
Regard de l'informaticien sur ses œuvres et leur originalité

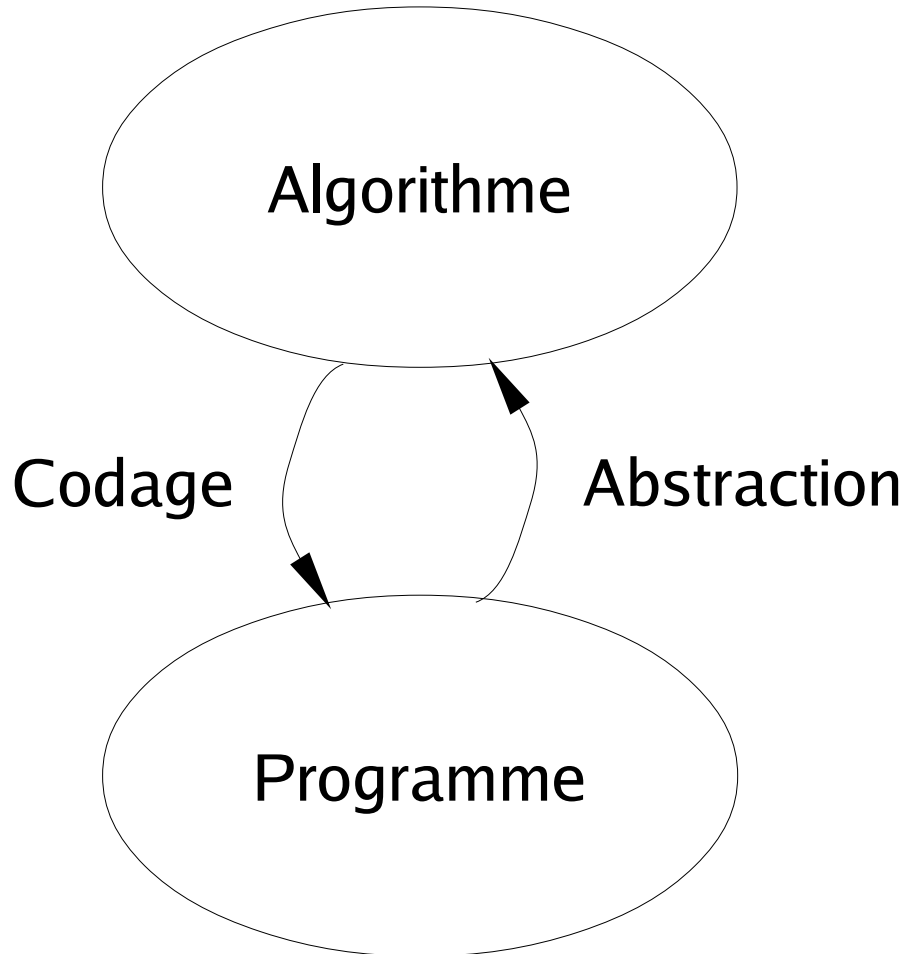
François Pellegrini
Professeur, Université de Bordeaux
francois.pellegrini@u-bordeaux.fr

Ce document est copiable et distribuable librement et gratuitement à la condition expresse que son contenu ne soit modifié en aucune façon, et en particulier que le nom de son auteur et de son institution d'origine continuent à y figurer, de même que le présent texte.

Les logiciels, des œuvres (presque) comme les autres

La création logicielle, une activité
intrinsèquement artisanale... voire un art !

Nature du logiciel



- Les algorithmes sont :
 - Des idées
 - Des mathématiques
- Les programmes sont :
 - Des œuvres de l'esprit
 - Du discours
 - Humain ↔ humain
 - Humain → ordinateur
 - Des processus, lorsqu'ils sont exécutés
- **Similaire au processus de création littéraire**

Droit d'auteur adapté

- Par son rattachement au droit d'auteur, le logiciel est assimilé à une œuvre de l'esprit
 - En France, loi du 3 juillet 1985
 - Directive européenne 91/250/CE (1991)
 - Article 10 des accords ADPIC (1994)
 - Article 4 du traité OMPI WCT (1996)
- Le logiciel est cependant aussi un produit substituable voué à rendre un service
 - Adaptation du droit d'auteur
 - Nous parlerons donc de « Droit d'auteur adapté »
 - Question de la garantie

Œuvre

- L'œuvre est une création de forme
 - C'est la forme qui sera protégée, et non les idées et les concepts qu'elle exprime
 - « Les idées sont de libre parcours » !
- Le droit d'auteur ne concerne que l'expression formelle créée par le programmeur
 - Le « code »

Fonds commun

- Le fonds commun contient l'ensemble des connaissances accessibles au public
 - N'est pas relatif aux œuvres, mais aux « archétypes », aux « algorithmes du monde réel »
 - À ne pas confondre avec le domaine public des œuvres de l'esprit
- Les algorithmes mathématiques utilisés en informatique appartiennent au fonds commun

Critère de protection

- Notion d'« originalité », reflétant la « personnalité de l'auteur »
 - Seul critère admis par la directive 91/250/CE
 - Reformulé maladroitement en France sous le terme d'« apport intellectuel »
- Les créations de forme non originales ne sont pas protégées par le droit d'auteur :
 - Tables mathématiques, etc.
 - Cependant, leur présentation graphique pourra être éligible si elle reflète la personnalité de son auteur (logos, décorations, etc.)

Critères non pertinents

- La « nouveauté »
 - « Encore une histoire de mousquetaires ?! »
 - Des navigateurs web différents sont bien chacun des œuvres originales
 - Leur comportement est régulé par des normes extrêmement précises
 - « Nouveauté » et « activité inventive » sont des concepts relatifs au droit des brevets
- Le « mérite »
 - Qui se souvient encore des peintres « officiels » de la fin du XIXe siècle, par rapport aux « refusés » ?

Quel statut pour les langages ?

- Un langage n'est pas une œuvre
 - Il permet d'écrire des œuvres
 - Il est d'un niveau d'abstraction supérieur
 - Pas de revendication possible sous le régime du droit d'auteur
- Un langage informatique est aussi une langue de communication humaine
 - Permet l'échange d'informations entre humains
- Les langages appartiennent au fonds commun

Organisation juridique et économique de la création logicielle

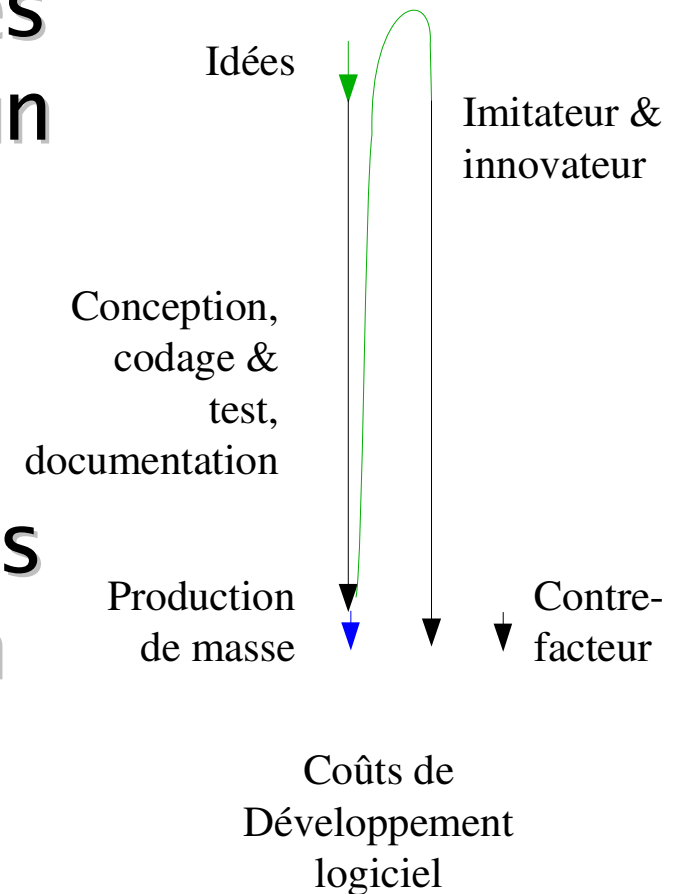
Car on peut (bien) vivre de son art...

Modèle économique de la création logicielle (1)

- La vision promue par la Commission européenne est parfaitement décrite dans les considérants de la directive 91/250/CE :
 - Secteur industriel de biens substituables
 - Nécessité d'établir les règles d'une concurrence libre et non faussée
 - Respect des droits des auteurs

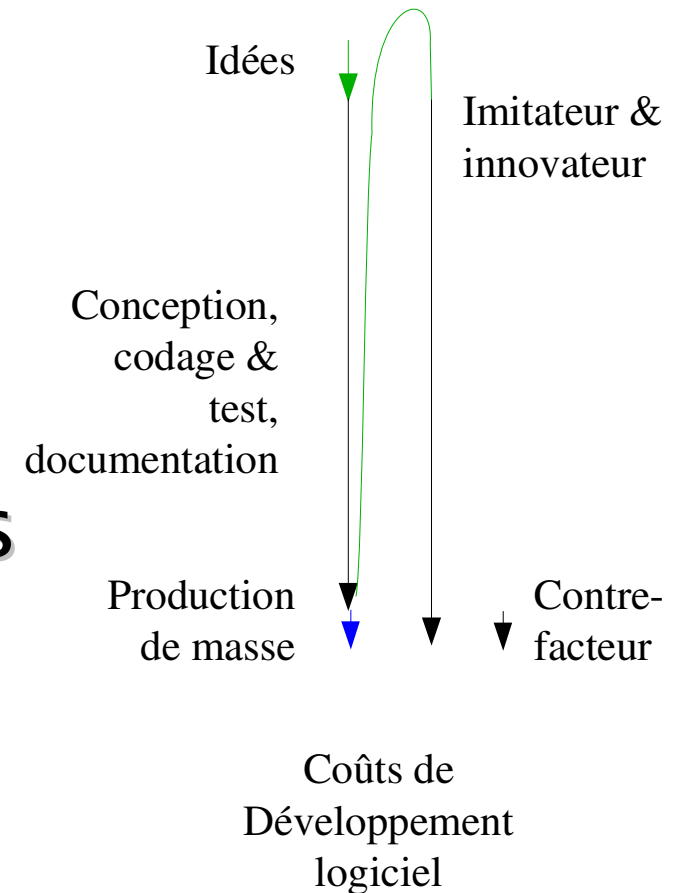
Modèle économique de la création logicielle (2)

- L'observation des fonctionnalités d'un logiciel existant permet à un nouvel arrivant de réaliser un logiciel concurrent
- Il n'y a concurrence libre et non faussée que s'il paye les mêmes coûts d'entrée sur le marché en re-codant son logiciel
 - Il ne peut donc réutiliser le code existant, par décompilation et/ou copie servile



Modèle économique de la création logicielle (3)

- Pendant ce temps, l'innovateur original peut progresser
 - S'il ne le fait pas, il sera dépassé par les innovations des autres
- Nécessité de traiter spécifiquement le problème des marchés captifs créés par les formats de données
 - Dispositions relatives à la décompilation « à fin d'interopérabilité »



Adaptation des droits patrimoniaux (1)

- L'adaptation des droits patrimoniaux vise à adapter les modalités d'exploitation économique aux spécificités de ce type d'œuvres
 - Droits individuels accordés au service d'une vision macro-économique
 - Tout comme le droit de suite est spécifique aux œuvres plastiques

Adaptation des droits patrimoniaux (2)

- Ajout de nouvelles exceptions
 - « Actes nécessaires pour permettre l'utilisation du logiciel »
 - « Observation du fonctionnement »
 - En fait, simple rappel de la loi
 - Copie de sauvegarde
 - Si aucun autre moyen fourni par l'éditeur
 - La décompilation
 - Interdite sauf « à fin d'interopérabilité »
- Suppression de l'exception de copie privée
 - Nécessité d'obtenir une licence pour chaque interaction avec le logiciel

Compilation, décompilation et interopérabilité

Faire et défaire,
c'est toujours travailler !

Langages de bas et haut niveau

- Les ordinateurs ne comprennent que le « langage machine », très rudimentaire et spécifique à chaque matériel
- Besoin d'écrire des programmes dans des langages plus expressifs, dits « de plus haut niveau »
- Des outils automatiques permettent la traduction de textes d'un langage vers un autre

Compilation et décompilation (1)

- La compilation consiste à traduire un programme écrit dans un langage de haut niveau en un programme fonctionnellement équivalent écrit dans un langage de bas niveau, susceptible d'être exécuté par un ordinateur
 - Le « code source » est le programme écrit en langage de haut niveau que l'on veut traduire
 - La « forme préférée » d'écriture d'un programme
 - Le « code objet » est le programme résultant écrit dans un langage de bas niveau
 - Définit le « programme exécutable »

Compilation et décompilation (2)

- On appelle « décompilation » l'action inverse de la compilation, permettant d'exprimer dans un langage de haut niveau un programme originellement écrit dans un langage de bas niveau
 - Bien plus difficile à mettre en œuvre
 - Informations structurelles de haut niveau « diluées » dans le code objet

Traduction automatique

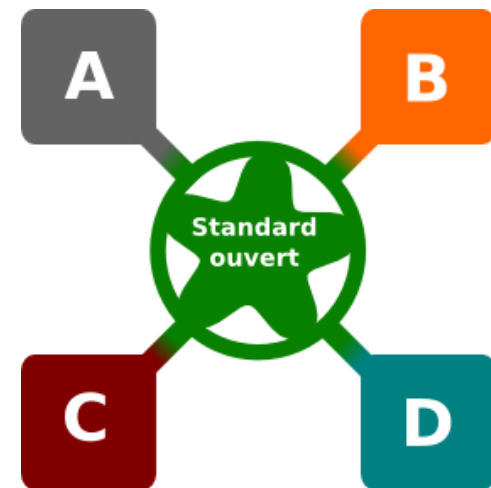
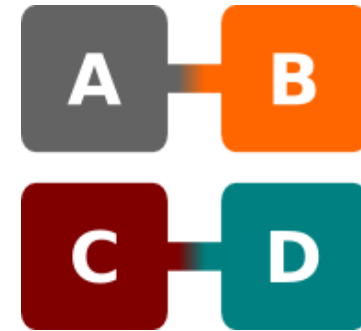
- La traduction du code source en code objet se faisant de façon automatique, le code objet est une œuvre intégralement dérivée du code source, sans aucun apport original d'un autre auteur
- Le créateur d'un outil n'a aucun droit sur les créations réalisées au moyen de cet outil

Interdiction de la décompilation

- Compiler et décompiler créent des œuvres dérivées de l'œuvre originale
- Ces copies ne peuvent donc être exploitées qu'avec l'accord de l'ayant droit des œuvres originales
- La décompilation est donc interdite, sauf permission explicitement donnée par l'ayant droit

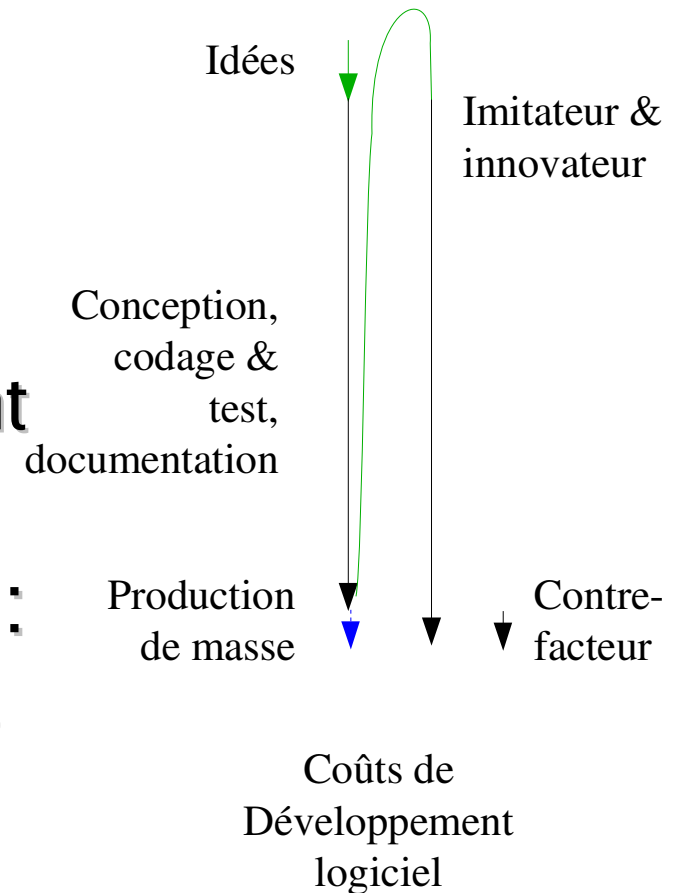
Compatibilité n'est pas interopérabilité !

- Il y a compatibilité quand deux produits peuvent fonctionner ensemble
 - Accord contractuel entre les parties
 - L'entité contrôlant le format « compatible » étend son monopole
- Il y a interopérabilité quand deux produits peuvent fonctionner ensemble et que l'on sait pourquoi



Recherche de l'interopérabilité

- Pour favoriser l'innovation :
 - Les investissements des innovateurs initiaux doivent être récompensés
 - Les nouveaux entrants ne doivent pas être empêchés de concourir
- Pour créer la libre concurrence :
 - La copie servile doit être interdite
 - Le *redesign* doit être légal
- Les marchés captifs doivent être découragés



Statut des interfaces (1)

- La libre concurrence suppose qu'on puisse interfacier un nouveau logiciel avec un logiciel existant
 - Voire le remplacer en s'interfaçant avec les logiciels tiers avec lesquels ce logiciel interagissait
- Or, lorsqu'on veut s'interfacier avec un logiciel existant, il faut respecter ses interfaces
 - Décrites dans des fichiers d'en-tête, des fichiers de classes, etc.
- Pour s'interfacier, il faut donc « recopier » le format des interfaces et leur nommage

Statut des interfaces (2)

- Pour s'interfacer, il faut donc « recopier » le format des interfaces et leur nommage
 - Il n'y a pas d'originalité dans le code d'interfaçage
- Une interface ne peut être protégée
 - Une interface donne l'accès à une extension d'un langage

Et l'originalité dans tout ça ?

Au travers de quelques études de cas...

Devant le tribunal (1)

- L'existence des droits ne transparaît qu'à travers les preuves que l'on peut exhiber aux yeux du juge
- Il y a litige s'il existe deux parties se disputant au sujet d'un ou plusieurs logiciels
- En termes de droit d'auteur, à partir de quel moment peut-on caractériser la contrefaçon ?
- Ne pas confondre les questions de l'originalité et de la titularité des droits !
 - Un plagiat est bien issu d'une œuvre « originale » !

Similarité des fonctionnalités ?

- A accuse B d'avoir contrefait son logiciel parce que L_B fait la même chose que L_A
 - Liberté d'observation des fonctionnalités
 - Liberté de refaire un logiciel réalisant les mêmes actions
 - Droit à la décompilation pour briser les marchés captifs autour des formats de données et/ou d'échange fermés
 - Y compris utilisation du code !
- Moyen invalide
 - Usage dévoyé des brevets algorithmiques (improprement appelés « brevets « logiciels »)

Similarité du code ? (1)

- A accuse B d'avoir contrefait son logiciel parce que le code de L_B ressemble « beaucoup » à celui de L_A
 - Existence d'une source commune ?
 - Existence d'une fuite de données ?
 - Espionnage industriel ? Salarié indélicat ?
 - Respect d'interfaces ?
 - Existence d'un espace de liberté formelle ?

Similarité du code ? (2)

- Ce n'est pas parce que le code de L_B diffère de celui de L_a qu'il n'y a pas contrefaçon !
 - Le code de L_B a pu être obtenu par traduction (semi-)automatique à partir de celui de L_a
- Rôle de l'expert dans la détermination du degré de fortuité des similarités d'expression du code

Conclusion

Conclusion (1)

- Synthèse des espaces couverts

Fonds commun : entités abstraites
- algorithmes, spécifications fonctionnelles et d'interfaces, etc.

Créations de forme

Créations non originales :

- tables de marées,
mathématiques,
mise en œuvre
d'interfaces, etc.

Œuvres de l'esprit

Entrées/placées
dans le domaine
public

Couvertes par
des droits
patrimoniaux
(licences libres et
privatives)

Conclusion (2)

- En tout état de cause, l'originalité d'une œuvre logicielle peut/doit toujours être présumée
 - L'auteur a juste « fait son travail »
 - L'équivalence des comportements observés n'est pas significative
- Aucun intérêt économique à dénier l'originalité à certains logiciels
 - Base juridique incertaine
 - Laisse les auteurs sans protection
 - Jugement au « mérite »
 - Non pertinent vis-à-vis des actions à réprimer

Conclusion (3)

- Les litiges en matière de droits d'auteur doivent se résoudre à travers l'étude du code source et de la façon dont il a été produit
 - Analyse qualitative et non pas quantitative
 - Pas de « *sweat of the brow* »
- Pas de renversement de la charge de la preuve
 - C'est à l'attaquant de faire la preuve qu'un code a été obtenu de façon déloyale, en violation du modèle économique de la création logicielle : copie servile, traduction automatique, etc.