RTFM! La chimie computationelle: une communauté scientifique loin du libre

Alexandre Hocquet

▶ **To cite this version:**

Alexandre Hocquet. RTFM! La chimie computationelle: une communauté scientifique loin du libre. Histoires et Cultures du Libre., Framabook, pp.487, 2013. halshs-00827881

HAL Id: halshs-00827881
https://shs.hal.science/halshs-00827881

Submitted on 29 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sous la direction de :
Camille Paloque-Berges, Christophe Masutti

Histoires et cultures du Libre

Des logiciels partagés aux licences échangées

CC BY

RTFM ! La chimie computationnelle : une communauté scientifique loin du libre

Alexandre HOCQUET

1. RTFM ! What FM ?

Read The Fucking Manual ! Cet acronyme geek est ce que l'on répond, excédé, lorsqu'une personne pose une question sans s'être donné la peine de chercher. Il est particulièrement courant sur les forums ou les listes de diffusion dans lesquelles on parle de logiciel. Mais le manuel (ou son absence) représente aussi autre chose. Aux débuts de l'histoire du logiciel, dans les années cinquante et soixante, une grande partie du développement était réalisée dans des laboratoires universitaires et des entreprises par des scientifiques et des ingénieurs. Il était normal à l'époque que le logiciel fasse partie intégrante de l'ordinateur pour lequel il avait été conçu. La portabilité des applications entre deux machines était quelque chose de très rare, et l'idée d'échanger, de vendre ou encore l'octroi de licences de logiciels ne venait jamais à l'esprit des scientifiques et ingénieurs, universitaires ou industriels, qui les concevaient. Ce n'est qu'à partir des années soixante que le logiciel est devenu un produit qui pouvait être acheté séparément de l'ordinateur lui-même. En 1967, l'expression *software crisis*

(crise du logiciel) est apparue pour la première fois dans le milieu des développeurs pour caractériser les difficultés rencontrées par l'industrie du logiciel. De manière complètement opposée à l'évolution spectaculaire des performances et du coût du matériel, les développements de logiciels étaient de plus en plus longs, nécessitaient de plus en plus de main d'œuvre, coûtaient de plus en plus cher. Les erreurs (les bogues/bugs) étaient de plus en plus difficiles à localiser et à corriger étant donnée la complexité croissante du code et des langages, et les révisions des versions avaient de plus en plus de difficultés à suivre l'évolution du matériel. La situation de crise ne s'est en fait jamais atténuée et la *software crisis* est, depuis lors, devenue un art de vivre pour les développeurs plutôt qu'un épisode passager [1].

Avec la popularisation de l'ordinateur, il est arrivé un moment où les logiciels ont dû être utilisés par des personnes différentes de ceux qui les concevaient. La question de la documentation, ou plus généralement de l'adéquation entre les pratiques des utilisateurs et les pratiques des développeurs (le *fucking manual*) est aussi un emblème des problèmes jamais résolus de la crise du logiciel : il s'est donc avéré que les plus grandes tensions dans l'industrie informatique (mais aussi dans l'histoire du calcul informatique en tant qu'activité de recherche scientifique) étaient liées à la question du logiciel. Depuis la crise du logiciel des années soixante, la maintenance, ou le service après vente, ou encore la gestion de la communauté, selon le point de vue duquel on se place, est un des points critiques dans les difficultés de production et de distribution de logiciels ainsi qu'une des sources de tensions entre développeurs et utilisateurs. Le manuel, ou, plus largement, la question de savoir comment utiliser un logiciel écrit par quelqu'un d'autre, est une de ces tensions. RTFM !, c'est aussi l'illustration de cette tension.

Dans le cadre de la recherche scientifique, ces tensions existent aussi et sont devenues particulièrement importantes quand l'ordinateur est devenu, dans le laboratoire, un objet quotidien, un objet de masse. Les idées et les pratiques du logiciel libre sont nées de ces tensions comme le suggère l'anecdote de Richard Stallman et ses déboires avec les pilotes d'imprimante Xerox. Mais ces tensions ont aussi parfois conduit les cher-

1. Nathan L. ENSMENGER, *The Computer Boys Take Over : Computers, Programmers, and the Politics of Technical Expertise*, Cambridge : MIT Press, 2010.

cheurs vers des voies complètement différentes du libre, montrant ainsi que monde académique et développement du libre ne font pas forcément bon ménage. C'est le cas de la chimie computationnelle [1].

2 La chimie computationnelle

Qu'est-ce que la chimie computationnelle ? Il s'agit du domaine scientifique défini comme l'utilisation des ordinateurs pour résoudre les problèmes de chimie. Si on cherche les mots *computational chemistry* dans Google Images, on obtient des images colorées de molécules ou de propriétés moléculaires littéralement en train de sortir hors de l'écran. C'est qu'il s'agit d'un domaine de la recherche scientifique pour lequel l'esthétique des représentations de molécules par les terminaux graphiques est souvent utilisé pour donner une image moderne et hautement technologique de la recherche.

D'une part, la chimie computationnelle remonte à l'équation de Schrödinger [2]. La « chimie théorique » ou « chimie quantique » consistait dans les années trente et quarante à appliquer le formalisme de la mécanique quantique pour comprendre la structure de systèmes simplissimes (comme la molécule d'hydrogène), en résolvant des équations sur un tableau noir ou une feuille de papier. Avec l'avènement des premiers ordinateurs, les calculs de chimie quantique ont pu être effectués sur les rares heures de calcul disponibles des superordinateurs. À l'époque, il s'agissait d'objets rares et chers, emblèmes de la *big science*, financés essentiellement par des crédits fédéraux ou nationaux. Sur ces superordinateurs, le temps de calcul disponible était partagé entre les différents utilisateurs. Les chimistes théoriciens n'étaient alors que des acteurs mineurs dans ce do-

1. « Chimie computationnelle » est un néologisme, traduction littérale de *computational chemistry*. Il est employé au Canada mais peu en France où l'on trouve parfois les termes « chimie informatique » ou « chimie numérique ». Mais les mots informatique ou numérique sont aussi liés à des époques. L'adjectif *computational* sied parfaitement aux années quatre-vingt et quatre-vingt-dix et au monde anglo-saxon, époque et lieu où se déroule notre histoire.

2. L'équation de Schrödinger est l'équation de base de la mécanique quantique. Datant de 1927, elle représente ce que doit résoudre un calcul dans le cadre d'un formalisme mathématique quantique, infiniment plus complexe pour le traitement par l'ordinateur que le formalisme mathématique de la physique classique.

maine, les calculs de chimie étant considérés comme moins stratégiques que la balistique ou l'aérodynamique, par exemple.

Par ailleurs, dans les années soixante et soixante-dix, une conception complètement différente de la façon de modéliser les molécules a émergé. Cette vision ressemblait à celle des modèles moléculaires tridimensionnels que les chimistes manipulent en classe ou au laboratoire et était basée sur un modèle énergétique de potentiel harmonique similaire à des ressorts. Elle a été nommée « mécanique moléculaire », en référence à une vision newtonienne des mouvements des molécules. Il n'y avait ni électrons ni réactivité dans ce modèle, mais la possibilité d'exécuter des calculs sur des molécules plus grosses (et donc plus utiles, comme par exemple les protéines) explique son succès immédiat quand les premiers terminaux graphiques ont permis aux scientifiques d'afficher effectivement des molécules sur l'écran dans les années soixante-dix.

Les modèles mathématiques de la chimie quantique, basée sur la mécanique quantique, comportent quant à eux des procédures mathématiques trop compliquées pour être calculées à la main, y compris pour les molécules les plus simples. L'apparition des premiers ordinateurs a laissé entrevoir l'espoir de résoudre l'équation de Schrödinger avec des méthodes numériques, de manière approchée. Mais il est vite devenu clair que la demande en puissance de calcul (exponentielle en fonction du nombre d'atomes des molécules) condamnait la chimie quantique à ne s'intéresser qu'aux petites molécules, malgré les perspectives inouïes de développement des performances du matériel informatique. La mécanique moléculaire, au contraire, s'appuie sur la physique newtonienne simple, permettant des calculs numériques sur les molécules les plus intéressantes, et s'appuie sur une façon de définir et de représenter les molécules familière aux chimistes. Le modèle ne repose pas sur des bases théoriques solides (contrairement à la chimie quantique). Il est, au contraire, une définition simple des liaisons et des atomes [1]. Il s'agit donc de deux concepts

1. En mécanique moléculaire, les liaisons sont considérées énergétiquement comme des potentiels harmoniques (c'est-à-dire des boules et des ressorts, inspirés par l'idée de mouvements moléculaires internes de la spectroscopie infra rouge). La mécanique moléculaire est aussi une autre façon de définir ce qu'est une molécule : Contrairement à la description de la chimie quantique pour laquelle une molécule est une collection de particules, négativement (électrons) ou positivement (noyaux) chargées, la mécanique moléculaire correspond à la vision classique des chimistes pour décrire les molécules (atomes liés par des liaisons) et

complètement distincts de modélisation, deux théories différentes. Avec l'avènement du terminal graphique, la possibilité de voir la structure tridimensionnelle des molécules sur un écran explique le succès de la mécanique moléculaire. Il a été immédiat, non seulement pour la chimie en tant que science computationnelle (à partir du moment où l'ordinateur a été accessible aux chimistes), mais aussi pour les constructeurs de matériel qui voyaient dans cette activité de recherche une publicité moderne et scientifique.

En ce qui concerne le matériel, la chimie computationnelle est donc un domaine scientifique qui a commencé comme un client mineur des ressources de calcul des superordinateurs des années soixante, mais s'est rapidement développé pour devenir un acteur dominant dans le domaine du calcul scientifique lorsque les terminaux graphiques sont devenus disponibles vingt ans plus tard. Lorsque l'ordinateur personnel (le PC) et les stations de travail ont popularisé l'informatique dans les laboratoires de recherche, une nouvelle ère de la « modélisation de bureau » (*desktop modeling*) a coïncidé, pour les chimistes computationnels, avec une demande croissante de logiciels de modélisation de la part de l'industrie pharmaceutique, et un investissement dans ce marché potentiel de la part des fabricants de matériel, y compris des géants comme IBM.

Par ailleurs, pendant les années quatre-vingt, les années Reagan, les universités américaines, bon gré mal gré, lançaient des projets de « transfert de technologie » (*technology transfer*). Les logiciels produits dans les universités commençaient à être considérés comme des revenus potentiels par ces dernières, à travers leur politique de brevet ou de licences exclusives portant sur les résultats de leurs activités de recherche. Le logiciel de modélisation moléculaire [1] est devenu l'un des ces objets susceptibles de rapporter de l'argent et ainsi est devenu un objet central dans la communauté des chimistes computationnels. Pour la première fois, les scienti-

inventer un modèle mathématique simple compatible avec cette définition d'une molécule. La complexité de la mécanique moléculaire réside plutôt dans sa difficulté à paramétrer des dizaines de types d'atomes, des centaines de liaisons, et des milliers de paramètres créés pour ajuster le modèle mathématique simple aux résultats expérimentaux de la structure moléculaire. Son inconvénient est l'absence totale d'électrons dans la définition de la molécule et donc de la simple idée de réactivité (ce qui est un gros manque pour un chimiste)

1. Le logiciel de « modélisation moléculaire » est le logiciel conçu par et pour les chimistes computationnels. Il englobe un modèle (de chimie quantique et/ou de mécanique moléculaire) et permet de calculer la structure et/ou les propriétés de molécules.

fiques qui concevaient des logiciels de modélisation moléculaire n'étaient plus les mêmes que ceux qui accomplissaient les calculs. Deux catégories distinctes de scientifiques concernés par la chimie computationnelle se formaient : les développeurs d'un côté, les utilisateurs de l'autre. La question « comment transférer le logiciel de modélisation moléculaire du développeur vers l'utilisateur » est ainsi née… et a suscité beaucoup de tensions dans la communauté.

3 Le terminal graphique

L'impact matériel décisif sur cette redéfinition des rôles a été l'avènement du terminal graphique dans les années soixante-dix. La capacité de voir une molécule en trois dimensions sur un écran au lieu de coordonnées sur une feuille de papier d'imprimante s'est avérée essentielle à la réussite de la chimie computationnelle en tant que discipline scientifique (et en particulier des modèles de mécanique moléculaire qui pouvaient représenter de plus grandes et plus belles molécules). En tant que discipline scientifique, mais aussi en tant que promoteur de ce nouveau matériel : les belles images de molécules sur un écran représentaient à l'époque la publicité la plus esthétique pour une utilisation scientifique des écrans graphiques de haute résolution. D'un côté, la chimie computationnelle suscitait l'intérêt des décisionnaires de recherche et développement de l'industrie pharmaceutique, qui voyaient dans la modélisation la « promesse technologique » qui allait faire baisser les coûts énormes de découverte et de mise sur le marché de nouveaux médicaments. De l'autre, les fabricants de matériel voyaient dans ce nouveau domaine scientifique un allié pour vendre la nouvelle technologie.

Au début des années quatre-vingt, la chimie computationnelle n'était plus un acteur mineur des ressources informatiques. Le tiers des heures des centres de calcul américains était consacré aux calculs en chimie computationnelle. Les chimistes computationnels étaient devenus importants, et étaient ainsi concernés par le monde commercial et industriel, et ce d'une triple façon.

D'abord, les chimistes computationnels étaient impliqués dans l'industrie de la fabrication d'ordinateurs. Les géants comme IBM ont vu dans

la chimie computationnelle une force majeure dans le domaine du calcul intensif, et IBM a activement financé les heures de calcul de groupes de recherche en chimie, mais aussi a investi une partie substantielle de ses efforts de R&D à l'élaboration d' *interactive molecular graphics* [1]. Dans le même ordre d'idée, IBM a également été un important investisseur dans l'un des grands éditeurs de logiciels de modélisation moléculaire, Polygen. Mais les chimistes computationnels ont également été parmi les meilleurs clients (et les meilleurs publicitaires) pour les fabricants de stations de travail aux puissantes capacités graphiques tels que Silicon Graphics et bien d'autres.

Ensuite, un marché potentiel pour la modélisation par la mécanique moléculaire a été pressenti par les chimistes du secteur pharmaceutique, ce qui a impliqué la création de quelques entreprises de logiciels à fortes ressources financières dans les années 1980. L'intérêt croissant de l'industrie pharmaceutique pour cette « promesse technologique » a mobilisé les domaines de la chimie déjà très liés à la pharmacie. En écho, les différents choix stratégiques des développeurs de logiciels en matière de politique de brevet ou de licences exclusives ont été influencés par les pratiques traditionnelles de l'industrie pharmaceutique, connue pour être l'une des plus secrètes et compétitives.

Enfin, ils étaient concernés par le logiciel vu comme une entreprise commerciale. Dans les années 1980 et 1990, les *success stories* des entreprises du domaine de l'ordinateur personnel (comme Microsoft et Autodesk) étaient autant d'exemples incitant les développeurs mais aussi et surtout les administrations des universités à se lancer dans l'exploitation commerciale de leurs logiciels.

Dans le même temps, les informaticiens qui ont imaginé les bases du logiciel libre et d'un Internet neutre, dans le monde académique ou non, représentaient une communauté scientifique de pairs, de coopération entre spécialistes, partageant des valeurs communes et séparée du reste de la société. Au tournant des années soixante-dix et quatre-vingt, cette communauté était institutionnellement, technologiquement, géographiquement proche des endroits où s'est construite simultanément la communauté

1. Eric FRANCOEUR et Jérôme SEGAL, « From model kits to interactive computer graphics », dans : *Models : The Third Dimension of Science*, sous la dir. de Soraya de CHADAREVIAN et Nick HOPWOOD, Palo Alto, 2004.

des chimistes computationnels. Pourtant, cette « république des informaticiens », décrite par le sociologue Patrice Flichy, était immergée dans un contexte économique et politique complètement différent, et possédait un « imaginaire » tout aussi différent de celui en vigueur en chimie [1].

4. Cinq pratiques caractéristiques du Libre

Alors que les développeurs de logiciels en chimie computationnelle cherchaient différentes voies pour distribuer leurs produits, la communauté scientifique à laquelle ils appartenaient n'avait que peu de relations (professionnelles ou militantes) avec les acteurs du mouvement du logiciel libre naissant. Le travail de l'anthropologue Christopher Kelty a été de s'intéresser à la question de savoir « comment » le mouvement/phénomène du logiciel libre marche, plutôt que savoir « pourquoi ». C'est-à-dire qu'il s'est intéressé aux pratiques des gens impliqués dans le logiciel libre, plutôt que de tenter d'expliquer son (éventuel) succès. En référence au concept de *public sphere* d'Habermas (qu'on peut traduire par espace public), Kelty propose l'idée de public récursif (*recursive public*) pour définir ce que sont ces pratiques caractéristiques [2]. Récursif, parce que ce public n'a pas seulement un discours sur la technologie : il utilise cette technologie pour produire son action politique, ou son activité en général, et son discours même. Récursif, parce que cette technologie dépend de différentes couches successives imbriquées, toutes concernées d'une manière ou d'une autre par le libre : logiciels d'application, compilateurs, langages, systèmes d'exploitation, matériel, protocoles Internet…

Kelty affirme qu'il y a cinq pratiques fondamentales qui définissent le public récursif du logiciel libre. Il s'agit bien de pratiques (et non de règles ou de normes), puisque d'une part, le travail de Kelty a consisté à observer en ethnologue les acteurs du logiciel libre dans leur milieu, et d'autre part, parce que Kelty estime plus pertinent, pour décrire les « gens du libre »,

1. Patrice FLICHY, « Internet ou la communauté scientifique idéale », dans : *Réseaux* 17.97 (1999), p. 77–120.

2. Christopher KELTY, *Two bits : the cultural significance of free software*, Durham : Duke University Press, 2008.

de recourir à ce qu'ils ont en commun dans ce qu'ils font plutôt que dans ce qu'ils disent [1].

Première pratique : partager le code source. La première pratique, inscrite comme principe de base par les acteurs du libre eux-mêmes, consiste à développer une communauté en partageant, donc en rendant ouvert le code source. Le fonctionnement du logiciel est ainsi accessible à tous.

Deuxième pratique : conceptualiser des systèmes ouverts. Pour construire un public récursif, le maximum d'objets technologiques doivent être ouverts, de l'ordinateur à l'Internet en passant par toutes les couches logicielles. Toutes ces technologies appartenant à des domaines différents sont rendues ouvertes chacune à leur façon (TCP-IP, WWW,...), c'est-à-dire qu'il faut inventer une idée de ce qu'est « être ouvert » dans des domaines différents, pour que la communauté puisse évoluer dans un milieu adapté au public récursif.

Troisième pratique : écrire des licences. Pour chaque système ouvert conceptualisé, la façon de le partager et de l'inclure dans une des couches technologiques est d'en écrire la licence. Écrire des licences matérialise les conditions selon lesquelles les objets ouverts circuleront dans la communauté.

Quatrième pratique : coordonner des collaborations. Comme le bazar décrit par Eric S. Raymond, les communautés autour du logiciel libre ne sont pas caractérisées par leur organisation hiérarchique [2]. Pour autant, mettre en œuvre des pratiques collaboratives fait partie de l'essentiel des mouvements du Libre. Les projets libres ne peuvent être viables que si la collaboration entre les acteurs est coordonnée. Cela n'implique pas forcément des organisations, mais pour le moins de l'adaptabilité.

1. Cette description des pratiques définies par Kelty est inspirée de celle faite par Andreas Lloyd qu'on peut trouver sur son blog : http://andreaslloyd.dk/2008/11/bit-by-bit. Par ailleurs, un autre chapitre du présent livre traduit un des chapitres du livre de Kelty (*Two Bits*).

2. Eric S. RAYMOND, *The Cathedral and the Bazaar*, Sebastopol, CA : O'Reilly, 2001, URL : http://www.catb.org/esr/writings/homesteading/.

Cinquième pratique : fomenter des mouvements. Les communautés du libre partagent aussi en commun la motivation de créer à partir de leur projet une façon de voir le libre. Construire une narration, voire être prosélyte, sont aussi des pratiques caractéristiques du libre.

La communauté des chimistes computationnels a quelque chose d'un public récursif : c'est une communauté dont l'objet technologique (le logiciel) est imbriqué au cœur même de son existence dans plusieurs couches technologiques successives, essentielles à la définition de l'activité de la communauté. Le logiciel de modélisation moléculaire est d'abord lié à l'ordinateur et la transformation de ce dernier d'objet rare et réservé à des experts en un objet accessible et modulable est essentielle. Le logiciel est particulièrement lié au terminal graphique, non seulement en tant qu'objet technologique définissant les caractéristiques du logiciel, mais aussi en tant qu'objet technologique façonnant la communauté elle même par ses implications économiques et commerciales.

Pour autant, la communauté des chimistes computationnels, par opposition à la « république des informaticiens » est une communauté pour laquelle les cinq pratiques de Kelty ont du mal à faire l'unanimité. On peut même avancer que l'ensemble des cinq pratiques caractéristiques, qui soude la communauté du libre, divise la communauté des chimistes computationnels. Qu'il s'agisse de partager le code source, de conceptualiser l'ouverture des systèmes, d'écrire des licences, de coordonner des collaborations ou de fomenter des mouvements, aucun consensus ne s'est dégagé et qu'au contraire, chacun de ces cinq sujets est source de tensions.

5. Trois contraintes à la pérennité du Libre

Dans une étude sociologique sur les logiciels économétriques, Gambardella et Hall avancent que « le logiciel (scientifique) librement partagé » peine à s'imposer comme pratique de distribution dans certaines conditions bien précises : lorsque les attentes de profits pécuniaires sont élevées, lorsque que la demande du marché potentiel d'utilisateurs est importante,

et quand les normes incitant au partage sont faibles ou difficiles à appliquer [1].

Pour que le partage libre de logiciels soit un succès, il faut qu'existe une sorte de coordination (comme la Free Software Foundation), ou ce que Kelty appelle « fomenter un mouvement » dans son approche anthropologique. Au contraire, dans la communauté de la chimie computationnelle, la variété des licences imaginées par les différents développeurs (ou leurs administrations), le faible nombre de références à la FSF ou le faible nombre de recours à ses licences comme la GPL exprime les tensions et l'absence de consensus.

Un profit potentiel élevé et une demande forte du marché sont les deux autres ingrédients de la recette pour faire obstacle au « libre partage de logiciels ». Les liens étroits entre la chimie computationnelle et l'industrie des constructeurs d'ordinateurs, d'une part, et avec l'industrie pharmaceutique, d'autre part, correspondent tout à fait à ces deux critères : d'un côté les constructeurs ont vu dans le logiciel de modélisation moléculaire un objet pouvant rapporter gros en ventes de matériel (particulièrement pour populariser les terminaux graphiques). Accessoirement, le logiciel de modélisation moléculaire pouvait rapporter gros aux développeurs de logiciels eux-mêmes. D'un autre côté, dans l'industrie pharmaceutique, la demande du marché était importante. L'industrie pharmaceutique était prête à payer cher, appartenait à une culture de recherche et développement très versée dans le secret ou dans les brevets (dans le but de protéger leurs découvertes), et donc avait une idée *a priori* du logiciel en tant qu'objet scientifique proche de cette culture [2]. Les incitations pour les développeurs à embrasser l'esprit d'entreprise, mais aussi d'avoir recours à des licences fermées, étaient ainsi élevées. À cela, on peut ajouter la tendance des universités américaines, à travers leurs administrations de « transfert de technologie » ou leurs « bureaux des brevets » à pousser vers la commercialisation. Ces incitations se produisaient parfois en harmonie avec les projets des équipes de recherche qui développaient des logiciels,

1. Alfonso GAMBARDELLA et Bronwyn H. HALL, « Proprietary versus public domain licensing of software and research products », dans : *Research Policy* 35.*6* (2006), p. 875–892.

2. D. C. MOWERY et al., « The growth of patenting and licensing by US universities : an assessment of the effects of the Bayh-Dole act of 1980 », dans : *Research policy* 30.*1* (2001), p. 99–119.

parfois contre leur gré, que ce soit contre la volonté de certains universitaires de faire du libre, ou en leur imposant des choix de licences pas forcément pertinents.

Le contexte politique, académique et économique de la communauté, mis en lumière par l'étude de Gambardella et Hall éclaire pourquoi il a été si difficile pour les chimistes computationnels de « fomenter des mouvements » au sens de Kelty : les normes du libre, telles que celles de la FSF, ont besoin d'un contexte académique adéquat pour pouvoir se répandre et s'imposer dans la communauté scientifique. Mais le logiciel et sa conceptualisation, et donc les pratiques de la communauté, sont aussi intimement liés à l'ordinateur lui-même, au matériel.

6. La rupture du *desktop modeling*

À l'époque où l'informatique consistait à attendre du temps de calcul sur un super-ordinateur, « la manipulation » ou l'exploration des modèles (*model tweaking*), c'est-à-dire l'exécution itérative de versions de modèles modifiées de manière mineure à chaque fois, était impraticable. Le super-ordinateur n'était accessible qu'à travers de nombreux intermédiaires.

L'époque suivante [1], selon les catégories de Lenhard et Johnson [2] est l'époque de la disponibilité des machines. Des ordinateurs de plus en plus petits avec des puissances de calcul de plus en plus grandes, envahissaient non seulement le laboratoire, mais aussi envahissaient le bureau même des scientifiques, y compris celui de ceux qui n'étaient pas spécialistes en informatique. L'informatique de bureau, telle que popularisée par l'ordinateur personnel d'IBM (le *PC*) et plus encore par les stations de travail (*workstations*) avec leurs meilleures capacités graphiques, pouvaient finalement transformer le travail des chimistes computationnels. Ils et elles leur donnaient le pouvoir de faire leurs propres calculs à portée de leurs claviers. Ils pouvaient enfin explorer les modèles, puisque les modèles

1. L'article de Johnsson et Lenhard fait partie d'un livre sur le thème d'un changement d'époque (*epochal break*) de l'activité scientifique.

2. Ann JOHNSON et Johannes LENHARD, « Toward a new culture of prediction : Computational modeling in the era of desktop computing », dans : *Science Transformed ? Debating Claims of an Epochal Break*, sous la dir. d'Alfred NORDMANN, Hans RADDER et Gregor SCHIEMANN, Pittsburgh : University of Pittsburgh Press, 2011, p. 189–200.

leur étaient enfin accessibles sans intermédiaire, dans leur propre ordinateur personnel.

La nature exploratoire de la modélisation est donc tributaire de l'accès à bas prix, pratique et local à la puissance informatique. Cette accessibilité n'est venue qu'avec la maturité de l'ordinateur de bureau (*desktop computer*), en particulier au moment où ces ordinateurs ont été mis en réseau, avec l'avènement d'Internet. Historiquement, cela signifie que le changement dans la pratique de la modélisation s'est produit plutôt dans les années quatre-vingt-dix (l'époque de la popularisation de l'ordinateur personnel) que dans les années cinquante (l'époque des super-ordinateurs, de l'introduction de l'informatique dans la pratique scientifique), ou que dans les années soixante-dix (l'époque de l'utilisation généralisée des ordinateurs centraux (*mainframe*) en recherche). Jusqu'aux années quatre-vingt, la modélisation était un type de calcul qui était commun à toutes sortes de domaines, et constituait une branche indépendante de la recherche (*computing*). À partir des années quatre-vingt, la modélisation dans les sciences a été intégrée entre théorie et expérience et s'est imposée comme une dimension indispensable de l'entreprise scientifique, spécifique à chaque domaine (*computation*). Avec l'omniprésence de l'ordinateur de bureau, l'exploration des modèles est devenue une pratique essentielle des ingénieurs ou des chercheurs utilisant l'ordinateur (*computational scientists*), et pas seulement un domaine réservé aux professionnels de l'informatique (*computing scientists*). Le *desktop modeling* est leur façon de modéliser. Ce mode d'exploration dépend aussi de la capacité des chercheurs à évaluer rapidement les résultats de différentes versions de modèles. L'évaluation des résultats du modèle est particulièrement possible depuis l'apparition du terminal graphique, comme dit plus haut, grâce à la visibilité des résultats de structures de molécules, par opposition aux résultats sur imprimante. La facilité d'accès pour l'utilisateur (*user-friendliness*) ainsi créée favorise le bricolage du modèle par l'utilisateur.

Il y a deux conséquences à cela du point de vue de la relation entre le monde du libre et les chimistes computationnels. La première est qu'on voit apparaître la frontière entre la « république des informaticiens » décrite plus haut, qui correspond aux *computing scientists* et les chimistes computationnels qui font partie des *computational scientists*. Cette fron-

tière correspond donc aussi à un rapport différent à l'ordinateur, d'abord dans la relation avec le matériel, ensuite dans la relation avec le logiciel.

Dans le second cas, Lenhard et Johnson parlent de « marchandisation » (*commodification*) de logiciels scientifiques. Ils se réfèrent à une étude antérieure de l'historien des sciences Terry Shinn qui introduit le concept de « technologie-recherche » (*research-technology*) pour les objets scientifiques à la frontière du monde de la recherche et du monde industriel. Shinn conceptualise la « transversalité », pour définir des instruments d'un genre spécifique, comme par exemple l'ultra-centrifugeuse [1], et Lenhard et Johnsson font valoir que le logiciel de modélisation de l'époque du *desktop modeling* correspond tout à fait aux caractéristiques définies par Shinn et il est tentant d'appliquer l'idée aux logiciels de modélisation de la chimie computationnelle :

- Il est conçu dans des communautés interstitielles : à l'interstice des informaticiens et des chimistes, à l'interstice entre le monde académique et le monde industriel, comme l'ultra-centrifugeuse a été conçue au cours d'une thèse académique sur la physique quantique puis a servi dans le domaine de la biologie, académique et industriel.
- c'est un dispositif générique : il peut servir à plusieurs usages. Le logiciel de modélisation sert dans la recherche fondamentale pour développer les modèles eux mêmes ou pour étudier les propriétés de molécules ou de matériaux, organiques, inorganiques, biomoléculaires ou macromoléculaires. Il sert aussi dans l'industrie, pour la rationalisation du design de médicaments, comme l'ultracentrifugeuse sert à la biologie, la séparation isotopique...
- il est utilisé de manière isolée de son contexte d'invention : dans le cas du logiciel, c'est exactement la tension provoquée par la différence entre le développeur et l'utilisateur.
- il est porteur de nouvelles métrologies. Dans le cas de l'ultracentrifugeuse, l'appareil de « recherche-technologie », a imposé son vocabulaire scientifique aux domaines très différents les uns des autres dans lesquels il a essaimé. Le *desktop modeling* comme le définissent Lenhard et Johnsson n'est pas seulement la vulgarisation et la démocratisation de l'outil informatique. Cela correspond aussi à un changement

1. Bernward JOERGES et Terry SHINN, *Instrumentation Between Science, State and Industry*, Boston : Kluwer Academic Publishers, 2001.

> radical dans la manière de mener des études de modélisation, et le logiciel de modélisation incarne l'objet de « recherche-technologie » qui porte les métrologies nouvelles : une nouvelle façon de modéliser.

Cette nouvelle façon de modéliser admet que la structure et la théorie du modèle ne déterminent pas complètement le calcul de modélisation. De nombreux paramètres ou procédures de calcul ne sont pas figés, ni déterminés par les seules théories mathématiques qui sous-tendent le modèle. On a vu plus haut qu'en mécanique moléculaire, la quantité de paramètres à définir puis évaluer puis calibrer est énorme. Même en chimie quantique, les procédures numériques destinées à résoudre les équations sont difficilement mathématisables. Selon les attentes et les besoins des utilisateurs, selon les contingences de la programmation ou du matériel, selon la disponibilité ou la publication de paramètres ou de procédures essentiels, les caractéristiques du modèle dans le logiciel peuvent être opaques et avoir besoin de tests, de comparaisons, d'adaptations, en un mot de manipulations avant de pouvoir être opérationnellement utilisées. L'appellation de « boîte noire » pour désigner cette opacité n'est pas juste le regret de ne pouvoir en maîtriser tous les paramètres pour l'utilisateur. C'est aussi l'essence même de la manipulation des modèles.

L'idée de logiciel « boîte noire » est souvent indiquée comme à déplorer du point de vue de la reproductibilité scientifique. Ne pas connaître les procédures exactes à l'intérieur de la « boîte noire » empêche le scientifique de contrôler ses calculs donc sa propre production scientifique. Les notions d'*Open Science* ou le *Science Code Manifesto* sont des initiatives récentes pour adapter les idées du libre à ce que devrait être l'activité scientifique, bien au-delà de l'utilisation de logiciels libres ou propriétaires. Il s'agit ici de la deuxième pratique conceptualisée par Kelty : conceptualiser des systèmes ouverts. Les initiatives comme l'*Open Science* sont des réflexions sur les pratiques de l'activité scientifique définies par la communauté elle-même, la communauté scientifique en tant qu'espace public.

Ce « peaufinage des modèles » n'est donc possible que si l'ordinateur est disponible (car il est, à la différence du supercalculateur, à la portée de n'importe quel scientifique) et standardisé. Il est en particulier primordial que les capacités graphiques de l'ordinateur jouent un rôle important dans

la pratique de la modélisation, parce que la lecture facilitée des résultats améliore la capacité à peaufiner le modèle instantanément. Un statut qui embrasse totalement la chimie computationnelle. Mais il est également favorisé par la diffusion des logiciels. Le logiciel de modélisation est l'artefact qui permet la diffusion de la nouvelle forme de modélisation.

Ainsi, ce logiciel est une marchandise à diffuser, ce qui renforce encore l'affirmation selon laquelle il agit comme une « technologie-recherche ». La marchandisation implique plusieurs caractéristiques énoncées par Shinn : les logiciels fonctionnent dans plusieurs disciplines, à travers le clivage public-privé, ils sont isolés des contextes de leur invention afin d'être largement distribués. Les logiciels, en tant que marchandises, sont des boîtes noires qui dépendent, en partie, de détails (paramètres ou routines) dont les mises en œuvre à l'intérieur du programme ne sont pas connues de l'utilisateur. La modélisation informatique est devenue elle-même une marchandise.

« Le mouvement des logiciels ainsi que l'adaptation locale [...] facilite l'échange d'informations scientifiques », affirment Lenhard et Johnsson. Leur description est généraliste, ils ne se concentrent pas sur un domaine scientifique précis. Pour d'autres domaines scientifiques, l'ère du *desktop modeling* s'est imposée plus tard, ou dans un autre contexte économique, académique et politique. Dans le cadre de la modélisation du climat, elle apparaît à une époque de plus grande ouverture aux standards ouverts. Dans le cadre de la bio-informatique, le *desktop modeling* a été la norme depuis le début de la discipline, plus récente. Pour ces deux domaines, les communautés scientifiques académiques ont épousé les pratiques caractéristiques de Kelty dans la création de logiciels, souvent en harmonie avec des pratiques scientifiques inspirées du libre pour « fomenter des mouvements » comme l'*Open Science* ou l'*Open Data*. Le logiciel libre a représenté ainsi une norme suffisamment forte (au sens de Gambardella et Hall) pour que les logiciels de modélisation soient créés puis diffusés dans ce cadre. La chimie computationnelle est unique dans le sens où il s'agit d'un domaine scientifique apparu à l'époque des supercalculateurs, et d'une science financée par les investissements fédéraux (ou nationaux). Pour la chimie computationnelle, contrairement à la modélisation du climat, le *desktop modeling* est apparu à l'ère de la science vécue comme une entreprise, et a impliqué que les gens de la communauté ont dû s'adapter

dans un contexte violent et incertain, un contexte qui a généré de nombreuses tensions et de nombreuses stratégies différentes pour la distribution de logiciels. En ce sens, le passage au *desktop modeling* a été une « épreuve » pour les chimistes computationnels [1].

7. La « générification » des logiciels

Cette vision de l'activité de modélisation a une conséquence : les logiques de diffusion impliquent que le logiciel de modélisation moléculaire soit conçu pour un marché, plutôt que pour un utilisateur. Cette idée est aussi une bonne définition de ce à quoi se réfère Shinn quand il décrit l'une des caractéristiques d'une « technologie-recherche » : elle est générique.

L'éternelle « crise du logiciel » décrite par Ensmenger, implique que les coûts de développement de logiciels sont de plus en plus élevés. Les entreprises qui commercialisent des logiciels scientifiques font facilement faillite (particulièrement quand l'industrie pharmaceutique est en crise, dans le cas de la chimie computationnelle) et celles qui survivent le font en élargissant leur champ d'application et leur base de clients, entre autres par fusion ou acquisition. Avec encore plus de clients à traiter, le coût de l'entretien et de la réactivité aux problèmes (comme la réalisation et l'actualisation d'un *fucking manual* par exemple) est encore plus exigeant. Neil Pollock et ses collaborateurs, en explorant des « biographies » de logiciels industriels, décrivent un processus qui convient parfaitement à cette situation : ils étudient la façon dont le logiciel évolue dans son processus d'acquisition de clients. Ils ne décrivent pas l'« appropriation » du logiciel par les différents utilisateurs locaux, mais plutôt la façon dont le logiciel est façonné de sorte qu'il puisse être utilisé aussi largement que possible [2]. Ils appellent ce processus une « générification » du logiciel, un terme qui renvoie à une des caractéristiques de la « technologie-recherche » employée par Terry Shinn mais aussi à la situation du logiciel

1. En référence à la « sociologie des épreuves ».

2. Neil POLLOCK, Robin WILLIAMS et Luciana D'ADDERIO, « Global Software and Its Provenance : Generification Work in the Production of Organizational Software Packages », dans : *Social Studies of Science* 37.2 (2007), p. 254–280.

de modélisation moléculaire dans un contexte de temps économiques difficiles.

La conception des logiciels « génériques » diffère radicalement des anciennes pratiques de développement de logiciels, particulièrement dans le domaine de la recherche scientifique. Dans ces dernières, les fournisseurs entretenaient traditionnellement des liens étroits avec les utilisateurs, selon l'idée qu'une meilleure connaissance des utilisateurs conduirait à une meilleure conception. À l'opposé, les fournisseurs de solutions « génériﬁées » gardent activement les utilisateurs à distance. Ils craignent que leur logiciel soit identifié et lié à des utilisations spécifiques, le rendant ainsi non commercialisable à grande échelle. Si l'idée est que le logiciel est conçu pour avoir l'étendue d'utilisation la plus large possible, le plus grand nombre d'utilisateurs possible, alors les fournisseurs évitent de traiter avec les utilisateurs individuels, pour minimiser leurs coûts de développements.

C'est exactement ce que les vendeurs de logiciels de modélisation moléculaire ont fait, en concevant des logiciels incluant le plus grand nombre de modèles possible, brouillant les frontières entre des modèles aussi différents que chimie quantique et mécanique moléculaire, dans une course à embrasser autant de clients que possible.

Le processus de « générification », selon Pollock et al., n'est pas simplement une tendance à devenir un logiciel polyvalent. Il consiste en une mécanique impliquant la communauté d'utilisateurs. Pollock et al. l'appellent « gestion de communautés » (*community management*) pour décrire le fait que l'évolution du logiciel a besoin de l'adhésion (*enrollment*) d'utilisateurs initiaux qui constitueront la base sur laquelle les développeurs de logiciels comptera, avec pour effet d'exclure le reste des utilisateurs (c'est-à-dire la majeure partie d'entre eux, plus tardifs, ou moins impliqués) du processus d'élaboration du logiciel.

L'idée est de constituer une communauté suffisante d'utilisateurs, puis de la façonner de sorte que ladite collectivité corresponde le plus possible aux capacités du logiciel. Dans un processus de façonnage mutuel (*mutual shaping)*, cette « première couche » d'utilisateurs va essayer de façonner le logiciel à ses besoins particuliers. La gestion des communautés consiste donc à s'appuyer sur une petite fraction des utilisateurs pour écarter la majeure partie d'entre eux. Abandonnés par les développeurs,

le reste des utilisateurs posent de nombreuses questions sur les forums et les listes de diffusion (*mailing lists*) et les injonctions à lire le *fucking manual* qu'on y trouve traduisent les tensions générées dans la communauté par le sentiment d'abandon. Les utilisateurs de logiciels, mécontents de la façon dont ils sont traités par la maintenance officielle des vendeurs ou des développeurs se tournent massivement vers d'autres sources d'aide et de réconfort... avec plus ou moins de succès. Cette gestion des communautés est à mettre en regard de la quatrième pratique décrite par Kelty qui est la coordination de collaborations. Pour qu'un projet libre soit efficace, mettre en place des collaborations dans la communauté d'une façon ou d'une autre est important. On voit bien, par opposition, qu'au logiciel « générifié » correspond une stratégie sélective dans laquelle la gestion de la communauté consiste à écarter la majorité de celle ci, et que cette stratégie est source de tensions, particulièrement si la communauté en question est une communauté scientifique, censée, selon ses normes mertoniennes communément admises, reposer sur l'universalité, voire le partage. Même s'il serait naïf d'invoquer l'éthique de Merton pour décrire la communauté scientifique (de même que Kelty préfère décrire des pratiques du libre plutôt que des normes), on ne peut que constater que la gestion de communauté du logiciel « générifié » implique des tensions dans la communauté scientifique, du à un choc avec les idées d'universalisme et de communalisme, a priori plus compatibles avec la quatrième pratique de coordination de collaborations.

Pollock décrit le logiciel générique comme un blob noir (*black blob*) au lieu d'une boîte noire (*black box*) : une entité amorphe, accessible seulement aux utilisateurs les plus proches et difficile à appréhender pour les utilisateurs les plus éloignés. L'ouverture (*openness*) des logiciels est ambiguë dans le cadre de cette « générification » : dans un premier temps, ouvrir le code source est une stratégie utile pour enrôler des utilisateurs dans une communauté. Puis, quand le logiciel se développe, l'ouverture devient un inconvénient pour le processus, précisément parce qu'il est considéré comme contre-productif que les utilisateurs soient trop au fait de la machinerie du logiciel. Cette idée est encore à mettre en regard de la première pratique du libre de Kelty : partager le code source. Il est bien sûr évident que le partage du code source est l'essence même du libre, mais il est encore intéressant de noter que dans notre cas, la question est

un problème et encore une fois une source de tension. Pour un logiciel scientifique commercial, la tension existe aussi entre la tendance à protéger son code de la concurrence (pour éviter le *reverse engineering*) et le souhait de la communauté d'accéder à l'intérieur de la boîte noire au nom de la reproductibilité scientifique.

À cet égard, les solutions imaginées par les chimistes computationnels sont multiples, mais l'un d'entre eux, le best-seller en chimie quantique, a développé une idée unique : rendre le code source disponible, mais pas ouvert : une initiative rare d'un logiciel sous licence propriétaire avec code source partagé. Cette approche surprenante a été la source des plus hautes tensions dans la communauté d'autant plus que la licence du logiciel interdit expressément aux utilisateurs de comparer (*benchmark*) et de rendre publiques les performances du logiciel.

La saga du logiciel en question, de ses licences, de ses procès, depuis les années quatre-vingt jusqu'à aujourd'hui est édifiante. Elle résume les tensions et les controverses à l'intérieur de la communauté entre développeurs et utilisateurs. D'autres logiciels ont imaginé de nombreuses autres stratégies, de nombreux autres *business models*. De licences annuelles renouvelables en produits d'appel gratuits mais propriétaires, code source ouvert ou code source fermé, publié ou non, aucune solution ne s'est vraiment imposée comme une norme au sein de la communauté scientifique, révélant la diversité des pratiques et la difficulté à les rendre compatibles entre elles. La troisième pratique décrite par Kelty, l'écriture de licences, prend ici tout son sens. Qu'on soit dans le cadre du libre, ou en train de développer un logiciel « générifié », c'est par l'écriture de la licence que les stratégies se développent (et s'analysent), et la multitude de choix différents dans le domaine de la chimie computationnelle illustre bien les incertitudes stratégiques dans le domaine de création de logiciels en chimie computationnelle, à l'ère du *desktop modeling*.

8. What FM ? Tensions dans la communauté

Ce qui s'est passé avec le logiciel de modélisation moléculaire au cours des années quatre-vingt et quatre-vingt-dix est unique. Le virage du *desktop modelling* en tant que rupture dans l'activité scientifique a été une

source de tensions chez les chimistes computationnels qui ont été les premiers à recevoir de plein fouet ce tournant. Ce qui est arrivé est caractéristique d'un contexte politique, économique, technologique et scientifique mais aussi une aventure de création de logiciels unique. La communauté (dans sa diversité et ses tensions) a façonné le logiciel, et en retour celui ci a transformé l'activité scientifique, à l'image de la communauté, dans un contexte de promesse économique puis de crise et en l'absence de normes : les voies explorées par les développeurs reflètent ces tensions.

Les nombreuses solutions (parfois uniques) trouvées par les développeurs de logiciels de calcul de chimie pour survivre dans le monde du *desktop modelling* ont été conceptualisées grâce à la notion de « générification », amenés d'une part par Shinn (et Lenhard) et de l'autre par Pollock.

Les pratiques caractéristiques des communautés du libre de Kelty ont été comparées aux choix effectués dans la chimie computationnelle. Il est frappant de constater comme les cinq pratiques nécessaires à forger une communauté sont autant de tensions dans le domaine de la chimie computationnelle… une communauté loin du libre.

L'ouverture du code source, tout d'abord, est loin d'être naturelle. Elle est loin de faire l'unanimité dans le monde académique des années quatre-vingt, d'une part aux yeux des administrations universitaires pour lesquelles le « transfert de technologie » est plus important que des valeurs abstraites comme l'universalité de la science, d'autre part aux yeux des développeurs de logiciels commerciaux qui ressentent une faible reconnaissance de la part de la communauté scientifique.

L'idée d'un système ouvert, à cet égard, pour les développeurs de logiciel, est donc vue comme un avantage éventuel au début du développement d'un projet (pour enrôler une communauté d'utilisateurs rapprochés) mais devient vite un défaut quand, au contraire, il faut gérer une communauté d'utilisateurs grandissante, multiple et diversifiée, dans le cadre de la « générification » du logiciel, logiciel vu comme une marchandise.

La communauté (d'utilisateurs, de développeurs) n'est donc pas forcément un endroit où l'on collabore. Le processus de *community management* des logiciels en voie de « générification » consiste plutôt à diviser les utilisateurs, afin que le logiciel de plus en plus polyvalent soit utilisé par le

plus grand nombre plutôt qu'approprié de manières diverses par diverses communautés d'utilisateurs.

Il n'y a donc pas non plus de mouvement, ni d'idée globale et partagée de ce que doit ou devrait être la modélisation moléculaire ou la chimie computationnelle. Une absence de normes, ou d'imaginaires communs, couplée à la demande du marché au moment où cette science a pris son essor ainsi qu'aux perspectives de profits à cette époque, représente les conditions typiques pour empêcher le succès du logiciel libre dans un milieu académique.

Références

ENSMENGER, Nathan L., *The Computer Boys Take Over : Computers, Programmers, and the Politics of Technical Expertise*, Cambridge : MIT Press, 2010.

FLICHY, Patrice, « Internet ou la communauté scientifique idéale », dans : *Réseaux* 17.*97* (1999), p. 77–120.

FRANCOEUR, Eric et Jérôme SEGAL, « From model kits to interactive computer graphics », dans : *Models : The Third Dimension of Science*, sous la dir. de Soraya de CHADAREVIAN et Nick HOPWOOD, Palo Alto, 2004.

GAMBARDELLA, Alfonso et Bronwyn H. HALL, « Proprietary versus public domain licensing of software and research products », dans : *Research Policy* 35.*6* (2006), p. 875–892.

JOERGES, Bernward et Terry SHINN, *Instrumentation Between Science, State and Industry*, Boston : Kluwer Academic Publishers, 2001.

JOHNSON, Ann et Johannes LENHARD, « Toward a new culture of prediction : Computational modeling in the era of desktop computing », dans : *Science Transformed ? Debating Claims of an Epochal Break*, sous la dir. d'Alfred NORDMANN, Hans RADDER et Gregor SCHIEMANN, Pittsburgh : University of Pittsburgh Press, 2011, p. 189–200.

KELTY, Christopher, *Two bits : the cultural significance of free software*, Durham : Duke University Press, 2008.

MOWERY, D. C. et al., « The growth of patenting and licensing by US universities : an assessment of the effects of the Bayh-Dole act of 1980 », dans : *Research policy* 30.*1* (2001), p. 99–119.

POLLOCK, Neil, Robin WILLIAMS et Luciana D'ADDERIO, « Global Software and Its Provenance : Generification Work in the Production of Organizational Software Packages », dans : *Social Studies of Science* 37.*2* (2007), p. 254–280.

RAYMOND, Eric S., *The Cathedral and the Bazaar*, Sebastopol, CA : O'Reilly, 2001, URL : http://www.catb.org/esr/writings/homesteading/.