

# Cours Logiciels Libres

## Économie du logiciel libre

Théo Zimmermann  
(Cours adapté à partir de matériel de Ralf Treinen)

Télécom Paris, Institut Polytechnique de Paris

Vendredi 31 mars 2023



## Quelques fausses idées

## Quelques fausses idées (1)

« *Logiciel propriétaire = logiciel professionnel* »

## Quelques fausses idées (1)

« *Logiciel propriétaire = logiciel professionnel* »

Réponse:

- Un logiciel propriétaire n'est pas forcément professionnel.
- Un logiciel professionnel peut très bien être libre, et il y a des bonnes raisons pourquoi les professionnels peuvent préférer le logiciel libre.

## Quelques fausses idées (2)

*« Le logiciel libre n'est pas utilisable par des professionnels car il n'y a pas de garantie »*

## Quelques fausses idées (2)

*« Le logiciel libre n'est pas utilisable par des professionnels car il n'y a pas de garantie »*

Réponse:

- Dans le monde propriétaire il n'y a pas de garantie non plus, sauf si on paye cher.
- On peut acheter un service de maintenance, et cela beaucoup plus facilement dans le cas du logiciel libre.

## Quelques fausses idées (2)

*« Le logiciel libre n'est pas utilisable par des professionnels car il n'y a pas de garantie »*

Réponse:

- Dans le monde propriétaire il n'y a pas de garantie non plus, sauf si on paye cher.
- On peut acheter un service de maintenance, et cela beaucoup plus facilement dans le cas du logiciel libre.

Pourquoi ?

## Quelques fausses idées (2)

*« Le logiciel libre n'est pas utilisable par des professionnels car il n'y a pas de garantie »*

Réponse:

- Dans le monde propriétaire il n'y a pas de garantie non plus, sauf si on paye cher.
- On peut acheter un service de maintenance, et cela beaucoup plus facilement dans le cas du logiciel libre.

Pourquoi ?

Car il existe souvent plusieurs concurrents qui vendent de la maintenance pour le même logiciel libre.



## Quelques fausses idées (3)

*« Le logiciel libre est dangereux pour la sécurité car tout le monde peut facilement trouver les failles de sécurité. »*

## Quelques fausses idées (3)

*« Le logiciel libre est dangereux pour la sécurité car tout le monde peut facilement trouver les failles de sécurité. »*

Réponse: La sécurité par l'obscurité ne fonctionne pas.

- Rétro-ingénierie.
- Espionnage, vente de secret industriels par des initiés.
- Révélation accidentelle de secrets.

Au contraire, dans le logiciel libre, si les failles sont visibles par tout le monde, cela veut dire qu'il y a plus de chances que **quelqu'un va les rapporter** et donc qu'elles vont être corrigées.

## Quelques fausses idées (4)

*« Si tout le monde ne fait que du logiciel libre, alors tous les informaticiens seront au chômage. »*

## Quelques fausses idées (4)

*« Si tout le monde ne fait que du logiciel libre, alors tous les informaticiens seront au chômage. »*

Réponse:

- De nombreuses entreprises rémunèrent des employés pour contribuer (à temps plein ou à temps partiel) à du logiciel libre.
- La production n'est pas l'activité la plus coûteuse dans le cycle de vie d'un logiciel. C'est plutôt la maintenance, qui, quant à elle, peut être vendue aussi chère dans le cas d'un logiciel libre.

# Modèles économiques

## Le modèle classique

- Dans le monde des logiciels propriétaires:  
Revenus basés sur la **vente de licences** d'utilisation.
- Production de logiciels: coût élevé de développement, coût de la copie très bas (quasiment nul).
- C'est un modèle de *rente* qui peut être extrêmement profitable (dépend davantage du nombre de clients que des efforts fournis par le vendeur).
- Modèle similaire pour la musique, les films, etc.

## L'exclusion d'usage commercial

- Logiciel **gratuit pour les usages non commerciaux** (ou pour les très petites entreprises / indépendants) mais payant pour les usages commerciaux (ou au-delà d'une certaine échelle).
- Cela permet d'avoir une plus large communauté d'utilisateurs (voire de contributeurs) sans renoncer à l'économie de rente.
- Même lorsque le code est public, ce n'est **pas du logiciel libre**, et beaucoup d'utilisateurs ou contributeurs potentiels en auront conscience. Par exemple :
  - **Pas de garantie sur l'avenir du logiciel** en cas de faillite / renoncement / rachat de son producteur.
- Exemples actuels : SublimeText (éditeur), Obsidian (notes).

# Modèles économiques du logiciel libre

- ① Les modèles basés sur le *développement* payant et non la *vente*.
- ② Les modèles basés sur la vente de versions propriétaires étendues par rapport à une version libre et gratuite (*open core*) ou bien de licences spécifiques.
- ③ Les modèles qui ne sont pas basés sur la vente des logiciels mais sur des services annexes : maintenance, hébergement, etc.

Chaque acteur peut combiner plusieurs modèles à sa guise. Dans la suite, on se focalise sur des *éléments* constitutifs de modèles économiques.



## Développement payant

# Explications

- Un client a besoin d'un certain produit logiciel.
- Il peut embaucher un développeur, ou une entreprise (éditeur de logiciel) qui développe le logiciel.
- Le client (qui est *a priori* propriétaire du logiciel qu'il a fait développer à ses frais) peut choisir de publier le produit sous licence libre pour des raisons diverses.
- Modèle fréquent lors de la commande de logiciels par des institutions publiques (notamment aux US).



- Exemple : Développement du compilateur ADA dans les années 80 sous licence libre.

# Risques à la commande de logiciel libre

**Attention** : licence libre  $\neq$  modèle de développement open source

- Pour bénéficier des avantages du modèle open source (exemple : rassembler une communauté de contributeurs), il **ne suffit pas** de produire le logiciel et publier son code sous licence libre.
- Si le client commande la production d'un logiciel libre à une entreprise dont le libre n'est **pas la spécialité**, il faut se méfier :
  - documentation inadéquate,
  - installation difficile,
  - dépendances sur du code propriétaire ou sur des versions ésoériques de logiciel libre.

## Conseils pour la commande de logiciel libre

- Karl Fogel conseille d'avoir recours à une seconde entreprise pour servir d'**assurance qualité open source**, qui va :

## Conseils pour la commande de logiciel libre

- Karl Fogel conseille d'avoir recours à une seconde entreprise pour servir d'**assurance qualité open source**, qui va :
  - Participer à la communauté comme contributeur **externe**.
  - Vérifier la qualité de la documentation, de l'accessibilité de l'installation, des processus de contributions.
  - Rapporter les problèmes **directement au client**.

## Conseils pour la commande de logiciel libre

- Karl Fogel conseille d'avoir recours à une seconde entreprise pour servir d'**assurance qualité open source**, qui va :
  - Participer à la communauté comme contributeur **externe**.
  - Vérifier la qualité de la documentation, de l'accessibilité de l'installation, des processus de contributions.
  - Rapporter les problèmes **directement au client**.

### Avantages :

- Meilleure **qualité** du logiciel produit (documentation, modularité, dette technique, etc.).
- Plus de garanties sur la possibilité d'avoir recours à une **entreprise indépendante** pour :
  - son installation / déploiement,
  - sa maintenance future.

## Les modèles basés sur la vente de logiciels

## En bref

- Vente d'exceptions à une licence copyleft ou de versions améliorées non libres.
- Coexistence de versions libres et de versions non libres du logiciel (modèle hybride).
- Lorsque cette stratégie repose sur une licence copyleft, cela nécessite d'avoir des droits que les concurrents n'auront pas (par l'utilisation d'un CLA ou CTA).



## Le modèle des licences décalées

- Idée : Vente de licences propriétaires pour les versions les plus récentes et les plus à jour.
- Après un **délai**, les versions sont distribuées sous licence libre.



- Exemple : L'ancien modèle de **ghostscript**, un interpréteur du langage Postscript (utilisé par les imprimantes).
- Problèmes :
  - Requier un CLA / CTA pour les contributions externes.
  - Les distributions Linux veulent distribuer du logiciel libre, donc **risque de création d'un fork** avec des correctifs à jour.


# Le modèle des extensions propriétaires (*open core*)

- Deux versions :
  - Une version libre.
  - Une version propriétaire **étendue avec plus de fonctionnalités** (qui peuvent finir par être ajoutées à la version libre).
- Les entreprises sont encouragées à acheter la version étendue :
  - Choix d'un vocabulaire (trompeur) :  
"Community Edition" / "Enterprise Edition"
  - **Vente combinée** de services (déploiement/maintenance) avec les extensions.
- Avantages :
  - Si la version libre n'est pas sous licence copyleft, alors le recours à un CLA / CTA n'est pas nécessaire.
  - Les distributions Linux peuvent inclure la version libre, qui sera néanmoins à jour.
- Exemple : Modèle passé de Netscape (licence MPL), actuel de GitLab (licence MIT).

## Le modèle de la vente d'exceptions

- Publication sous licence libre *copyleft* (par exemple GPL).
- Vente de licences qui permettent l'utilisation du logiciel dans des produits qui ne seront pas publiés sous GPL.
- C'est un modèle que la FSF trouve acceptable, contrairement aux deux précédents (mais qui requiert aussi un CLA ou CTA) :  
<https://www.gnu.org/philosophy/selling-exceptions.html>



- Exemple :  un des systèmes de bases de données les plus importants (entreprise rachetée par Sun pour un milliard de dollars, et ensuite par Oracle).

## Modèles basés sur les services / produits annexes

## En bref

- Beaucoup d'entreprises peuvent contribuer à du logiciel libre car il leur est utile sans être le produit qu'elles vendent.
- Parfois ce qu'elles vendent peut être en lien direct :
  - Un autre produit lié.
  - Un service lié (maintenance, support, hébergement).

## La lutte pour les quasi-standards

- *Quasi-standard* : un standard *de fait*, en opposition à un standard *de jure* (porté par un organisme avec le pouvoir de définir des standards).
- Logiciel libre: la grande disponibilité d'une technique peut mener à la formation d'un quasi-standard, ou empêcher un concurrent d'établir un quasi-standard basé sur une autre technologie (peut-être propriétaire).
- Exemple : La décision de *Netscape* de convertir son navigateur vers un produit libre (*Mozilla*), lutte contre la menace d'un quasi-monopole de Microsoft (enjeu : le marché des serveurs).



## Vente de maintenance / support : exemple de RedHat



- Fondé en 1994.
- Chiffre d'affaires : 1 milliard de \$ en 2012.
- Résultat net : 199 millions de \$ en 2012.
- Nombre d'employés : 6 100 en 2014.
- Vente de support : entre 350\$ et 2 500\$ par an et serveur.
- Basée sur une distribution gratuite et libre : *Fedora*
- 82% des revenus (2012) viennent de la vente de contrat de support, le reste de la formation et du conseil.
- 18% des coûts (2012) sont pour le développement Open Source (les résultats reviennent donc à la communauté).

# Hébergement (modèle SaaS) : exemple de Zulip



- SaaS = Software as a Service
- Zulip est un logiciel libre de chat / alternative à Slack très populaire depuis quelques années.
- L'entreprise Zulip propose d'héberger des serveurs pré-configurés et automatiquement mis à jour :
  - Gratuitement pour les projets open source, les équipes de recherche...
  - De manière payante pour les entreprises.



# Vente de produits annexes : Doom



- Jeu créé en 1992, technologie d'animation très innovante à l'époque.
- Code source publié en 1997, licence GPL en 1999.
- Améliorations du code source et portage vers des nouvelles architectures d'ordinateurs par la communauté.
- Depuis : revenus de l'entreprise *Id Software* basés sur la vente de scénarios.