Cours sur les Logiciels Libres (LL)

Master 2 Professionnel – Mention Informatique

Spécialité : Conception et développement de solutions informatiques intégrées

Teresa Gomez-Diaz

Laboratoire d'informatique Gaspard-Monge – PLUME

Université d'Angers, 21-23 février 2011

Objectifs, motivation du cours (1/2)

Présenter les logiciels libres :

- Définition, exemples
- Origine, histoire, philosophie(s)
- Les projets libres : exemple PLUME
- Aspects juridiques : droit d'auteur (CPI), licences
- Développement : méthodes, XP, outils, ...
- Développeurs : qui ?, comment ?, pourquoi ?, où ?
- Les sociétés et les métiers, la migration
- Les logiciels libres aujourd'hui : où, évènements, ...

Ce cours réutilise le cours de 2010 donné par Jean-Luc Archimbaud.

Objectifs, motivation du cours (2/2)

Objectifs plus généraux :

- Vous êtes confrontés aux logiciels libres :
  utilisateurs et peut-être développeurs
- Comprendre les problèmes associés aux développements :
  techniques, légaux, formation, organisation et management, ...
- Réflexion : loi, politique, modèles économiques,
  gestion de projets, communautés, associations, communication, ...
- Motiver, encourager l'utilisation de PLUME :
  recherche et diffusion d'information

Vidéo : Journée PLUME-Cléo, Pierre Mounier
http ://www.projet-plume.org/ressource/journee-plume-cleo-CMS-wiki-blog

Document à lire : Stage RMLL, http ://www.annabelleoliveira.com/

Principaux documents de référence

- Jean-Luc Archimbaud, Les logiciels libres : caractéristiques, utilisation dans le développement et place dans les Systèmes d'Information, Angers, 2010. http ://www.projet-plume.org/ressource/cours-logiciels-libres-01-2010
- Karl Fogel, Produire du logiciel libre, traduction en français par Framalang, 2011. http ://www.framabook.org/Produire_du_logiciel_libre.html
- Eric S. Raymond, La cathédrale et le bazar (The Cathedral and the Bazaar), traduction en français par Sébastien Blondeel, 1998. http ://www.linux-france.org/article/these/cathedrale-bazar/cathedrale-bazar_monoblock.html
- Carlo Daffara, *Open Source adoption : best practices from European experiences*, OWF 2010. http ://www.openworldforum.org/attend/speakers/carlo_daffara-open_source_adoption.pdf
- PLUME - http ://www.projet-plume.org/
- ENVOL - T. Aimé, A. Laprevote, F. Miller, F. Pellegrini, L. Perochon http ://www.projet-plume.org/ressource/presentations-envol-2008 http ://www.projet-plume.org/ressource/supports-de-cours-et-de-tp-envol-2010

Définition

Selon la Free Software Fondation, 1985, (FSF, http ://www.fsf.org), un logiciel est libre si ces quatre libertés sont garanties :

- liberté d'exécuter le logiciel
- liberté d'étudier le fonctionnement, de l'améliorer
  $\Longrightarrow$ **disponibilité du code**,
- liberté de redistribuer des copies,
- liberté de publier les améliorations.

Un logiciel est libre parce qu'il a une licence (libre) qui garantit ces quatre libertés : *Ce logiciel est libre* ne veut rien dire, il faut une licence.

Tout logiciel qui n'est pas libre est propriétaire mais un logiciel peut être libre **et** propriétaire.

Trois cas de figure : utilisateurs, développeurs, ou les deux à la fois.

Carte conceptuelle du logiciel libre

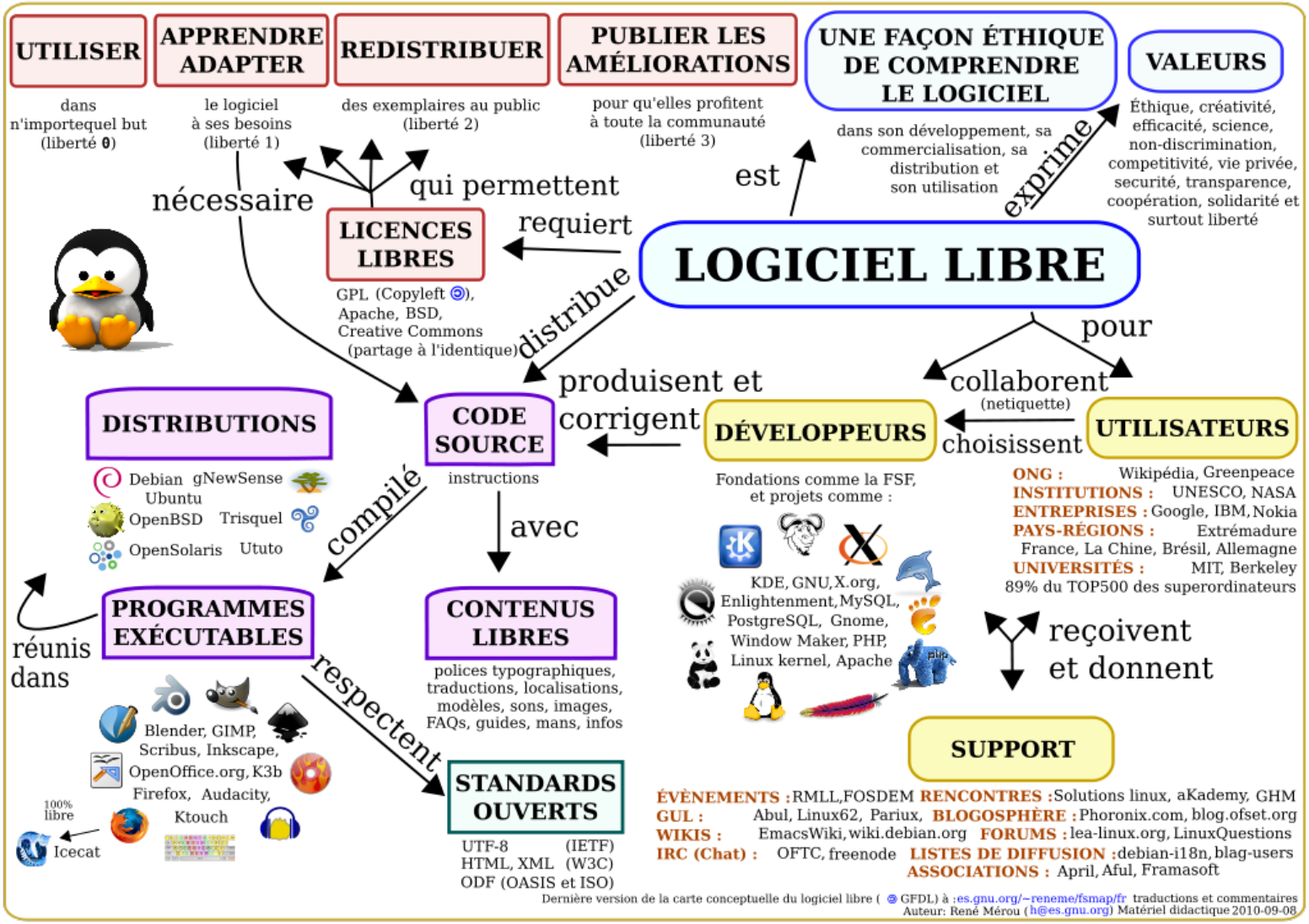


Figure: http ://fr.wikipedia.org/wiki/Fichier :Carte_conceptuelle_du_logiciel_libre.svg

Quelques exemples de logiciels libres

En connaissez vous ? :

- OS : distributions GNU Linux, MAC OS
- Serveur Web : Apache
- Navigateur : Mozilla Firefox
- Bureautique : OpenOffice.org, TeX-Latex
- Images : GIMP
- Base de données : MySQL (admin Web avec phpMyAdmin)
- Multimédia : VideoLAN VLC
- CMS : Joomla !, SPIP, Drupal
- Embarqués : GPS TomTom, FreeBox, iPhone, TV, avions, ...
- des millions, ...

Préhistoire des logiciels libres

Le partage de logiciels est aussi ancien que les logiciels :

- ce qui est payant est le matériel, le temps de calcul,
- les acheteurs (beaucoup en milieu scientifique) sont capables de modifier et d'améliorer les logiciels, et distribuent leurs correctifs,
- les fabricants encouragent cette pratique qui rend les machines plus attrayantes (et plus fiables).

Deux aspects importants :

- matériel non standard, logiciels dépendant de l'architecture (les utilisateurs ne vont pas acheter autre chose),
- internet n'existe pas, échanges compliqués (matériel, temps).

C'est-à-dire pas de concurrence ...,
une fois une architecture matérielle choisie, on y reste.

Naissance des logiciels libres

Trois conditions/évolutions :

- matériel moins cher, plus standard, diminution des écarts de performances,
- naissance des langages de haut niveau, les logiciels ne sont plus dépendants de l'architecture matérielle,
- naissance d'internet.

Les acheteurs sont moins liés à une architecture matérielle, le modèle économique est déplacé vers les logiciels qui deviennent payants, les constructeurs n'encouragent plus les échanges de code (qui peut parvenir aux concurrents) et appliquent le droit d'auteur (*US copyright law*) de façon stricte, avec des licences propriétaires.

Existence des LL, mais pas du concept

- Donald Knuth développe TeX, 1978, suivi de Latex par Leslie Lamport. La licence de TeX permet la libre distribution et modification mais à condition de changer le nom du produit résultant.
- UCB (University of California-Berkeley), 1977, développement de BSD (Berkeley Software Distribution), autour d'un noyau AT&T. Exemple LL non-idéologique, terrain d'entraînement.
- MIT (Massachusetts Institute of Technology), 1985,
  X Window System : la licence MIT donne les libertés d'utilisation, modification, redistribution, publication, ...
  Condition : inclure le Copyright MIT dans toute copie du code (ou des parties importantes du code).

En France, le laboratoire LIP6 avait un *repository ftp anonyme* :
http ://ftp.lip6.fr/ où on pouvait déposer du code pour sa libre redistribution.

Pas de composante idéologique de *hacker* (attention aux significations).

R. Stallman, FSF, GNU, GPL

Richard Stallman, chercheur au MIT (décennie 70, début 80), ambiance de développement et de collaboration (*hacker*) au laboratoire.

Le laboratoire est rattrapé par les changements de l'industrie informatique :

- start-up avec des membres du labo pour faire un nouvel OS concurrent de celui du laboratoire, mais avec licence exclusive,
- acquisition de nouveaux matériels livrés avec des systèmes d'exploitation propriétaires.

En 1984 R. Stallman lance la Free Software Fondation et le projet GNU pour créer un système d'exploitation pour ordinateur complètement *libre* et ouvert.

En 1989 : première version de la licence GNU GPL avec clause pour éviter qu'un logiciel sous GPL ou ses dérivés deviennent propriétaires.

L. Torwalds, Linux kernel

Le projet GNU a des projets divers pour des composantes d'un système d'exploitation : gcc, emacs, ... Il y avait aussi un projet de noyau (kernel) avec des spécifications très avancées, mais difficiles à mettre en œuvre ...

En 1991, Linus Torvals, étudiant en informatique finlandais, produit une première version d'un noyau pour son ordinateur personnel et il la distribue très rapidement sous licence GPL.

Ce noyau attire l'attention des développeurs, il évolue rapidement avec leur collaboration.

Avec les autres composantes existantes (GNU, MIT, BSD, ...) on obtient l'ensemble GNU/Linux et la base des distributions Linux qu'on connait aujourd'hui.

Document à lire : Eric S. Raymond, La cathédrale et le bazar.

Naissance de l'Open Source

Le mot *free* pose des problèmes en anglais, on dira souvent *free as in free speech, not as in free beer*.

Cela implique des difficultés dans le monde des affaires (plusieurs entreprises sont nées autour de ces OS), et après réflexion le concept de *open source* ou code ouvert apparait.

L'Open source initiative (OSI, http ://www.opensource.org) donne une définition de code open source en 10 points et forme un mouvement dans le monde des logiciels libres qui est plus proche du monde des affaires que du monde idéologique et des libertés du logiciel libre.

Cela n'empêche pas les collaborations étroites et pragmatiques entre ces deux mondes avec des idéologies un peu différentes mais des buts communs.

Ils ont une bonne raison pour rester unis : la qualité du code produit.

Un peu de réflexion

Initialement il s'agit d'une réponse idéologique à un modèle économique qu'on n'accepte pas, avec des conséquences qui dépassent aujourd'hui la frontière de l'informatique et de la technologie : aspects de communauté, de travail en équipe, aspects légaux, nouveaux modèles économiques, ... dont on parle aujourd'hui dans le monde culturel et de production artistique (par exemple), on parle aussi de *free hardware* ...

Le vocabulaire est très important, l'utilisation du mot *free* freine les contacts avec des investisseurs, il a fallu une re-adaptation pour le monde des affaires.

Même s'il s'agit de deux philosophies différentes entre le Free software et l'Open source, du point de vue légal il n'y a pas de différence dans les concepts et droits qui interviennent.

On parle parfois de FLOSS : Free/Libre Open Source Software.
J'utiliserai les deux terminologies, en français ou en anglais.

Définition de Projet Open Source

Le concept de Projet Open Source dépasse les frontières du logiciel. Voici la définition donnée par Venkatesh Hariharan (Red Hat) lors des Journées IRILL, Paris 4-5 octobre 2010.

Projet Open Source

Un Projet Open Source s'appuie sur trois piliers :

- la collaboration (*collaboration*),
- la communauté (*community*),
- le partage de la propriété d'une connaissance (*shared ownership of knowledge*).

La clé du succès : l'architecture de la participation.
The key of succes of open source projects, the most compelling : the architecture of participation.

On verra l'exemple de PLUME comme projet Open Source dans ce sens.

- Le projet PLUME

PLUME

Promouvoir les **L**ogiciels **U**tiles **M**aîtrisés et **E**conomiques
dans l'Enseignement Supérieur et la Recherche

http ://www.projet-plume.org/
plume@services.cnrs.fr

- Initialiement porté par l'UREC/CNRS, né vers la fin 2006.
- Depuis juin 2010 : ARESU/DSI/CNRS.
- Directeur du projet : Jean-Luc Archimbaud
- Partenaires officiels : 45 laboratoires et autres entités,
- dont 27 avec un fort soutien (personnes, financement...)
- Succès reconnu : + de 200 000 l./mois, très bien indexé (Google)

- Le projet PLUME (2)

Le projet a 4 objectifs :

- Mutualiser les compétences sur les logiciels (et les valoriser)
- Promouvoir les développements internes
- Animer une communauté autour du logiciel
- Promouvoir l'usage et la contribution aux logiciels libres

Pour atteindre ces objectifs :

- plate-forme PLUME
- publication de fiches descriptives sur les logiciels
- fils RSS, agenda d'évènements LL, brèves
- écoles thématiques (ENVOL), journées PLUME
- réseau DEVLOG, ...

Visiter : http ://www.projet-plume.org/

- Statistiques

Il y a 6 types de fiches sur PLUME (statistiques, février 2010) :

- fiches orientées vers des utilisateurs potentiels :
  - (322) fiches de logiciel validé : en production, +3 sites
  - (35) fiches de logiciel à valider : en production, 1 ou 2 sites
  - (11) fiches de logiciel en test : compte-rendu, rédaction collaborative
- fiches avec des informations autour des logiciels :
  - (224) fiches ressource (articles, FAQ, évènements, ...)
- fiches orientées recherche, international, laboratoire, tutelles, patrimoine, valorisation, évaluation :
  - (216) fiches dév. ESR (RELIER)
  - (56) fiches dév. ESR en anglais (PLUME-FEATHER)
- 11 archives (garder l'information à jour), +200 fiches en cours

Travail fait par des personnes : 1567 membres, 676 contributeurs et 25 responsables thématiques dont 1 rédacteur en chef.

- Développement (1/4)

La plate-forme PLUME utilise des LL, son développement commence par le choix et configuration de briques libres, il n'y a pas vraiment du développement logiciel.

- Maquette : 6 mois
  - Avec un existant SPIP et des fiches en PDF :
  - Est-ce faisable, ça répond à des besoins ?
- Rédaction CdC fonctionnel (en 2 jours) :
  - Utiliser du libre, min de développement, V1 rapide.
  - Prévoit une architecture avec 3/4 composants.
- Choix des briques : 3 mois (maquette continue).
  - Étude des produits libres existants : 5 produits retenus.
  - Tests et décision : CMS Drupal (un seul composant) et 30 modules.

Document à lire : choix CMS - http ://www.projet-plume.org/fr/ressource/description-du-processus-de-choix-dun-cms-pour-le-projet-plume

- Développement (2/4)

- Rédaction nouveau CdC : 3 jours
  - Reprend le 1er CdC et on trie selon les fonctionnalités de Drupal :
    - Facile → on fait
    - Difficile et pas besoin fondamental → version suivante
    - Difficile et besoin fondamental → repense, contourne
  - Décrit 2 phases précisément et des idées pour les versions suivantes.
- Développement V1 : 4 mois
  Plutôt installation et configuration (workflow...)
- Transfert maquette vers Drupal : 2 mois
  Avec rédaction de pages, menus, mots-clés, ...
- Recette : documentation, à chaque fin de version.

- Développement (3/4)

- Ouverture du serveur (nov 07)
- Développement V2 : sur 9 mois
  - Classement des fonctionnalités (selon besoin et facilité)
  - Progressivement avec machine de développement
  - Intégration régulière des nouvelles fonctionnalités sur le serveur de production
- Recette : documentation
- Dév V3 (identité visuelle, site EN) : 6 mois
- Recette : documentation
- Contacts permanents entre 'MOA' et 'MOE'
  - La MOA met la main à la configuration

- Développement (4/4)

- C'était la bonne méthode, efficace et très rapide !
- Objectifs clairs et bien définis, écrits très simplement dans le CdC
- On a eu des surprises par rapport au CdC V1 :
  Drupal, totalement inconnu pour les développeurs, a les 3 briques
- Les problèmes actuels viennent :
  - des développements faits (qd MAJ de modules Drupal)
  - de processus trop compliqués avec du dév (relecture)
  - il faut faire simple avec minimum de développement
  - stabilité de la plate-forme : blocage : conf ? Robots ?, ...
- Et passage Drupal V5 à V6 : 8 mois ETP :
  - beaucoup de configurations à refaire, ...
  - prévoir les implications des évolutions de Drupal (roadmap)

- Architecture de la participation

PLUME fonctionne donc selon une structure pyramidale :

- 1 rédacteur en chef,
- 25 responsables thématiques :
  validation et gestion de fiches selon des domaines de compétence,
- 676 contributeurs : écriture, relecture des fiches,
- répond aux besoins de plus de 1500 membres.

Et en plus :

- (6) un comité d'exploitation : plate-forme et autres tâches techniques,
- (25) un comité technique : rassemble les responsables thématiques,
- (8) un comité de suivi : propositions, aide aux décisions.

4 PLUME

5 Aspects juridiques

- Réflexion sur la définition
- Le droit d'auteur du logiciel
- Les licences de logiciels
- Les types de licences de logiciels
- Mettre en place une licence
- Choisir une licence
- La liberté de diffusion
- Aux USA et dans le monde
- Architecture de licences
- Tableau recapitulatif

6 eXtreme Programing : méthode de développement

Qu'entend-on par *logiciel* ?

Logiciel en termes juridiques

Selon l'arrêté du Ministre de l'Industrie du 22 décembre 1981 relatif à l'enrichissement du vocabulaire de l'informatique :

Ensemble des programmes, procédés et règles, et éventuellement de la documentation, relatifs au fonctionnement d'un ensemble de traitement de données.

C'est donc un concept large, qui contient le code source, le code compilé et qui peut contenir la documentation.

D'un point de vue légal, un logiciel est une œuvre de l'esprit, avec un titre, des auteurs et des droits associés.

Le droit d'auteur du logiciel

Dès lors qu'on parle de loi, on parle des pays et de conventions internationales, mais où sont les frontières du Web ?

En France, un logiciel est protégé par le Code de la propriété intellectuelle (CPI), voir http ://www.legifrance.gouv.fr/.

Le cadre légal international est donné par la Convention de Berne pour la protection des œuvres littéraires et artistiques.
Cette convention est gérée actuellement par l'Organisation mondiale de la propriété intellectuelle (OMPI).

Deux questions importantes :

- qui est l'auteur ?
- de quelle œuvre ?

Le droit d'auteur (1/2)

Les droits protégés par le Code de la propriété intellectuelle (CPI) sont automatiquement associés à l'auteur lors de la création de l'œuvre, sous condition de son **originalité** (ceci dépend de la date).

L'œuvre doit être **mise en forme** : les idées, les concepts ne sont pas protégeables.

Deux types de droits associés : droits moraux et droits patrimoniaux.

Droits moraux : ce sont des droits imprescriptibles, inaliénables, incessibles, ils sont en général associés à des personnes physiques (les auteurs ou ses héritiers). Il y en quatre :

- Droit à la paternité, relatif à la mention de l'auteur.
- Droit de divulgation, relatif au moment et aux conditions de livraison.
- Droit de repentir, permet de retirer une œuvre.
- Droit au respect de l'œuvre, permet de s'opposer aux modifications.

Le droit d'auteur (2/2)

Droits patrimoniaux : concernent l'exploitation de l'œuvre, ce sont des droits monnayables, cessibles, temporaires.

On considére qu'il y a deux types d'exploitation :

- la représentation (par exemple d'une œuvre de théatre) et
- la reproduction (musique sur CD par exemple).

Ce sont des droits associés souvent à des personnes morales (suite à des cessions effectuées par les auteurs), on parle alors des **détenteurs** des droits patrimoniaux, ou des **propriétaires**.

- Œuvres orphelines :
  il n'y a plus de personne physique associée aux droits moraux.
- Œuvres de domaine public :
  fin des droits patrimoniaux, 70 ans après décès auteur.
  Note : ce terme est parfois (mal) utilisé dans le cadre de LL.

Le droit d'auteur du logiciel : exception

Pour les logiciels, il y a des **exceptions** aux règles générales :

- L'auteur ne peut (sauf stipulations contraires) s'opposer à la modification de l'œuvre ou exercer son droit de retrait.
- Les droits patrimoniaux (sauf stipulations contraires) sont dévolus à l'employeur. Cela s'applique aussi à leur documentation.
- La durée des droits patrimoniaux est de 50 ans (après décès de l'auteur).

Les détenteurs des droits patrimoniaux (propriétaires) d'un logiciel sont établis en fonction de :

- les auteurs
- leur statut (ou mode de collaboration)
- les contrats (employeurs, collaboration)

Les licences de logiciel (1/2)

Qui peut utiliser un logiciel ?

Art. L. 335-2 du CPI

Toute personne utilisant, copiant, modifiant ou diffusant le logiciel sans autorisation explicite du détenteurs des droits patrimoniaux est coupable de **contrefaçon** et passible de trois ans d'emprisonnement et de 300000 euros d'amende.

Les licences sont des **contrats** et protègent les auteurs, les utilisateurs et les éventuels collaborateurs au développement.

S'il n'y a pas de licence, le CPI (en France) s'applique de façon stricte.

Le CPI parle de cession de droits, pas de licences, mais cela ne pose pas de problème en pratique (jurisprudence).

Les licences de logiciel (2/2)

Les licences font intervenir des droits patrimomiaux, ce sont les propriétaires du logiciel qui en décident.

Les contrats sont activés (même s'ils ne sont pas signés) dès lors que le logiciel est récupéré : avant d'utiliser un logiciel il faut s'assurer qu'on en a le droit.

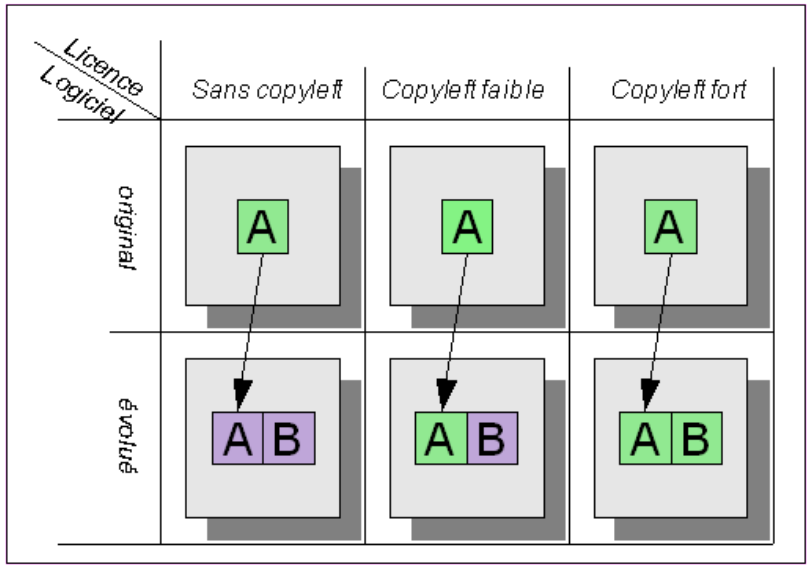
Deux types de licences : **libres** si elles respectent les 4 libertés indiquées dans la définition de la FSF, ou **propriétaires** dans le cas contraire.

Une licence ne peut pas être libre et propriètaire.
Pourquoi un logiciel peut être libre et propriétaire à la fois ?

Les licences peuvent entraîner des obligations sur l'utilisation du code comme brique de nouveau code.

Les types de licences libres

- Copyleft fort
  - Licence initiale s'impose sur tout.
  - Licence contaminante.
- Copyleft faible
  - Licence initiale reste.
  - Ajouts peuvent avoir autre licence.
- Sans Copyleft
  - Licence initiale ne s'impose pas.
  - Les dérivés peuvent avoir n'importe quelle licence.



GPLv2 : « *You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.* »

Mettre en place une licence

La licence doit être mise en place **avant la diffusion** du logiciel.
Attention aux cahiers des charges et aux contrats (clauses PI, licences).

En-tête pour tous les fichiers :

- Nom du fichier, nom du logiciel
- Copyright (©, Droits patrimoniaux), année(s), p. morale ou physique
- Auteur(s)
- Licence
- Utiles : date de création, une adresse de contact

Et en plus :

- Ajouter un fichier de licence au package des fichiers source avec le texte complet ou une URL
- Indiquer les briques logicielles utilisées et leurs licences
- Indiquer la licence (et les auteurs) dans la doc, sur le site Web
- Donner des licences aux documentations (par ex. GNU FDL, CC, LAL).

Choisir une licence

Le choix peut se faire selon le :

- droit anglo-saxon : GNU GPL et autres licences,
  http ://www.gnu.org/licenses/
- droit européen : EUPL (European Union Public Licence)
  http ://www.osor.eu/eupl
- droit français : CeCILL (CEA, CNRS, INRIA), http ://www.cecill.info/

Validité de GPL en France ?, de CeCILL aux USA ? Langue du contrat ?

Exemples de licences selon le degré de copyleft :

- sans copyleft : Apache, BSD, MIT, CeCILL-B
- avec copyleft faible : MPL, GNU LGPL, CeCILL-C
- avec copyleft fort : GNU GPL, CeCILL v2

On peut utiliser plusieurs licences, l'utilisateur choisit.

La liberté de diffusion : Affero GPL

La clause de copyleft fort de la GPL (dite aussi virale ou contaminante) empêche qu'un code GPL devienne un code exclusivement propriétaire, c'est la licence la plus utilisée (70 % des LL).
Mais les modifications d'un code GPL peuvent ne pas être diffusées (4ième liberté d'un logiciel libre).

Ce concept de diffusion, parfois mal défini juridiquement, entre en jeu lors de litiges comme celui de la FreeBox en France (en cours).

Les codes non-diffusés ne reviennent pas à la communauté et ne contribuent pas aux évolutions du bien commun. (Exemple ?)

Pour réagir sur ce problème, la licence GNU Affero GPL a été mise en place et indique des clauses contaminantes pour des codes utilisés en réseau.
Réponse légale à un modéle économique.

Aux USA et dans le monde

Dans le monde anglo-saxon il existe une loi sur le *Copyright*, concept qui ne doit pas être confondu avec celui des droits patrimoniaux, qui dans le monde du logiciel libre a donné, par réaction, le concept de *copyleft*.

Le 13 août 2008, la cour d'appel des États-Unis affirme :

- que les licences libres sont bien plus que des contrats
- que le non-respect d'une licence libre constitue une violation de la loi sur le copyright

et en conséquence, les réparations en cas d'infraction ne se mesureront pas seulement à la hauteur d'éventuelles pertes financières.

Les logiciels libres apportent un réel bénéfice à la société, comme cela peut être évalué en termes financiers et de marché ?

En général il y a peu de contentieux : re-codage, éviter mauvaise réputation. Même si le cadre jurique est en évolution, **en pratique le modèle des licences libres fonctionne**.

Architecture de licences

Les licences libres indiquent des obligations, pas seulement des libertés. Quand il y a beaucoup de briques logicielles, il peut être difficile de comprendre toutes les licences des briques et il y a deux questions avec des conséquences légales : la compatibilité et l'héritage des licences.

Des logiciels pour analyser les licences :

- FOSSology, http ://www.projet-plume.org/fiche/fossology
- OSLC, http ://www.projet-plume.org/fiche/oslc
- payants : Black Duck, Palamida, ...

Des standards pour indiquer les informations des licences et aider le travail automatique des logiciels sont en cours d'étude par un groupe international (avec liste de 100 licences) :

- SPDX, http ://www.projet-plume.org/ressource/spdx

On étudie aujourd'hui les licences à utiliser en 2030...

Tableau recapitulatif

Pour mieux comprendre le droit d'auteur et les licences des logiciels.

Aspects légaux		
	Article	**Logiciel**
Code PI	œuvre protégée	œuvre protégée + exception
Œuvre	article	code source, objet, doc, ...
Auteurs	même %	% de participation, pb. legal
Propriétaires	auteurs	tutelles + autres (stage, contrats)
Dates	soumission, publi	matériel prép. + versions
Évolution	nouv. publi	nouv. œuvre ?, revoir auteurs, dates
Travaux préc.	références	briques + compatibilité, héritage lic.
Diffusion	éditeur, web	web, forges, besoin de licence
Droits	lire, pas copier	lire, pas utiliser, pas modif, ..., lic.
Licences	CC (web)	libres, propriétaires

Génie logiciel

Ensemble d'activités conduisant à la production d'un logiciel.

Nombreux processus de développement :

- modèle en cascade
- modèle en V
- UP (Unified Process), ...
- et méthodes agiles, XP (eXtreme Programming)

Choix en fonction de l'organisation des développeurs, du type de logiciel, des personnes impliquées.

4 activités communes :

- définition des spécifications
- conception et implémentation
- test et validation
- évolution

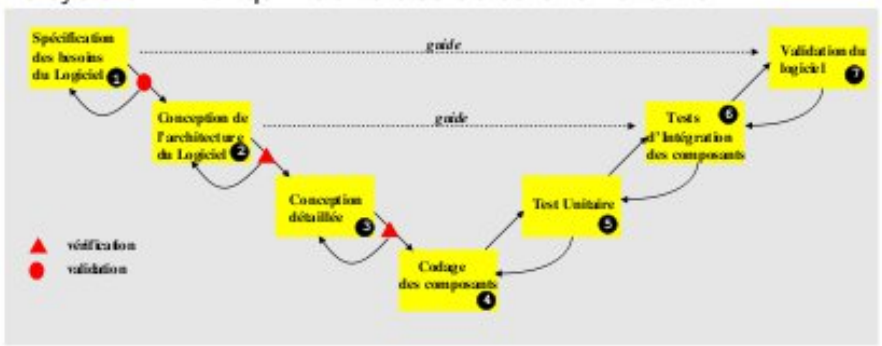
Méthodes agiles : motivation

Contexte économique actuel :

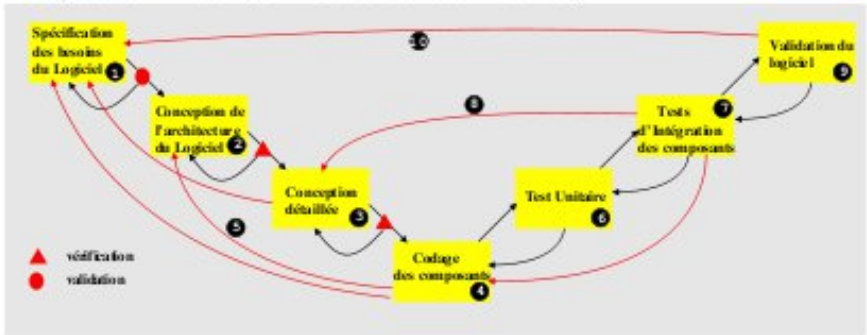
- Économie des moyens humains, financiers...
- Efficacité / besoins : date mise à disposition
- Réactivité : gestion du changement continu

Méthode de développement avec cycle en V mal adaptée :
Besoins → spécifications → conception générale → détaillée → codage → tests unitaires → intégration → validation → et enfin livraison
lourdeur du modèle, manque de souplesse.

Le cycle en V tel qu'il devrait se dérouler en théorie



Le cycle en V tel qu'il se déroule en réalité,



eXtrem Programming

En 2001, 17 personnalités du Genie Logiciel signent l'Agile Manifesto.
Ce manifeste fonde eXtreme Programming, qui est :

- un processus de développement
- un état d'esprit et des valeurs
- un ensemble de bonnes pratiques

Les principes :

- Le client travaille avec les développeurs (en binôme)
- Le développement est itératif, itérations courtes ($\sim$ 2 semaines)
- Les développeurs sont polyvalents et inter-opérables (compétences et connaissances)
- Chaque composant est vérifié par une campagne de tests automatisés - Programmation pilotée par les tests
- Chaque itération est validée par le client

XP : les valeurs

Les 5 valeurs de XP :

- La communication (entre les personnes)
  - Evite la plupart des problèmes
  - Travail avec client, en binôme, réunions de code ...
- La simplicité (des solutions)
  - Meilleur moyen d'arriver à un résultat, évolution facile
  - Faire simple, ne pas anticiper les extensions futures
- Le feedback (avoir des retours réguliers)
  - Tests unitaires, fonctionnels, versions régulières (retours client)
- Le courage
  - De changer d'architecture, de jeter du code : faire mieux, plus simple
- Le respect
  - Mutuel des membres, du travail

XP : les pratiques

- Une représentation du client sur site de développement
- Planning établi par le client et l'équipe de développement
- Intégration continue (à chaque itération)
- Petites livraisons fréquentes
- Rythme soutenable (pour les développeurs)
- Tests fonctionnels à chaque itération
- Tests unitaires / programme
- Conception simple, doc minimale
- Utilisation de métaphores pour expliquer
- Refactoring régulier
- Appropriation collective du code
- Convention de nommage
- Programmation en binôme

XP : bilan

Points négatifs de XP :

- Dogmatique : il faut appliquer tous les principes
- Difficile avec un prestataire (MOA et MOE)
  - Quel contrat ? Quel appel d'offre ?
  - Mal accepté par certains prestataires de services
- Importance primordiale du représentant du client

Points positifs de XP :

- Agile (léger, adaptatif), 1ère version rapide
- Bien adapté au développement libre
  - Ex : Code_Aster (EDF) : analyse des structures et thermo-mécanique
  - 1,5 Millions de lignes http ://www.projet-plume.org/fr/fiche/code-aster
- Très utilisé pour les sites web
- On peut reprendre des principes 'de bon sens'. Ex. PLUME

Document : F. Miller, eXtreme Programming, vers plus d'agilité (ENVOL 2008).

Liste d'outils

Les indispensables de base :

- Les licences (GPL en 89)
- Internet (devenu utilisable en 90) :
  - de recherche
  - de diffusion : instantanée
  - (presque) partout, peu cher
  - permet travail et communication entre spécialistes éloignés

Nécessaire : méthode de développement (agile ?, adaptée au projet).

Les bons outils :

- aide au développement :
  gestion de versions, gestion de bugs, tickets, ...
  forges, Environnement de développement intégré (IDE), ...
- communication :
  site web, listes de communication, wiki, ...

Ces outils sont souvent intégrés dans des forges.

Outils de gestion de versions (1/2)

Buts :

- Historique des modifications d'un ensemble de fichiers
- Travail en équipe (verrous, gestion des conflits) mais aussi pour une seule personne
- Modifications en parallèle (branches)
- Sécurité (intégrité, disponibilité, confidentialité)
- Utile pour logiciels, documentation, fichiers de configuration, et plus généralement pour toute sorte de documents
- Souvent est un service intégré dans une forge

Vocabulaire : versions diffusées d'un logiciel

- version stable, pour utilisateurs
- version beta, pour développeurs, beta-testeurs,
- conventions de nommage (numéro de version).

Outils de gestion de versions (2/2)

Vocabulaire : la gestion des versions des fichiers

- Patch : V2 - V1 = Patch et V1 + Patch = V2
- Dépot (repository) : répertoire partagé par tous (contient historique)
- Révision : chaque fichier a un numéro de révision unique
- Commit : soumission d'ajout et/ou modification
- Update : mise à jour d'un fichier/ensemble de fichiers
- Branches : arborescence de versions du projet,
  on peut gérer plusieurs branches en parallèle, fusionner, ...

3 types :

- Local : SCCS, RCS, système de fichiers local (pas de réseau),
- Centralisé ou Client/Serveur : CVS, évolution en Subversion
  Un serveur centralise le dépôt, accessible à distance.
- Distribué : bitkeeper, monotone, arch, darcs, mercurial, git, bazaar
  Multiples copies du dépôt, branches locales.

Forges (1/2)

Définition

Une forge ou plate-forme d'hébergement de projets logiciels désigne un environnement Web constitué d'un ensemble d'outils du travail coopératif et du génie logiciel pour le développement collaboratif et distribué de logiciels.
C'est aussi le logiciel qui permet de créer la plate-forme.

On peut :

- Créer un espace de travail (un projet)
  - avec un administrateur et plusieurs contributeurs
- (optionnel) Gérer du code
  - gestion des sources, des versions, des bugs, des tickets
  - gestion des documents (Wiki), forums, listes, sondages, ...
- (optionnel) Diffuser (faire connaître)
  - communication : captures d'écran, pages descriptives, news, ...

Forges (2/2)

On y trouve donc des projets, avec des outils de recherche, des interfaces pour poser des questions, demander de l'aide, contacter des personnes...

Avec des règles : réservé à une communauté, pour certaines licences, ...

Quelques exemples :

- Sourceforge, US : http ://sourceforge.net/
  - Le plus connu : plus de 180 000 projets
  - Gratuit, ouvert à tous les développements LL
  - Très peu de gouvernance
  - On ne détruit rien : 90 % des projets sont morts
- SourceSup, FR : http ://sourcesup.cru.fr/
  - Réservé Ens Sup et Recherche : forge de communauté
- INRIA, FR : http ://gforge.inria.fr/
  - Réservé INRIA : forge d'entreprise
  - Permet de 'contrôler' et conseiller les développeurs

Logiciels pour faire des forges

Si on ne trouve pas la forge idéale, on peut l'installer.

Au départ un logiciel libre : SourceForge.
Après des forks, on a des versions qui évoluent de façon indépendante :

- une devient propriétaire, SourceForge Enterprise Edition
- l'autre version reste libre, et a donné plusieurs projets :
  - Gforge : GPL, http ://www.gforge.org
  - Codendi
  - Savane
  - Fusionforge

Vocabulaire : fourche ou *fork*, créer une version indépendante d'un logiciel libre, d'un projet.

Document à lire : FAQ : hébergement des développements logiciels de laboratoire : forges, http ://www.projet-plume.org/ressource/faq-forge

Les autres outils : utiliser des briques libres

Pourquoi utiliser des briques libres ?

- ne pas ré-inventer la roue,
  - sauf s'il y a un besoin de comprendre, de tester, ...
- on ne peut pas avoir toutes les compétences,
- c'est gratuit.

Comment choisir les briques libres ?

- Fonctionnalités, évidemment
- Selon ses compétences : C++, Java, PHP, ... ?
- Respect des standards : formats, interfaces
- Pérennité :
  - Qui développe ? Une communauté qui fonctionne.
  - Comment ? Un projet qui évolue dans des bonnes conditions.
  - Un projet qui est utile et utilisé, (quantité, qualité) ?
  - Avec quelle licence ?
- Architecture : des briques, des licences (compatibilité, héritage).

Classement des problèmes

Les problèmes auxquels on peut être confronté lors du développement d'un logiciel peuvent être classés selon différentes catégories :

- formation, support
- techniques : outils, briques, méthodes de développement, ...
- juridiques : licences, droits d'auteurs, brevets, contrats, ...
- économiques : financement, modèle économique d'exploitation, ...
- administratifs : dépôt APP, ...
- communauté : gestion, prise de décisions, communication interne, résolution de conflicts, ...
- politiques/législatifs : définition de standards, brevets, ...

Ces problèmes sont liés au développement et non pas aux types de logiciels développés, libres ou propriétaires.

7 Développement : les bons outils

8 Développeurs : qui ?, comment ?, pourquoi ?, où ?

- Qui ?
- Pourquoi ?
- Comment ?
- Lancer un projet
- Où sont-ils/elles ?

9 Les sociétés et les métiers autour des LL

10 Migration vers les logiciels libres

11 Les points forts et faibles des LL

Développeurs : qui ?

Du point de vue légal, on regarde si le développement est lié à l'activité professionnelle (par ex. si on utilise les moyens de l'entreprise, ...).

Particuliers, activité professionnelle autre

- Passionnés de la programmation : *gourou-hacker-geek*
- Étudiants (concours google, ...)

Employés, activité professionnelle liée

- Sans connaissance de l'employeur (devient officiel, + de contrats)
- Monde académique : chercheurs-enseignants-ingénieurs (PLUME)
- Certaines administrations (bien commun)
  - ADULLACT (http ://www.adullact.org/),
    patrimoine commun LL utiles aux missions de service public
- Entreprises : de plus en plus (+ de contrats)
  - Services internes, Services de R&D
  - Sociétés de services en informatique, Éditeurs informatiques, ...

Développeurs : pourquoi ?

Il y a des motivations très diverses (autres que économiques) pour faire un nouveau logiciel libre ou contribuer aux évolutions d'un logiciel existant.
Professionnelles : répondre aux besoins

- de son métier
- de son entreprise, d'un client (sociétés services informatiques)

Personnelles :

- pour résoudre un problème, un besoin, une envie
- pour développer (création, parfois aspects artistiques)
- apprendre, comprendre (techniquement, science)
- échanger, être ensemble, mutualiser effort (communauté)
- éthique : donner au monde, bien public, patrimoine
- idéologiques : anti-propriétaire (Windows, SAP, ORACLE, ...)
- valorisation personnelle en dehors de son travail
- projet avec succès

Développeurs : comment ? (1/2)

La production de LL est de plus en plus organisée, indépendamement du mode de collaboration : privé ou professionnel.

- Parce qu'on connait mieux les bénefices, la production, les aspects légaux, ...
- Parce que les codes évoluent et les projets sont plus complexes.

Les méthodes de développement sont souvent peu théorisées : pas de MOA-MOE, pas de cahier des charges fonctionnel, ...

Elles répondent souvent au bon sens, au pragmatisme, et parfois à l'économie de moyens et au besoin de réactivité : elles sont souvent proches des méthodes agiles.

Document : évolution SPIP http ://vimeo.com/3078702

Développeurs : comment ? (2/2)

Où est passé le bazar ?

- Un développeur seul
  - des utilisateurs remontent des bugs, des demandes de nouvelles fonctionnalités, des idées, ...
- Une équipe : N développeurs, 1 responsable
  - des développeurs (contributeurs) proposent du nouveau code pour améliorer l'existant
  - le responsable l'intègre ou pas, gère les contributeurs, définit les évolutions et les versions
- Une arborescence hiérarchique de contributeurs et de responsables
  Exemple du noyau Linux : +2000 contributeurs, 300 mainteneurs de pilote, mainteneurs de ss-système, mainteneurs principaux

  → les mainteneurs décident et intégrent le code des contributeurs.

Développeurs : lancer un projet (1/2)

- connaître l'existant, ne pas ré-inventer la roue
- bien définir les objectifs
- faire un CdC avec des priorités
- choisir un joli nom, sympa, facile à retenir
- avoir une stratégie d'avancement : feuille de route, phases
- déterminer les outils (forge, bugs, ...)
- suivre et respecter les standards
- choisir les briques à utiliser, vérifier si elles sont libres
- choisir une licence
- établir les outils de communication : site web, listes, wiki, autres
- à chaque étape :
  - établir la documentation
  - tenir à jour les avancements : fait/à faire

Développeurs : lancer un projet (2/2)

Les personnes :

- ne pas rester isolé :
  participer à une communauté, ou penser, créer communauté
- penser aux nouveaux contributeurs (com, doc)
- prévoir la gestion des contributeurs
- prévoir la gestion des problèmes internes

Prévoir la méthode :

- diffuser une première version simple
- être agile, évoluer selon les retours

Le moment de la diffusion est clé :

- une première phase interne, les suivantes publiques
- licence, documentation, site web, adresse de contact, ...

Document : F. Pellegrini, Développer en logiciel libre : empaquetage et diffusion (ENVOL 2008)

Développeurs : où sont-ils/elles ? (1/2)

Les logiciels se développent principalement en deux types de milieu :

- associatif, avec plus ou moins de personnes, d'organisation,
- en entreprise, avec une équipe interne qui fait le développement.

Ces deux milieux et modèles, en principe différents et indépendants, peuvent évoluer pour se rejoindre :

- si le logiciel évolue, le milieu initial associatif peut devenir une entreprise, avec des collaborateurs externes qui continuent à s'impliquer dans le projet,
- l'entreprise peut chercher à s'appuyer sur un ensemble d'utilisateurs externes, qui deviennent des contributeurs chouchoutés par l'entreprise ; ils ne sont pas payés mais retrouvent un bénéfice (influence, reconnaissance, bien commun).

Développeurs : où sont-ils/elles ? (2/2)

Il y a aussi le milieu académique avec un fonctionnent proche du milieu associatif. Les personnels académiques collaborent aux logiciels (libres ou propriétaires) faits en milieu associatif ou entreprise. La motivation principale est de faire évoluer la science.

Le milieu associatif initial peut aussi évoluer et s'organiser dans des *fondations* ou associations non gouvernementales, dans des consortiums pour structurer des ensembles de personnes, institutions et/ou services, pouvoir payer des permanents, demander des financements aux institutions, organiser des évènements, ...
Par exemple OW2 (http ://www.ow2.org).

Finalement des grandes entreprises soutiennent et s'impliquent dans des LL pour des raisons stratégiques et de marché (ou autre) :
par exemple IBM et Eclipse, Google et Mozilla, SUN et OpenOffice, ...
Note : SUN acheté par ORACLE en 2010, évolutions de OpenOffice ?

Les sociétés de services informatiques et LL

Définition

Le service informatique consiste à vendre du temps et de l'expertise.

Donc vrai aussi pour les logiciels libres : SSLL (sociétés de services en LL).

Expertise sur :

- sécurité
- internet
- développement Web
- infrastructure
- réseau
- serveurs (fichiers, mail, ...)
- bureautique
- multimedia
- gestion
- juridique, ...

sous forme de :

- conseil,
- formation,
- installation,
- support,
- infogérance,
- intégration (choix),
- adaptation,
- développement,
- certification,
- migration, ...

Métiers autour de LL (1/2)

- Hébergeur (matériel, application, information, service, ...)
  - Utilisation LL pour infrastructure, admin, surveillance, ...
- Vendeur de matériel
  Par exemple : Linux embarqué avec piles logicielles standard (IP, HTTP, audio-video...)
- Éditeur de logiciels libres
  services associés : maintenance (MAJ), support, formation
- Éditeur de logiciels propriétaires :
  - double licence : libre mais aussi propriétaire avec services associés
    exemples : MySQL, Zimbra...
  - utilisation de briques libres (Java, Apache...) dans des produits propriétaires

Métiers autour de LL (2/2)

- Éditeur de distributions linux
  exemples : RedHat, Suse, Novell, Ubuntu, Mandriva, CentOS, ...
- Société de formation
- Métier juridique
  - Formation, conseil sur les licences, ...
  - De plus en plus important, la contractualisation
- Agence de communication
  - Promouvoir les projets LL, organisation d'évènements, ...

Migration vers LL - Introduction (1/2)

Beaucoup d'entreprises et administrations font, ont fait ou essayé, ou préparent en ce moment une migration vers des logiciels libres.

Par exemple de grandes administrations françaises ont décidé d'utiliser la suite bureautique libre OpenOffice.org :

- le ministère de l'Agriculture et de la Pêche,
- le ministère de l'Équipement,
- le ministère de l'Intérieur,
- la Direction générale des douanes,
- la Direction générale des impôts.

La Gendarmerie nationale française (utilisatrice d'OpenOffice depuis 2005) a choisi de migrer 70 000 postes clients sous l'environnement GNU/Linux d'ici 2013.

De même, l'Assemblée nationale a fait en mars 2007 le choix de Linux pour l'équipement informatique de ses députés, avec une économie estimée à 500000 euros.

Migration vers LL - Introduction (2/2)

Cela veut dire qu'il y a des vraies économies, et qu'on peut faire des migrations avec succès.

Mais beaucoup de migrations n'ont pas eu de succès, ou pire elles ont été un échec total, ce qui a comme conséquence une mauvaise réputation pour les logiciels libres **en général**.

Il y a quatre facteurs importants à prendre en compte dans une migration :

- la décision : adaptée à la solution libre choisie, aux besoins locales
- le côté technique : installation, test et validation, et adaption/évolution (*patch*)
- la procédure à suivre : étapes, bonnes pratiques
- le facteur social : besoin d'accompagnement, communication

Une mauvaise migration est souvent liée à la trop grande place reservée aux aspects techniques et une certaine négligence du facteur social, de l'étude préalable d'une bonne procédure de migration et de sa mise en œuvre.

Migration : une bonne procédure doit ... (1/2)

- s'assurer que le logiciel répond aux besoins
- s'assurer de l'accord et du soutien des responsables pendant toute la procédure : éviter de revenir en arrière au moindre problème
- avoir une vision claire des bénefices attendus, éviter le 'c'est gratuit'
- avoir un délai raisonnable pour effectuer la migration
- revoir et adapter les méthodes d'acquisition du matériel et des logiciels, qui peuvent changer de l'achat vers l'achat de support
- vérifier que l'acquisition de LL passe par les mêmes phases que les logiciels propriétaires (peut éviter des critiques à posteriori)
- éviter les gros changements, faire des migrations incrementales
- chercher du conseil et des informations sur des procédures similaires d'adoption de LL dans d'autres entités

Migration : une bonne procédure doit ... (2/2)

- adopter aussi des techniques de évaluation/comparaison du monde LL (FLOSSMETRICS, ...)
- vérifier, mésurer les différences entre ce dont on a besoin et ce qu'on a
- vérifier que les logiciels sont libres
- identifier fonctionnalités, pas les noms (marques)
- identifier les besoins de support : seulement un support de la communauté du LL ?, un support plus adapté à votre situation fait par une société spécialisée ?
- rester pragmatique : cela peut être satisfaisant de jeter tous les logiciels propriétaires par la fenêtre, mais on peut en avoir besoin ...

Migration : les aspects sociaux

- informer les utilisateurs sur les étapes, la valeur ajoutée (pas seulement des économies, mais aussi qualité, ...), les bénéfices attendus.
- utiliser la migration pour promouvoir l'acquisition de nouvelles compétences
- communication :
  - faire connaître la feuille de route pour la migration, ses étapes
  - établir un blog, wiki, ... pour encourager la communication, faciliter les questions des utilisateurs
- encourager l'expérimentation, faciliter l'adoption (avec une plateforme de test)
- identifier les experts, les encourager à migrer avant, à acquérir et partager des nouvelles compétences

Document : http ://www.openworldforum.org/attend/speakers/carlo_daffara-open_source_adoption.pdf

Les points forts des LL (1/4)

- Liberté par rapport aux logiciels propriétaires
  - obligation d'acheter windows ? licence pour le nouveau poste, licence pour la migration vers une nouvelle version
  - nouvelle version de SAP (plusieurs mois de travail, attendre une nouvelle version pour la fonctionnalité dont on a besoin)
  - que se passe-t-il avec vos données si vous souhaitez abandonner un logiciel propriétaire ?
  - quelle est la vraie valeur d'une licence propriétaire ?
- La majorité des licences impliquent une utilisation gratuite : gain d'argent et de temps
  - coût de base souvent négligéable, on fait des économies
  - possibilité d'essai/test rapide
  - 'achat' immmediat (évite des longues procédures dans des administrations et grandes entreprises)
  - possibilité d'utilisation rapide

Les points forts des LL (2/4)

- Le code est lisible : peut être vérifié, certifié
  - comprendre le code
  - s'assurer de la sécurité, pas de cheval de Troie caché, ...
  - détecter les bugs : les signaler, les corriger
  - s'assurer qu'il n'a pas de faille

  → le LL apporte compréhension, sécurité, fiabilité, certification

  → pas de sécurité, de certification si le code n'est pas ouvert
- Le code est lisible : peut être modifié
  - correction des erreurs soi-même
  - modifications locales
  - si intégrées à la version stable : répondent à des vrais besoins

  → capacité d'adaptation rapide aux besoins du 'client'
- Le code est lisible : transparence
  - comparaison de deux produits, leur codage, leur conception ...
  - les meilleurs restent et concentrent les efforts !

  → qualité

Les points forts des LL (3/4)

- Diffusion rapide des versions
  - Time to market très court
  - En avance par rapport aux produits commerciaux
  - Le produit est plus adapté au marché, le suit de près

  → innovation ! + remise en question de l'industrie du logiciel
- Ethique
  - Le logiciel est un bien public, fait partie du patrimoine à conserver
  - Le logiciel doit être partagé sans contrainte
  - Mouvement FSF - Richard Stallman
- Ré-utilisation du code
  - Pour faire d'autres développements
  - Gratuit
  - Assure une pérennité du code d'origine
  - Effet 'standard' : nouveaux logiciels seront 'compatibles'
  - Fork possible si l'évolution ne va pas dans le sens désiré (dév, util.)
    → Garantie d'évolution (pas le cas de SAP, ORACLE...)

Les points forts des LL (4/4)

Comparaison avec la chaîne de production d'un logiciel propriétaire

- Étude de besoins (service marketing)
- Étude du marché (service marketing)
- Décision de lancer le développement
- Spécifications fonctionnelles (service marketing)
- Spécifications détaillées (service développement)
- Développement (service développement)
- Tests et validation (service test-validation)
- Sortie d'une version, complète et avec peu de bugs
- Vente de cette version (service commercial)
- Un premier client utilise le logiciel 'en vrai'

Total : un ou plusieurs années ... le produit n'est plus innovant.

Les points faibles et risques des LL (1/2)

- Remise en question de l'industrie du logiciel
  ses modèles économiques et ses méthodes de développement : résistance aux changements
- Pérennité
  si les contributeurs principaux disparaissent, se marient, changent de job, ne sont plus intéressés ...
- Perte de motivation des contributeurs
  problèmes de reconnaissance, d'encadrément, si utilisation non souhaitée de leur travail ...
- Il faut que les gens mangent
  les acteurs du libre doivent gagner leur vie, ce qui donne des modèles économiques variés, pas tous vont survivre ...

Les points faibles et risques des LL (2/2)

- Eparpillement de certains efforts :
  - Distributions Linux trop nombreuses avec des communautés trop concurrentes
- Créer un 'monde nouveau', gratuit (Free software) vs. économique (Open source) :
  - Changement de dimension et d'objectif
  - Il faut rester pragmatiques
- Logiciels → services ? cloud computing
  - Services google : le navigateur est le seul logiciel dont on a besoin !
  - Les entreprises n'achétent que des services (calcul, stockage, mail, ...) : plus besoin de logiciel sur les postes

Comment et où trouver des logiciels libres ? (par exemple)

- Bouche à oreille
- Google (et si on ne connait pas le nom de la solution cherchée ?)
- Framasoft http ://www.framasoft.org :
  - descriptifs, FR, généraliste, Windows surtout
- Wikipedia fr.wikipedia.org :
  - descriptifs, encyclopédie libre
- Ohloh ohloh.net, OSalt osalt.com
- PLUME http ://www.projet-plume.org :
  - descriptifs, FR, professionnel, orienté communauté ESR
- Developpez.com, pour développeurs
- FSF/UNESCO directory.fsf.org (descriptifs, patrimoine)
- Sourceforge sourceforge.net, autres forges
- Apache apache.org, licence Apache
- Distributions Linux (ex. Debian Science)

Informations, associations, évènements, ...

- Quelques informations sur les LL, les associations autour des LL : http ://www.projet-plume.org/ressource/plume-documentation-et-associations
- Open Source Observatory and Repository (EU), http ://www.osor.eu : on y touve la EUPL, une forge, un rassemblement des informations de forge en EU

Évènements en France : (par exemple)

- Solutions Linux http ://www.solutionslinux.fr
- RMLL http ://rmll.info
- OWF http ://www.openworldforum.org
- et des réunions regionales : JDLL par ALDIL à Lyon, ...

Conclusion (1/2)

Un logiciel libre c'est à la fois :

- une question de liberté (R. Stallman)
- du juridique (c'est l'origine de la définition)
  les licences donnent le droit d'utilisation, et de faire du commerce
- un objet : programme, avec le code source lisible
- un produit +/- fini qu'on utilise, que tout le monde peut utiliser
- un produit pour construire d'autres produits (architecture), qui est disponible pour tous
- une méthode de production : ré-utiliser ce qui existe, gérer des contributeurs...
- une économie : des sociétés qui en vivent
- une manière de vivre et une action sociale :
  - contributeurs et communauté : groupe
  - contributions gratuites : dons

Conclusion (2/2)

- Dans le monde des LL il y a beaucoup d'objectifs orthogonaux, ce qui implique des équilibres à trouver, mais cependant une certaine stabilité.
- Les fondateurs veillent aux dérives : évolution des licences, dénonciations, alertes médiatiques, production de documents de re-centrage, de bonnes pratiques, ...
- Les LL sont un fait, il y a une croissance de leur utilisation, ils sont reconnus comme des logiciels professionels et ils sont maintenant au cœur des innovations.
- (R)évolution pour les professionnels de l'informatique

Les économies émergentes

India's National Mission on Education : A Case Study
Alolita Sharma, OWF 2010

Projet pour établir et gèrer un environement LL pour l'éducation en mathématiques et autres sciences techniques.

$1 billion USD pour mettre en place le réseaux physique (cable, serveurs) et humain (formateurs), développer les compétences en LL et les matériels de cours.

- création et distribution des contenus éducatifs avec LL
- création des cours sur les outils LL en sciences
- augmenter l'utilisation de LL avec audio-vidéo et des laboratoires virtuels
- ...

Pour réflechir plus loin

- Michel Serres à Interstices :
  http ://interstices.info/jcms/c_16371/serres
- Eben Moglen, Professor of law and legal history at Columbia University
  The System of Ownership of Ideas (1 de 3) :
  http ://www.youtube.com/watch ?v=98vD6TTDpZY
- Noirin Shirley, Vice-President, Apache Software Foundation (USA), OWF 2010
- Prof Eben Moglen and Venkatesh Hariharan speaking in the workshop "Software Patents and the Commons", New Delhi, 1 Sept 2010.
  Eben Moglen and Venkatesh Hariharan - Part 1
  http ://www.youtube.com/watch ?v=JFW26VRthRo
  Eben Moglen and Venkatesh Hariharan - Part 2
  http ://www.youtube.com/watch ?v=nf3G2PmRyVY