 1976, *Comparative Economic Systems : a Decision-Making Approach*, Boston, Allyn and Bacon Inc.

[113] Durlauf S.N., 1993, «Non-ergodic economic growth», Review of Economic Studies 60, pp. 349-366.

[114] European Information Technology Observatory (EITO), 1999, rapport annuel, EITO, Francfort.

[115] Eurostaf, 1996, *L'industrie européenne des services informatiques. Tome 1 : les marchés*, Eurostaf.

[116] Eymard-Duvernay F., 1989, «Conventions de qualité et formes de coordination», Revue Économique, n°2, mars, pp. 329-359.

[117] Farrell J., 1989, «Standardization and intellectual property», Jurimetrics Journal, printemps.

[118] Farrell J., Saloner G., 1985, «Standardisation, compatibility and innovation», Rand Journal of Economics, n°16, pp. 70-83.

[119] Farrell J., Saloner G., 1988, *The Economics of Converters*, MIT.

[120] Flamm K., 1987 a, *Targeting the Computer*, The Brookings Institution, Washington.

[121] Flamm K., 1987 b, *Creating the Computer : government, industry, and high technology*, The Brookings Institution, Washington.

[122] Foray D., 1989, «Les modèles de compétition technologique. Une revue de littérature», Revue d'économie industrielle, n°48, 2^e trimestre, pp 16-34.

[123] Foray D., 1990, «Exploitation des externalités de réseau versus évolution des normes : les formes d'organisation face au dilemme de l'efficacité dans le domaine des technologies de réseau», Revue d'économie industrielle, n°51, pp. 113-140.

[124] Foray D., 1996, «Diversité, sélection et standardisation», Revue d'économie industrielle, n°75.

[125] Foray D., 1997, «The dynamic implications of increasing returns : technological change and path-dependent inefficiency», International Journal of Industrial Organization 15, pp. 733-752.

[126] Foray D., 2000, *L'économie de la connaissance*, col. Repères, La Découverte, Paris, 124 p.

[127] Foray D., Freeman C. (eds), 1992, *Technologie et richesse des nations*, Economica, 518 p.

[128] Foray D., Hilaire-Perez L., 2000, «The economics of open technology : collective organization and individual claims in the "fabrique lyonnaise" during the old regime», Conference in honor of Paul David, Turin, Mai.

[129] Foray D., Zimmermann J.-B., 2001, «L'économie du logiciel libre : organisation coopérative et incitation à l'innovation», Revue Économique, Volume 52, numéro hors série sur l'économie d'Internet, édité par Brousseau É, Currien N., octobre.

[130] Gadrey J., 1996, réédition de 1992, *L'économie des services*, coll. Repères (113), col. Repères, La Découverte, 125 p.

[131] Gadrey J., 1998, «La caractérisation des biens et des services, d'Adam Smith à Peter Hill : une approche alternative», document de travail.

[132] Gasperoni F., 2001, «Présentation d'ACT Europe», Workshop RNTL sur la nouvelle économie du logiciel, document de travail http ://parmentille.enst-bretagne.fr/~njullien/rntl/workshop1/ACT-Europe_gasperoni.pdf.

[133] Genthon C., 1995, *Croissance et crise de l'industrie informatique mondiale*, Syros.

[134] Genthon C., 1998, «Innovation et changements structurels : l'exemple de l'industrie informatique», Revue d'économie industrielle, n°85, 3^e trimestre, pp 31- 48.

[135] Genthon C., 2000, «Le cas SUN Microsystems», cours donné à l'ENST Bretagne, http ://www-eco.enst-bretagne.fr/Enseignement/2A/1999-2000/EST201/sun/sun00.htm.

[136] Genthon C., 2001, «Le libre et l'industrie des services et logiciels informatiques», Workshop RNTL sur la nouvelle économie du logiciel, document de travail http ://parmentille.enst-bretagne.fr/~njullien/rntl/workshop1/genthon.pdf.

[137] Genthon C., Phan D., 1999, «Les Logiciels Libres : un nouveau modèle ?», Terminal 80/81, L'Harmattan, pp. 167-188.

[138] Gerard-Varet L.A., Zimmermann J.-B., 1985, «Concept de produit informatique et comportement des agents de l'industrie», contribution au colloque «Structures économiques et économétrie», Lyon, 24-24 mai.

[139] Gomez P.-Y., 1994, *Qualité et théorie des conventions*, Economica, 266 p.

[140] Gorgeu A., Mathieu R., 1991, «Les pratiques de livraison en juste à temps en France entre fournisseurs et constructeurs automobiles», CEE, Dossier de recherche, n°31.

[141] Granovetter M., 1985, «Economic Action and Social Structure : the Problem of Embeddedness», American Journal of Sociology, 3, pp 481-510.

[142] Hayek F. A., 1948, *Individualism and Economic Order*, Chicago University Press.

[143] Hayek F. A., 1967, *Studies in Philosophy, Politics and Economics,* Chicago University Press.

[144] Hayek F. A., 1973, *Law, Legislation and Liberty Vol.1 Rules and Order*, Londres, Routeledge & Kegan.

[145] Heckathorn D. D., 1996, «The dynamics and dilemmas of collective action», American Sociological Review, vol. 61, avril, pp. 250-277.

[146] Hill P., 1997, «Tangibles, Intangibles and Services : a New Taxonomy for the Classification of Output», CSLS Conference on Service Productivity, 11-12 avril, Ottawa, 22 p.

[147] Hippel (von) E., 1986, «Lead users : a source of novel product concepts», Management Science, Vol. 32, n°7.

[148] Hippel (von) E., 1988, *The Sources of Innovation*, Oxford University Press, New York.

[149] Horn F., 1999a, «Diversité des informations traitées par des moyens informatiques. Standardisation optimale et acteurs du processus de standardisation», Communications & stratégies, n°33, 1^{er} trimestre, pp. 85-117.

[150] Horn F., 1999b, «L'importance du logiciel libre dans l'amélioration de l'efficience des logiciels», Terminal n°80-81, pp. 119-148.

[151] Horn F., 2000a, «La diversité de l'économie du logiciel : pluralité et dynamique de quatre mondes de production», communication à la conférence internationale «Économie et socio-économie des services», Lille, Juin, 20 p.

[152] Horn F., 2000b, *L'économie du logiciel. Tome 1 : De l'économie de l'informatique à l'économie du logiciel. Tome 2 : De l'économie du logiciel à la socio-économie*

des «mondes de production» des logiciels, Thèse de doctorat d'économie industrielle, Université de Lille I, 570 p.

[153] Irwin D. A., Klenow P. J., 1994, «Learning by Doing : Splillovers in the Semiconductor Industry», Journal of Political Economy, 106(6), pp. 1200-1227.

[154] Janssens A., 1991, *Unix sous tous les angles*, Eyrolles.

[155] Jaruzelski B., Horkan G., Lake R., 2000, *Linux : Fade or Future ?*, Booz-Allen & Hamilton, http ://www.bah.com/greatideas/.

[156] Johnson J. P., [2001], «Some Economics of Open Source Software», présenté à la conférence , GREMAQ.

[157] Johnson W. R., [1985], «The Economics of Copying», Journal of Political Economy.

[158] Jones P., 2000, «Brooks' Law and open source : the more the merrier ? Does the open source development method defy the adage about cooks in the kitchen ?, mai, http ://www-106.ibm.com/developerworks/library/merrier.html.

[159] Jullien N., Phan D., 1999, «From the 'Unix World' towards the 'Linux' community' : An historical co-evolutionary perspective on 'small worlds'», présenté à l'AEPE (novembre 1999).

[160] Jullien N., 1999, «Linux : la convergence du monde Unix et du monde PC», Terminal, n°80/81, pp. 41-70.

[161] Kahin B., 1990, «The Software Patent Crisis», Technology Review, Avril.

[162] Karpik L., 1989, «l'économie de la qualité», revue française de sociologie, vol. 30, pp. 187-210.

[163] Karpik L., 1996, «Dispositifs de confiance et engagements crédibles», sociologie du travail, n°4/96, pp. 527-550.

[164] Katz M.L., Shapiro C., 1985, «Network Externalities, Competition, and Compatibility», American Economic Review, vol. 75 :3, pp. 424-440.

[165] Katz M., Shapiro C., 1986, «Technology Adoption in the Presence of Network Externalities», Journal of Political Economy, vol. 94, n°4, pp. 822-841.

[166] KATZ M., SHAPIRO C., 1994, «Systems Comptetition and Network Effects», Journal of Economic Perspectives, vol. 8, n°4, pp. 93-115.

[167] KAWASAKI S., MCMILLAN J., 1987, «The Design of Contracts : Evidence from Japanese Subcontracting», Journal of The Japanese and International Economies, vol. 1, septembre, pp. 327-349.

[168] KIRMAN A. P., 1992a, «Variety : the Coexistence of Techniques», Revue d'économie industrielle, pp. 62-75.

[169] KIRMAN A. P., 1992b, «Whom or what does the representative individual represent ?», Journal of Economic Perspectives vol. 6, pp. 117-136.

[170] KIRMAN A. P., 1993, «Ants, rationality and recruitment», Quarterly Journal of Economics 111 : 137-156.

[171] KIRMAN A. P., 1998, «The Economy as an Interactive System», dans Arthur W.B., Durlauf S.N. & Lane D (eds), *The Economy as an Evolving Complex System II*, Addison Wesley : New York, pp. 491-531.

[172] KOLLOCK P., 1998, «Design Principles for Online Communities», PC Update, 15(5), juin, pp. 58-60.

[173] KUWABARA K., 2000, «Linux : A Bazaar at the Edge of Chaos», First Monday, vol. 5 n°3,
http ://www.firstmonday.dk/issues/issue5_3/kuwabara/.

[174] LAFFONT J.-J., 1989, *The Economics of uncertainty and information*, Cambridge, MIT Press.

[175] LAFFONT J.-J., 1991, *Fondements de l'économie publique*, Paris, Economica.

[176] LAKHANI K., VON HIPPEL E., 2000, «How Open Source Software Works : Free User to User Assistance», travail de recherche, http ://opensource.mit.edu/online-pap-abst.html.

[177] LANGLOIS R. N., MOWERY D. C, 1996, «The federal Government Role in the Development of the U.S. Software Industrie», dans Mowery [1996], pp. 53-85.

[178] LANCASTER K. J., 1966, «A new approach to consumer theory», Journal of Political Economy, vol 74, avril, pp. 132-157.

[179] Lancaster K. J., 1971, *Consumer Demand : A New Approach*, New York.

[180] Lancaster K. J., 1975, «Optimal Product Differenciation», American Economic Review, vol 65, septembre, pp. 567-585.

[181] Leborgne D., Lipietz A., 1988, «Deux stratégies sociales dans la production des nouveaux espaces économiques», CEPREMAP, n°8911.

[182] Le Nagard E., 1997, «Les stratégies de compatibilité dans les industries de la communication», Communications & Stratégies, n°27, 3^e trimestre, pp. 103-129.

[183] Lerner J., Tirole J., 2000, «The simple economics of Open Source», mimeo, http ://opensource.mit.edu/online-pap-abst.html.

[184] Lévy P., 1992, *De la programmation considérée comme un des beaux-arts,* , La Découverte, 246 p.

[185] Liebowitz S. J., 1985, «Copying and Indirect Appropriability : Photocopying of Journals», Journal of Political Economy.

[186] Liebowitz S. J., Margolis S. E., 1994, «Network Externality : An Uncommon Tragedy», Journal of Economic Perspectives, vol. 8, n°2, printemps, pp. 133-150.

[187] Lorenzi J.H., 1980, «Analyse d'un plan industriel au travers du concept de filière», Annales des mines, janvier.

[188] Lundvall B., 1988, «innovation as an Interactive Process : From User-Producer Interaction to the National System of Innovation», dans Dosi & al., pp. 349-369.

[189] Mairesse J., 1998, «Sur l'économie de la recherche technique», dans Guesnerie, Hartog (eds.), *Des sciences et des techniques : un débat*, EHESS Éditions.

[190] Mangematin V., 1992, «Entre marketing et innovation : le début du processus de compétition technologique», Recherches et Applications en Marketing, vol 7, n°4, pp. 33-54.

[191] Mansfield E., 1961, «Technical change and the rate of imitation», Econometrica, 29, pp. 41-66.

[192] Mansfield E., 1985, «How rapidly does new industrial technology leak out ?», Journal of Industrial Economics, vol. 34, 2.

[193] Manski C. F., 2000, «Economic Analysis of Social Interaction», Journal of Economic Perspectives, vol. 14, n°2, pp. 523-536.

[194] Marwell G., Oliver P., 1993, *The Critical Mass in Collective Action : A Micro-Social Theory*, Cambridge, Cambridge University Press.

[195] Mayère A.,1990, *Pour une économie de l'information*, Éditions du CNRS, Paris

[196] McKelvey M., 1999, «Internet Entrepreneurship (Advances in the Social and Economic Analysis of Technology) Fifth International Conference, 14-16 septembre 1999, Manchester, RU.

[197] McKusick M. K., «Twenty Years of Berkeley Unix Standard», dans DiBona & al. [1999].

[198] Ménard C., 1990, *L'économie des organisations*, Col. Repères, La Découverte, Paris, 129 p.

[199] Mocus A., Fielding R. et Herbsleb J., 2000, «A case study of Open Source software development : the Apache server» actes de la 2000 Int'l Conference on Software Engineering (ICSE2000), Limerick, Ireland, Juin 4-11, pp. 263-272.

[200] Morin M.-L., 1994, «Sous-traitance et relations salariales ; aspects de droit du travail», CEJEE, université de sciences sociales de Toulouse.

[201] Mowery D. C. (ed.), 1996, «The International Computer Software Industry, A comparative Study of Industry Evolution and Structure», Oxford University Press, 324 p.

[202] Netcraft, 1999, «Web Server Survey», http«Institutions», Journal of Economic Perspectives», volume 5, n°1, hiver 1991, p 97-112.

[203] Novos I. E., Waldman M., 1984, «The Effects of Increased Copyright Protection : An Abnalytic Approach», Journal of Political Economics.

[204] Oliver P., Marwell G., Teixeira R., 1985, «A Theory of Critical Mass. I. Interdependence, Group Heterogeneity, and the Production of Collective Action».

[205] Plouraboué F., Steyer A, Zimmermann, J.B., 1998, «Learning Induced Criticality in Consumer's Adotion Pattern : a neural network approach», Economics of Innovations and New technologies, vol. 6, pp. 73-90.

[206] Olson M., 1965, *The logic of Collective Action*, Cambridge, Harvard University Press.

[207] Orbiten Free Software Survey, 2000, avril, http ://orbiten.org/ofss/01.html.

[208] O'Reilly T., 1999, «Lessons from Open-Source Software Development», Communications of the ACM, vol. 42, n°4, pp. 33-37.

[209] Orléan A., 1995, «Bayesian interactions and collective dynamics of opinion : Herd behavior and mimetic contagion», Journal of Economic Behavior and Organization, vol. 28, n°2, pp. 257-274.

[210] Orléan A., 2001, mimeo.

[211] Ousterhout J., 1999, «Free Software Needs Profit», communications of the ACM, Avril 1999, n°4, p 44-45.

[212] Passeron J.-C., 1991, *Le raisonnement sociologique : l'espace non-poppérien du raisonnement naturel*, Nathan (Éssais et recherche), 408 p.

[213] Pigou A. C., 1946, *The Economics of Welfare* (4^e édition), Londres, Macmillan.

[214] Postrel S., 1986, «Bandwagons and the Coordination of Standardized Behavior», Mimeo, MIT

[215] Ravix J.-L. (ed.), 1996, *Coopération entre les entreprises et organisation industrielle*, Paris, CNRS Éditions.

[216] Raymond E. S., 1998 a, «The Cathedral and the Bazaar», http ://www.tuxedo.org/~esr/writings/cathedral-bazaar/.

[217] Raymond E. S., 1998 b, «Homesteading the Noosphere», http ://www.tuxedo.org/~esr/writings/homesteading/.

[218] Raymond E. S., 1999 a, «The Magic Cauldron», http ://www.tuxedo.org/~esr/writings/magic-cauldron/.

[219] Raymond E. S., 1999 b, «A brief history of hackerdoom», http ://www.oreilly.com/catalog/opensources/book/raymond.html, édité dans Di Bona & al. [1999].

[220] Raymond E. S., 1999 c, *The Cathedral & the Bazaar ; Musing on Linux and Open Source by Accidental Revolutionary*, O'Reilly.

[221] Richardson G. B., 1972, «The Organization of Industry», Economic Journal, n°82, pp. 883-896.

[222] Richardson G. B., 1997, «Economic analysis, public policy and the software industry», DRUID Working Paper, n°97-4, avril, dans *The Economics of Imperfect Knowledge - Collected papers of G.B. Richardson*, Edward Elgar.

[223] Robbins S. P., 1987, *Organization Theory*, Englewood Cliffs, NJ, Prentice Hall, 2^e éd.

[224] Romer P., 1993, «The economics of new ideas and new goods», proceedings du congrès annuel (1992) de la Banque Mondiale «Conference on Development Economics», Washington D. C., Banque Mondiale.

[225] Rosenberg N., 1982, *Inside the black box : technology and economics*, Cambridge University Press.

[226] Sahal D., 1981, *Patterns of technological innovations*, Addison Wesley.

[227] Sako M., 1992, *Prices, Quality and Trust : Inter-firm Relations in Britain an Japan*, Cambridge, Cambridge Univ. Press.

[228] Salais S., Storper M., 1993, *Les mondes de production : enquête sur l'identité économique de la France*, édition de l'École des Hautes Études en Sciences Sociales (Civilisations et sociétés), 467 p.

[229] Saloner G., Steinmueller W. E., 1996, «Demand for Computer Products ans Services by Large European Organizations», rapport à la Commission Européenne, 29 p.

[230] Samuelson P. A., 1954, «the Pure Theory of Public Expenditure», Review of Economics and Statistics, n°36, pp. 387-389.

[231] Samuelson P. A., 1955, «A Diagrammatic Exposition of a Thoery of Public Expenditure», Review of Economics and Statistics, n°37, pp. 350-356.

[232] Schumpeter J. A., 1912, *The Theory of Economic Development*, Cambridge (1936), Harvard University Press.

[233] Scotchmer S., 1991, «Standing on the shoulders of giants : cumulative research and the patent law», Journal of Economic Perspectives, vol. 5, n°1, hiver, pp. 29-41.

[234] Smith M., Kollock P., (eds), 1999, *Communities in Cyberspace*, Routledge, London.

[235] Stallman R. M., 1999, «The GNU Project», dans DiBona & al. [1999].

[236] Stanback T., 1979, *Understanding the Service Economy*, John Hopkins University Press, Baltimore.

[237] Steinmueller W. E., 1999, *Networked knowledge and knowledge-based economies*, telematica Institut, Delft, février.

[238] Steyer A., Zimmermann J.-B., 1996, «Externalités de réseau et adoption d'un standard dans une structure résiliaire», Revue d'économie industrielle, n°76, 2^e trimestre, pp. 67-90.

[239] Storch H., 1815, traduction française 1823, *Cours d'économie politique*, Paris, Aillaud, 2 vol.

[240] Tellier A., 1996, «Exemples d'action stratégique dans les compétitions technologiques à externalité de réseau», document de travail, IUT de Tours.

[241] Thumm N., 2000, «Breveter pour protéger : un bilan s'impose», IPTS n°43, avril.

[242] Tirole J., 1995, *Théorie de l'organisation industrielle*, Economica, traduit de Tirole, 1988, *The Theory of Industrial Organization*, MIT Press.

[243] Torvalds L., 1999, «The Linux Edge», Communications of the ACM, vol. 42, n°4, pp. 38-39.

[244] Tuomi I., 2001, «Internet, Innovation, and Open Source : Actors in the Network», First Monday, janvier, vol 6, n°1, http ://www.firstmonday.dk/issues/issue6_1/tuomi/index.html.

[245] Turkle S, 1986, *Les enfants de l'ordinateur*, Denoël (Présence de la science), 318 p.

[246] Vanberg V., 1986, «Spontaneous Market Order and Social Rules, a Critical Examination of F. A. Hayek's Theory of Cultural Evolution», Economics and Philosophy, pp. 75-100.

[247] Van Alstyne M., Brynjolfsson E., 1997, «The Net Effect : Modeling and Measuring the Integration of Electronic Communities», Telecommunications Policy Research Conference (TPRC), Washington D.C., September 26-28.

[248] Vicente J., 2000, Interactions économiques et coexistence spaciale des modes de coordination, Thèse de doctorat en sciences économiques, Université de Toulouse 1, 231 p.

[249] Watts D. J., 1999, *Small Worlds : The Dynamics of Networks between Order and Randomness*, Princetown University Press,

[250] Wheeler D. A., 2001, *Why Open Source Software / Free Software (OSS/FS) ? Look at the Numbers !*, http ://www.dwheeler.com/.

[251] Williamson O., 1975, *Market and Hierarchies : Analysis and Anti-Trust Implications*, Free Press, New-York.

[252] Williamson O., 1985, *The Economic Institutions of Capitalism*, Free Press, New-York.

[253] Williamson O., 1991, «Comparative Economic Organization : the Analysis of Discrete Alternative», Administrative Science Quaterly, vol. 36, pp. 269-296.

[254] Williamson O., 1993, «Calculativeness, Trust and Economic Organization», Journal of Law and Economics, n°36, avril, pp. 453-486.

[255] Zimmermann J.-B., 1989, «groupes industriels et grappes technologiques», Revue d'économie industrielle, vol. 47, n°5, 1^{er} trimestre.

[256] Zimmermann J.-B., 1995a, «L'industrie du logiciel : de la protection à la normalisation», dans Basle, Dufourt, Héraud, Perrin (eds), *Changement institutionnel et changement technologique Évaluation, droits de propriété intellectuelle, système national d'innovation*, CNRS Éditions, pp. 195-207.

[257] Zimmermann J.-B., 1995b, «Le concept de grappes technologiques. Un cadre formel», Revue économique, vol. 46, n°5, septembre, pp. 1263-1295.

[258] Zimmermann J.-B., 1998, «Un régime de droit d'auteur : la propriété intellectuelle du logiciel», Réseaux, n°88-89, pp. 91-106.

[259] Zimmermann J.-B., 1999, «Logiciel et propriété intellectuelle : du Copyright au Copyleft», Terminal n°80-81, pp. 151-166.

Abstract

LIBRE software, i.e. software produced and redistributed freely, with the source code and the right to modify and redistribute it in the same term and conditions, has recently become a major interest for both the software industry and the economic theory. Our aim is to propose an economic analysis of this phenomenon.

Although this system of production seems to be anachronistic considering the progressive merchandization of the software production, we defend the idea it is an efficient and sustainable economic organization. The technological progress, the interconnection of the computers (via Internet) has increased the users needs and demands for compatible, perennial and personalized solutions. The openness and the modularity of such software allow the production of norms and grant this compatibility and a minimum level of quality inciting the producers to use these solutions. Because of the increasing importance of the services, they are also incited to co-produce, to signal their competency and to maintain the level of their expertise on the way the programs they propose and adapt work.

Comparing the Libre and Proprietorial organization, we show that the first is probably more efficient because of a better management of incremental innovation and because it diminish the inefficiency due to the control of standards by monopolies the proprietarial organization engenders. But, because of standardization effects, the diffusion of free software will be slow. It should first concern companies and technical programs and after end-users and the program they use. The survey we have conducted on French companies selling offers using free software shows that, even if this first stage is at its beginning, it has started, with the characteristics expected. We analyse the possible form of the second stage using a model of technology competition, based on Dalle [1995] and Dalle & Jullien [2000]'s works.

This drives us to the following conclusion : as far as software is concerned, the technological evolution revitalizes the cooperative production of knowledge and allow a new articulation between this non-market production and a market valorization. More than a private/public collaboration, the Free software organization creates a sustainable mechanism of market/non-market collaboration which could be Pareto-superior to the simple market one.

Résumé

LE «Libre», c'est-à-dire la production de logiciels distribués avec leur code source et avec l'autorisation de les modifier et de les redistribuer librement (à condition que cette redistribution soit faite aux mêmes conditions), est en train de devenir un des modèles économiques importants de l'industrie informatique. L'objectif de ce travail est d'en proposer une analyse économique.

Bien que ce système de production semble aller à contre courant de la marchandisation progressive de la production du logiciel, nous défendons l'idée que c'est une organisation économique efficace et pérenne. En effet, les progrès technologiques actuel, l'arrivée des réseaux (et notamment d'Internet) transforment la demande, qui est de plus en plus sensible à ces problèmes de compatibilité, de pérennité et de personnalisation des solutions informatique. Or, le caractère ouvert et modulaire de la production Libre favorise le développement de normes, garantissant l'interopérabilité et un niveau minimum de qualité pour ces logiciels, ce qui incite les producteurs à les utiliser. Parce que la relation de service devient de plus en plus importante dans l'industrie informatique, ils sont incités à contribuer à leur production, afin de signaler la qualité de leur travail et de maintenir leur niveau d'expertise sur le fonctionnement des logiciels qu'ils adaptent.

Nous comparons alors le Libre à l'organisation de production propriétaire et nous montrons qu'elle est sans doute plus efficace parce qu'elle gère mieux l'innovation incrémentale et donc diminue l'inefficacité du contrôle monopolistique des standards qu'engendre le système propriétaire. Mais à cause de ces effets de standardisation, la diffusion des logiciels libres et donc du Libre sera lente, concernera d'abord les entreprises et les logiciels techniques, ensuite seulement l'ensemble des utilisateurs et les logiciels qu'ils utilisent (système d'exploitation, logiciels de traitement de l'information personnelle, etc.) L'enquête que nous avons menée auprès des entreprises françaises utilisant le libre pour construire leurs offres commerciales semble indiqué que, même si elle est à ses débuts, la première phase a commencé et a les caractéristiques attendues. Nous analysons les caractéristiques possibles de la deuxième phase grâce à un modèle de concurrence technologique, inspiré des travaux Dalle [1995] et Dalle & Jullien [2000].

La conclusion que nous tirerons de ce travail est la suivante : dans le cas du logiciel, l'évolution technologique revitalise la production coopérative de connaissances et permet

une nouvelle articulation entre cette production non marchande et une valorisation marchande. Plus qu'une collaboration public/privé, qui existait déjà, le Libre engendre un mécanisme, auto-entretenu, de collaboration marchand/non marchand, qui peut se montrer supérieur (au sens de Pareto), aux simples mécanismes de marché.