Sensibilisation à la propriété intellectuelle appliquée au logiciel

Auteurs:
CPPIs et juristes du CRI Lille
avec modifications de **Indiana Zurbach** et **Maike Gilliot**

Objectifs de la formation

- Partager des principes/règles de base à connaître en propriété intellectuelle
- Présenter et sensibiliser aux enjeux en matière de propriété intellectuelle liés plus précisément **au processus de développement** et **au métier d'éditeur** d'Inria
- Identifier et présenter les rôles des différents acteurs en interne (chercheurs, ingénieurs SED/ITI, CPPI, juristes…)
- ✖ A contrario, l'objectif n'est pas de former au métier de juriste

Motivation: le rôle du logiciel

- Les logiciels produit par les chercheurs et les ingénieurs appartiennent à leurs employeurs.

- Un logiciel est une « valeur » pour une équipe:
  matérialisation (partielle) de leur savoir-faire et de leurs compétences

- Les logiciels sont objets de contrats et d'engagements
  - Projet collaboratif → droit d'utilisation des « connaissances antérieures »
  - Transfert → licence

→ De là la nécessité de connaitre et de veiller aux règles liés à l'utilisation de code (=licence)

PLAN

I – Introduction au droit d’auteur - principes de bases

II – Intentions et choix de licence

III – Incidence du développement en mode collaboratif

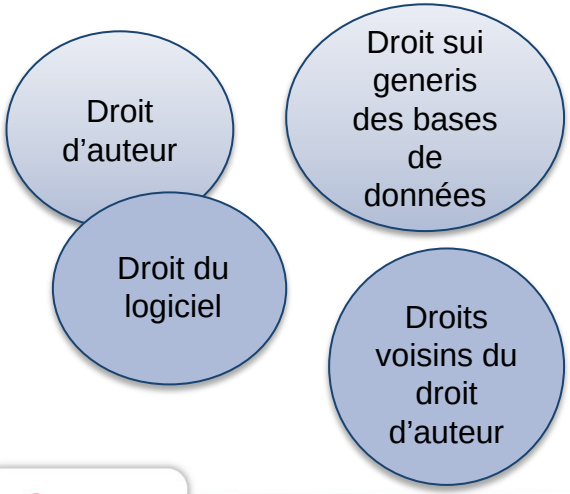
IV – Incidence de la réutilisation de code

Introduction au droit d’auteur – principes de base

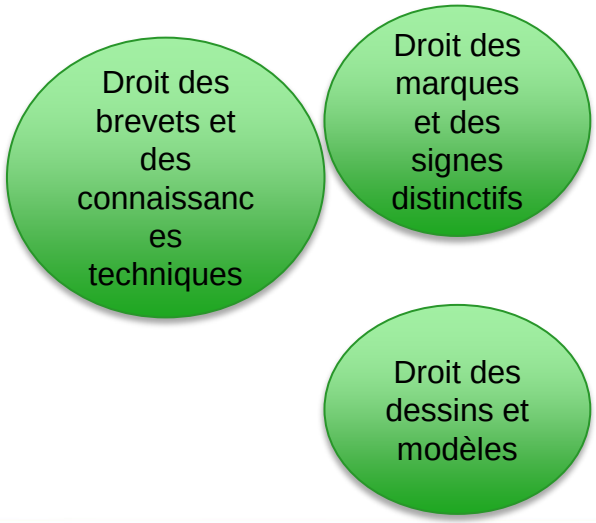
I – 1 le droit de la propriété intellectuelle, c'est quoi?

- Définit le ou les régimes juridiques applicables aux créations immatérielles (inventions, œuvres de l'esprit).
- Plus précisément, définit les conditions dans lesquelles ces créations peuvent être protégées juridiquement (et non techniquement: par exemple, DRM ou code source inaccessible pour les logiciels) et êtres exploitées

Droit de la propriété littéraire et artistique



Droit de la propriété industrielle



Alternative aux droits de PI: mise au secret

I – 2 Logiciel et propriété intellectuelle, quelle régime de protection juridique?

→ Les différents régimes de protection en droit de la PI existaient longtemps avant l'apparition des logiciels: Siècle des Lumières, Révolution française puis début de la révolution industrielle.

→ C'est dans les années 80, lorsque le logiciel devient une valeur économique en forte progression, que les européens s'intéressent à sa protection par la PI. Trois possibilité:

- Régime propre:
- Rattaché au droit d'auteur;
- Rattaché au droit des brevets (US).

→ **France**: Arrêt Babolat c/ Pachot du 7 mars 1986: originalité = effort personnalisé allant au-delà de la simple mise en œuvre; Puis L. 1994.

→ **Europe**: Directive européenne de 2009 venant harmoniser les législations des Etats membres.

→ Du fait de la particularité du logiciel par rapport aux autres œuvres régies par le droit d'auteur, certaines règles du droit d'auteur s'appliquent à lui de façon différentes et dérogatoire.

→ Du fait qu'il s'agit d'un objet « atypique » pour le droit d'auteur, il soulève des questions souvent plus complexes à traiter et à apprécier

I – 3 l’existence de lien étroit avec d’autres droits de propriété intellectuelle

→ Un logiciel n’est pas protégeable en tant que tel par voie de brevet (en Europe)

→ En revanche, une invention technique peut être implémentée dans un logiciel (dispositif, procédé…)

→ Les logiciels à Inria utilisent/intègrent souvent des données ou des bases de données, qui font elles-mêmes l’objet de disposition légales spécifiques

Cette présentation ne portera que sur les questions relatives au droit d’auteur

I – 4 Conditions de la protection

- **Originalité**: Effort personnalisé de l'auteur allant au-delà de la simple mise en œuvre d'une logique automatisée et contraignante et que la matérialisation de cet effort réside dans une structure individualisée (Ass. Plen. 7 mars 1986, SA Babolat c/ Pachot).
- **Sur un support tangible**.

I – 5 Objet de la protection du logiciel par le droit d'auteur

Ce qui est protégeable:

- Le code source;
- le code objet;
- l'organigramme;
- le matériel de conception préparatoire (c'est l'ensemble des travaux de conception aboutissant au développement d'un programme, à condition qu'ils soient de nature à permettre la réalisation d'un programme d'ordinateur à un stade ultérieur);
- la police de caractères.

Ne sont pas protégeables :

- les algorithmes, car ils sont assimilés à de simples idées;
- les interfaces;
- les fonctionnalités en tant que telles (c'est leur expression écrite ou graphique qui est protégeable).

I – 6 Droit d'auteurs, quels types de droits?

Distinction entre 2 types de droits:

Les droits moraux:

- appartiennent à l'auteur;
- ils sont perpétuels et inaliénables et imprescriptible (transmission aux héritiers après la mort de l'auteur).
- En matière de logiciel, ces droits se réduisent essentiellement au droit à la paternité (droit au nom) + Droit de s'opposer à la modification si préjudiciable à son honneur ou à sa réputation (exemple);

Les droits patrimoniaux:

- Correspondent aux droits d'exploitation (droit de représentation + droit de reproduction);
- La durée de cette protection est la durée de la vie de son auteur et 70 ans après sa mort (ou du dernier auteur survivants en cas d'œuvre de collaboration) ou en cas d'œuvre collective 70 ans après l'année de sa publication.
- Lorsque l'auteur est salarié et a réalisé le logiciel dans le cadre de ses fonctions ou d'après les instructions de son employeur, **les droits patrimoniaux sont dévolus à son employeur (art. L. 113-9 CPI)**

Droit de modification = Droit moral + Droit patrimonial

I – 7 droit d'auteurs, comment se formalise l'exploitation des droits patrimoniaux ?

→ Cette exploitation peut se faire par le propriétaire du logiciel lui-même

→ Cette exploitation peut également être faite par un tiers, auquel le propriétaire du logiciel a cédé un ou plusieurs de ses droits d'exploitation (contrat écrit);

→ Une licence d'exploitation est un contrat qui définit les conditions dans lesquelles on autorise le licencié à exploiter le logiciel

→ Ceci suppose d'avoir préalablement définit une stratégie de distribution et d'exploitation, afin d'adopter ou rédiger une licence qui soit cohérente avec cette stratégie

→ Dans la pratique, cela se traduit par l'existence d'une grande diversité de licences dans le monde du logiciel: licences FLOSS, licences propriétaires

I – 8 conséquences pratiques lorsque l'employeur (Inria) est titulaire des droits d'exploitation

→ Inria, **en qualité d'employeur et titulaire des droits patrimoniaux**, est *seul habilité à choisir le régime de diffusion/distribution* du logiciel développé par ses salariés (y compris pour le choix d'une distribution sous une licence FLOSS).

→ Dans la pratique, logique de discussion et de consensus entre les différents acteurs en interne: chercheur, ingénieur, CPPI, juriste PI, responsable du patrimoine logiciels…

I – 9 comment bénéficier de la protection par le droit d'auteur?

→ Contrairement aux brevets, un logiciel est protégé par le droit d'auteur dès sa création, sans formalité particulière

→ Cette protection semble simple à obtenir (« je n'ai rien besoin de faire! »)…

→ … mais dans la pratique, cela se révèle plus compliqué: il faut être en mesure de prouver qu'on est effectivement titulaire des droits sur un logiciel

→ Cette preuve se fait par tout moyen

Il est nécessaire de pouvoir prouver qu'on est titulaire des droits sur un logiciel pour:

→ justifier du fait que l'on soit légitime/dans son droit à en assurer l'édition et l'exploitation

→ pour se donner les moyens de défendre ses droits sur le logiciel en cas de contrefaçon

I – 10 quel moyen de preuve de sa titularité des droits?

→ Certains moyens de preuve plus fiables que d'autres sur le plan juridique;

→ Les preuves techniques (par exemple: copie du gestionnaire de version à une certaine date) peuvent constituer des commencements de preuve mais ne sont pas considérés comme suffisants à eux seuls pour faire valoir ses droits;

→ Il existe également des dépôts probatoires auprès des tiers de confiance (notaires, organisme réalisant des dépôts certifiés…), qui constituent à ce jour l'un des meilleurs moyen de faire valoir ses droits;

→ Un tel dépôt permet de prendre date et facilite l'identification et la défense du logiciel (en cas de contrefaçon: défense ou attaque)

I – 11 quelle politique d'Inria en ce qui concerne les dépôts probatoires?

→ Inria a fait le choix de procéder à des dépôts certifiés auprès de l'APP (agence de protection des programme)

→ Permet d'apporter la preuve de l'existence d'une œuvre précise à une date certaine;

→ Il est fortement recommandé de procéder à un dépôt APP à chaque étape importante de la vie d'un logiciel, notamment:

- à chaque nouvelle version stabilisée
- lorsque le logiciel doit faire l'objet de travaux de recherche dans le cadre d'un partenariat avec un tiers
- en cas de cession ou de concession de licence d'exploitation
- après le départ d'un membre clef de l'équipe de développement

→ Pour procéder à un dépôt APP à Inria:

- les auteurs remplissent avec le CPPI un formulaire de demande de dépôt APP (de façon à identifier les informations nécessaires pour l'APP)
- les auteurs remettent au CPPI le code source en 3 exemplaires (sur CD-ROM)
- le CPPI fait le nécessaire pour que le dépôt soit fait à l'APP
- l'APP retourne à Inria un certificat et une copie scellée du dépôt (Logibox), qui est archivée à la DTI

II – Intentions et choix de la licence

II – 1 Rappel: acteurs du choix de licence

→ l'employeur en qualité de titulaire choisit la licence

→ A l'Inria, ce choix est le fruit d'une discussion interne entre les acteurs concernés (chercheur, ingénieur, CPPI, juriste, responsable patrimoine logiciels…)

Lorsque ce choix se fait, il est important de réfléchir sur le long terme!

II – 2 Comment choisir la licence?

→ Prendre le temps de discuter et définir de façon aussi précise que possible, avec le CPPI, des intentions de l'équipe, tant au niveau du développement que du type d'exploitation/distribution envisagée (stratégie d'exploitation et de développement)

→ Eviter au contraire d'arriver avec un choix de licence en pensant pouvoir « faire rentrer » les intentions dans les « cases » de la licence

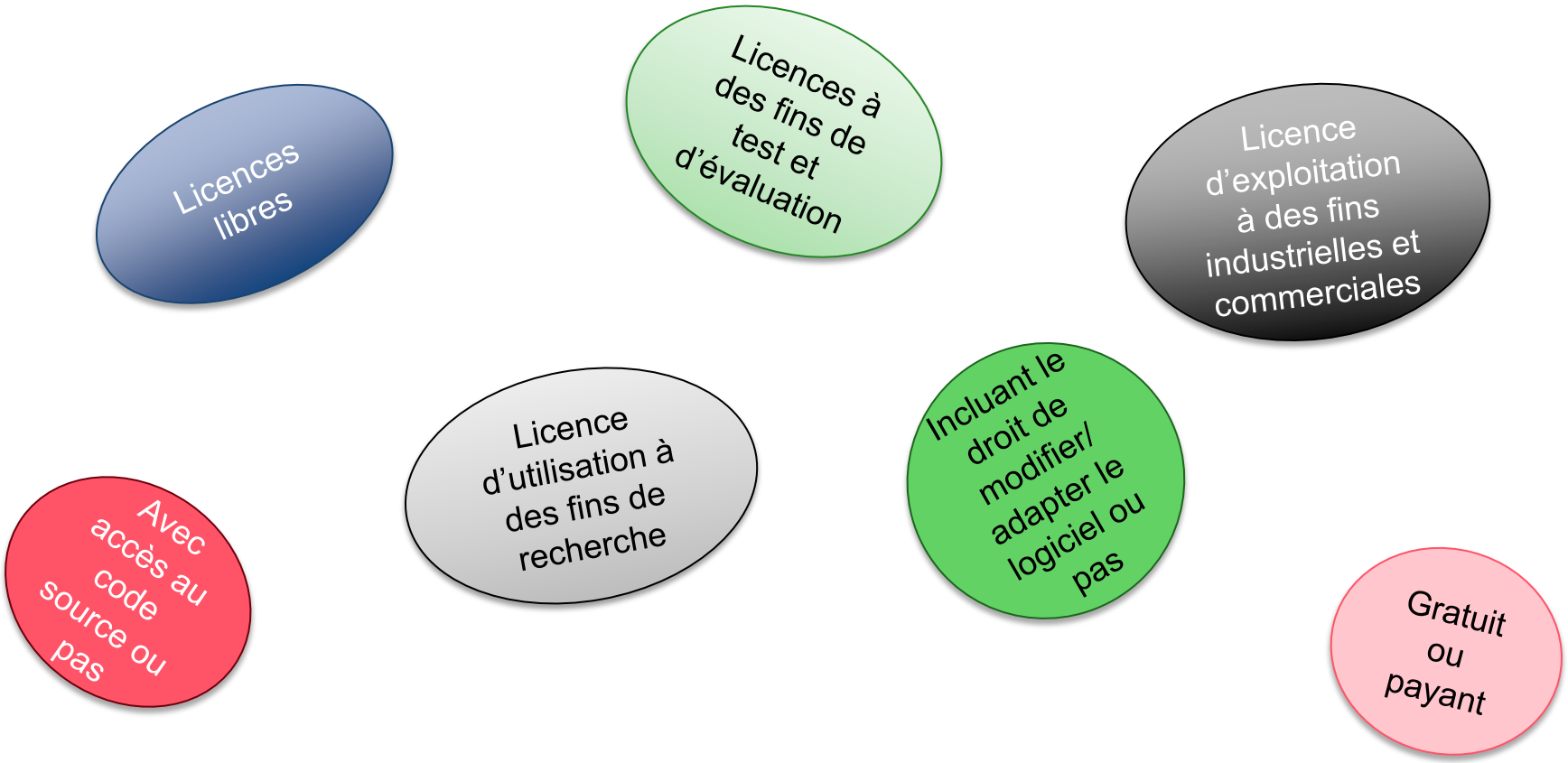
→ Avoir conscience qu'il existe une grande diversité de licences, et qu'il ne faut pas s'arrêter à une seule des caractéristiques d'une licence pour définir un choix

→ Il faut envisager la licence dans sa globalité pour s'assurer de son adéquation avec les intentions de développement et d'exploitation/distribution

(exemple: je veux donner accès au code source à mes confrères d'autres universités à des fins de recherche => une licence libre donne accès au code source => mais une licence libre est-elle néanmoins adaptée, dans sa globalité?)

II – 3 Quelles sont les différentes licences existantes?

Au-delà du clivage réducteurs entre licences libres et licences propriétaires, il existe une grande diversité de licences



Et beaucoup d'autres encore…

II – 4 Quelle différence entre une licence libre et une licence non-libre donnant accès au source (à code ouvert)?

Pour être libre, une licence doit répondre à 4 critères (définis par la FSF):

- Liberté d'étudier librement le logiciel
- Liberté d'utiliser et reproduire le logiciel
- Liberté de modifier le logiciel
- Liberté de redistribuer le logiciel modifié ou non, a minima sous la licence d'origine

L'obligation de donner accès au code source est implicite (nécessaire pour étudier et modifier le logiciel)

Si une licence ne répond pas à un seul de ces critères, elle n'est pas libre.

L'OSI a également défini une dizaine de critères, qui sont inclus de façon implicite dans les 4 libertés de la FSF.

II – 5 Exemples de « fausses » licences libres à code ouvert

→ Accès au code source, droit d'utiliser, modifier, **mais pas de droit de redistribuer**

→ Accès au code source, droit d'utiliser, modifier, et droit de redistribuer **à des fins non-commerciales uniquement**

→ Accès au source, droit de modifier, de redistribuer, **mais pas le droit d'utiliser à certaines fins (par exemple, à des fins militaires)**

II – 6 Licences libres: typologie du lieu du métier de l'éditeur réutilisant du code préexistant

- Proposition d'une typologie fondée:
  - sur les conditions de redistribution d'un composant…;
  - … quand à la liberté laissée pour le choix de licence de « licensing out »;

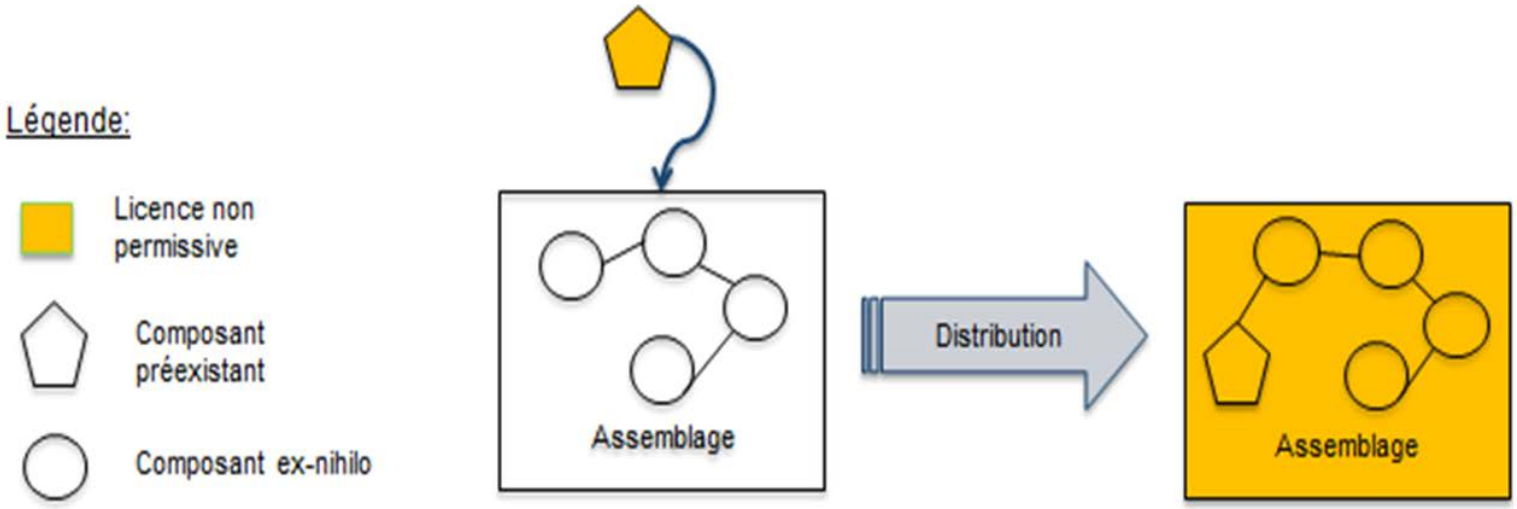
- Permet de distinguer trois types de licences libres:

  - Non permissives;
  - Permissives en composition;
  - Permissives en composition et dérivation.

II – 7 Les licences libres non permissives (1)

- On parle également de licences « copyleft », « restrictives » ou « contaminantes »;
- Impose l'utilisation de la même licence, ou d'une licence dite compatible, pour redistribuer le logiciel, précédemment distribué sous ce type de licence, qu'il soit modifié ou non.
- Exemples de licence: GNU GPL; CeCILL; OSL v3.
- Objectif: Diffusion du code et de toutes ses dérivées sous licence libre (philosophie).

II – 7 Les licences libres non permissives (2)

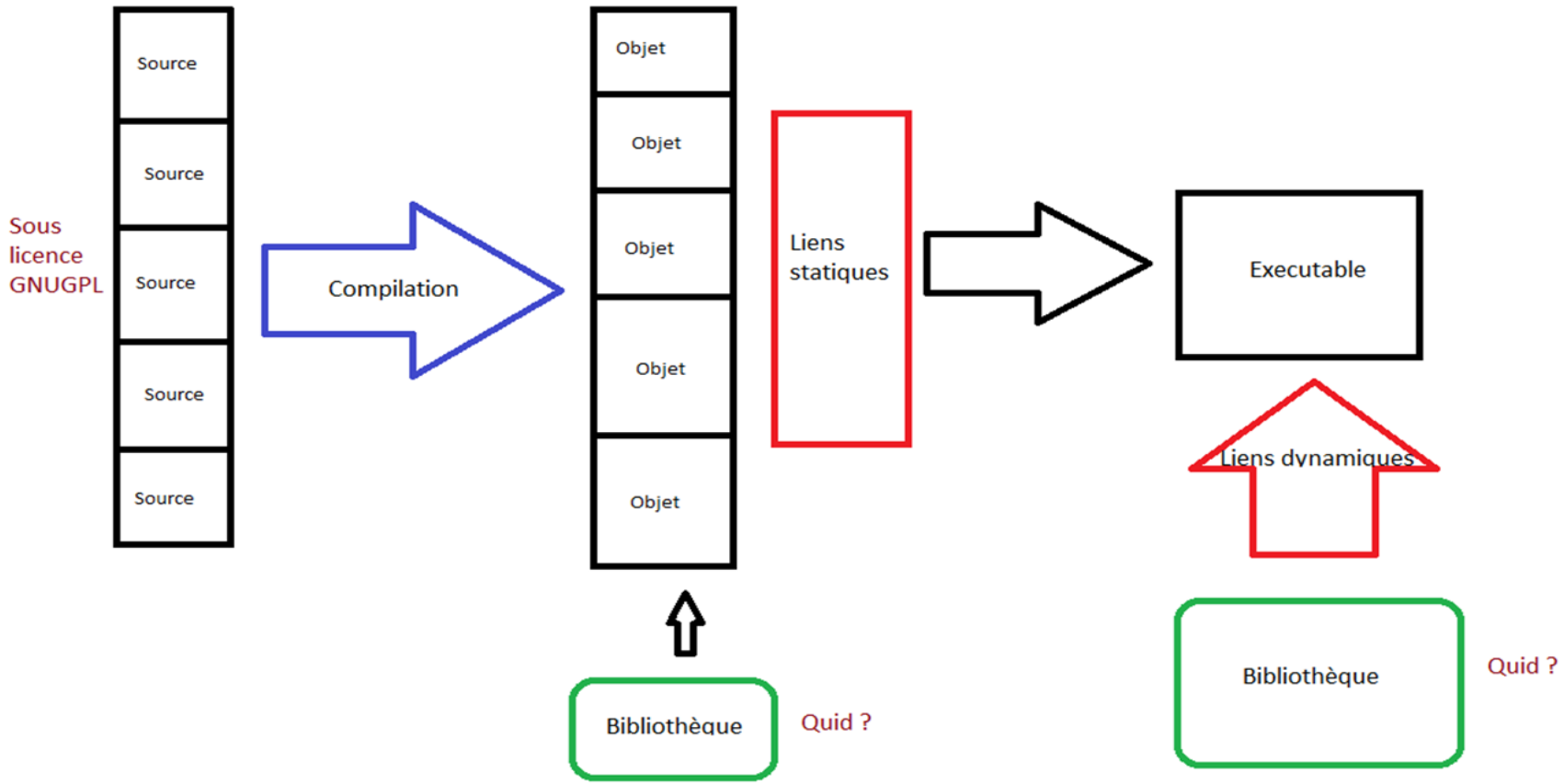


II - 8 Liens dynamiques et liens statiques

- Lien statique: Lien permettant de lier les fonctionnalités de plusieurs programmes dans le même exécutable (phase de compilation). C'est un lien qui est différenciable au niveau du code objet, ne l'est plus au niveau de l'exécutable ;

- Lien dynamique: Lien permettant à un programme 1 de faire appel aux fonction d'un programme 2 (ex: librairie), après la phase de compilation. C'est un lien ponctuel.

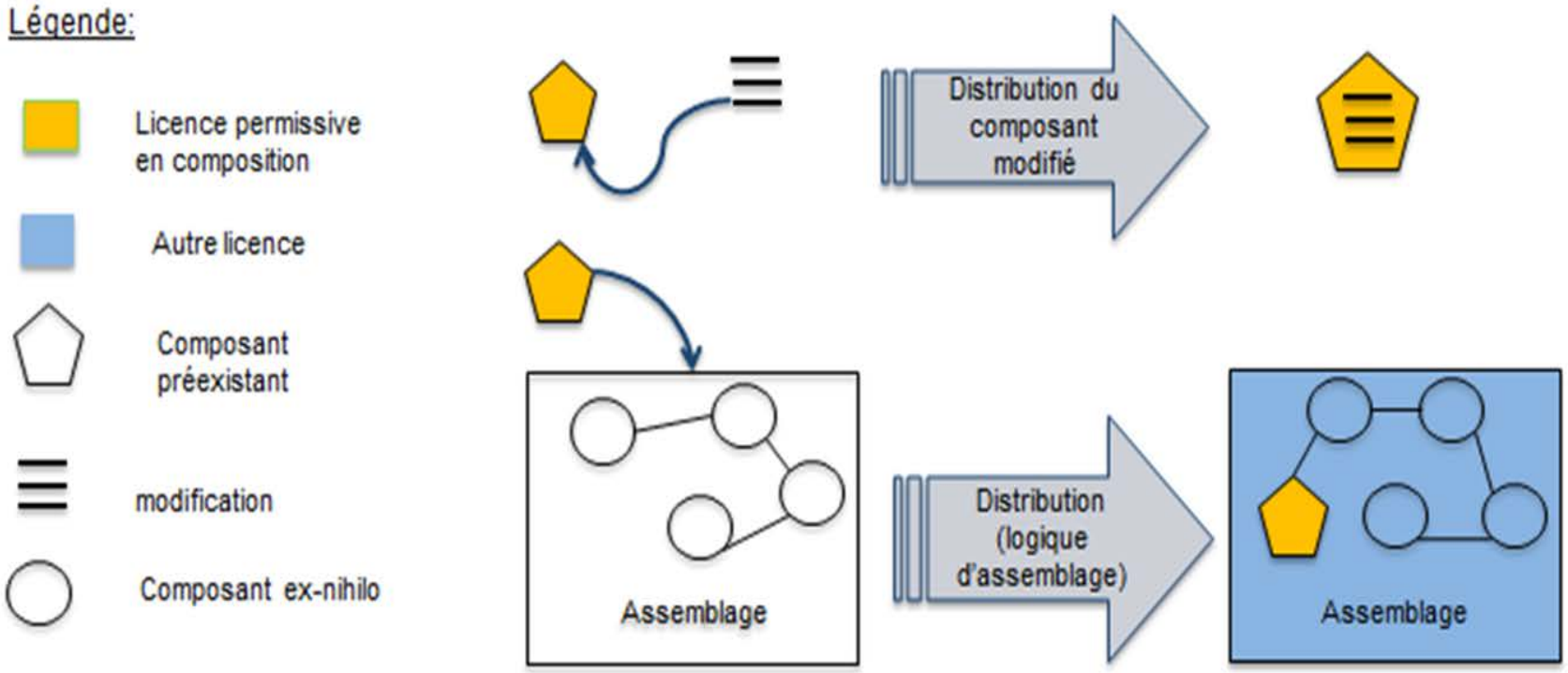
II – 9 Liens dynamiques et liens statiques: Exercice



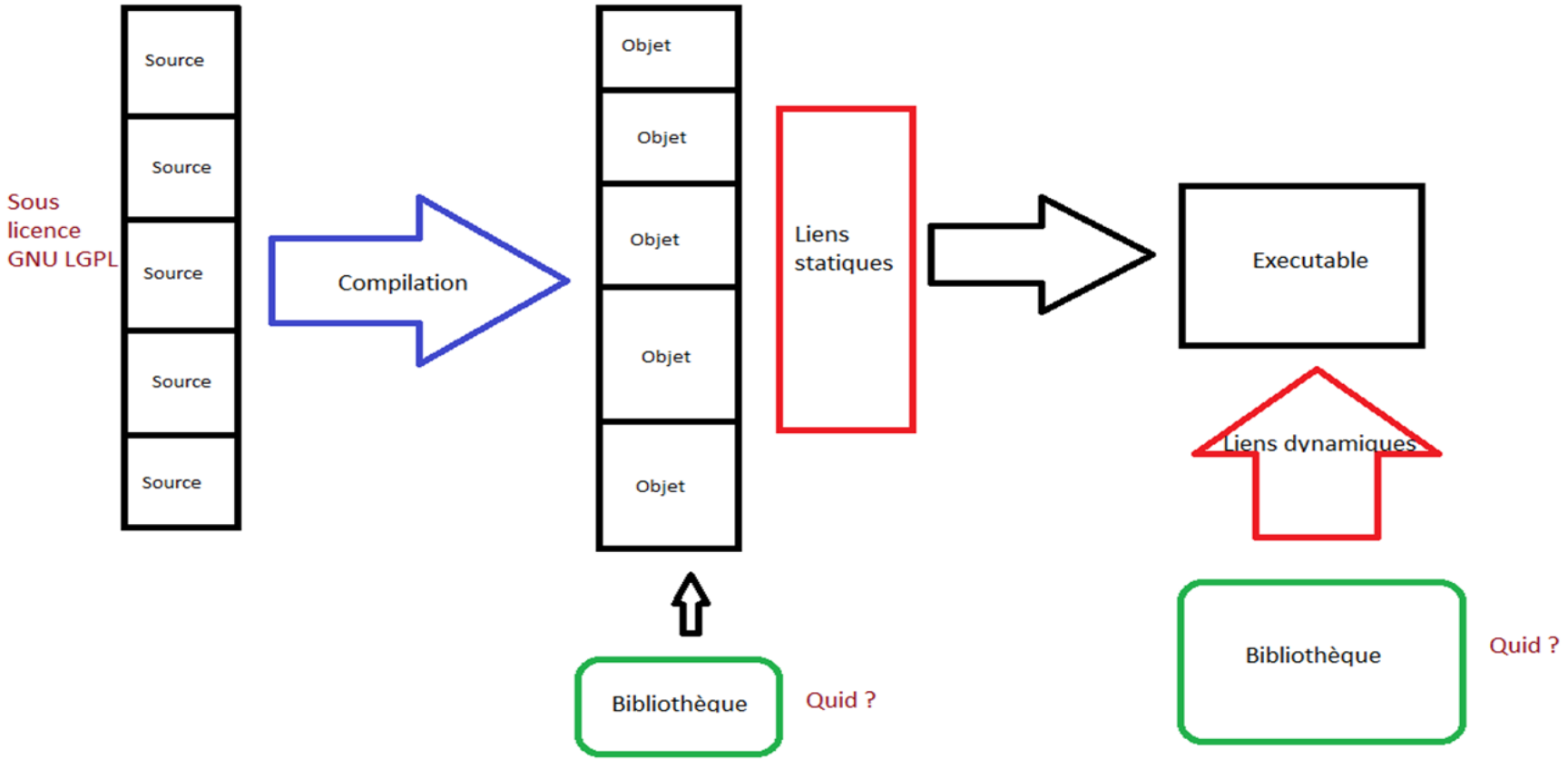
II – 10 Les licences libres permissives en composition (1)

- Impose l'utilisation de la même licence, ou d'une licence dite compatible, pour redistribuer le logiciel, précédemment distribué sous ce type de licence, qu'il soit modifié ou non.
- Par contre, dans l'hypothèse où un autre composant serait linké à ce logiciel (les deux doivent être facilement dissociable), la licence n'impose aucune restriction quant à la redistribution de ce composant.
- Exemples: GNU LGPL; CeCILL-C; MPL; EPL v1.0; CPL v1.0.
- Objectif: Diffusion du code et de ses dérivées sous licence libre. Permettre une utilisation plus large du code (même avec des composants propriétaires);

II – 10 Les licences libres permissives en composition (2)



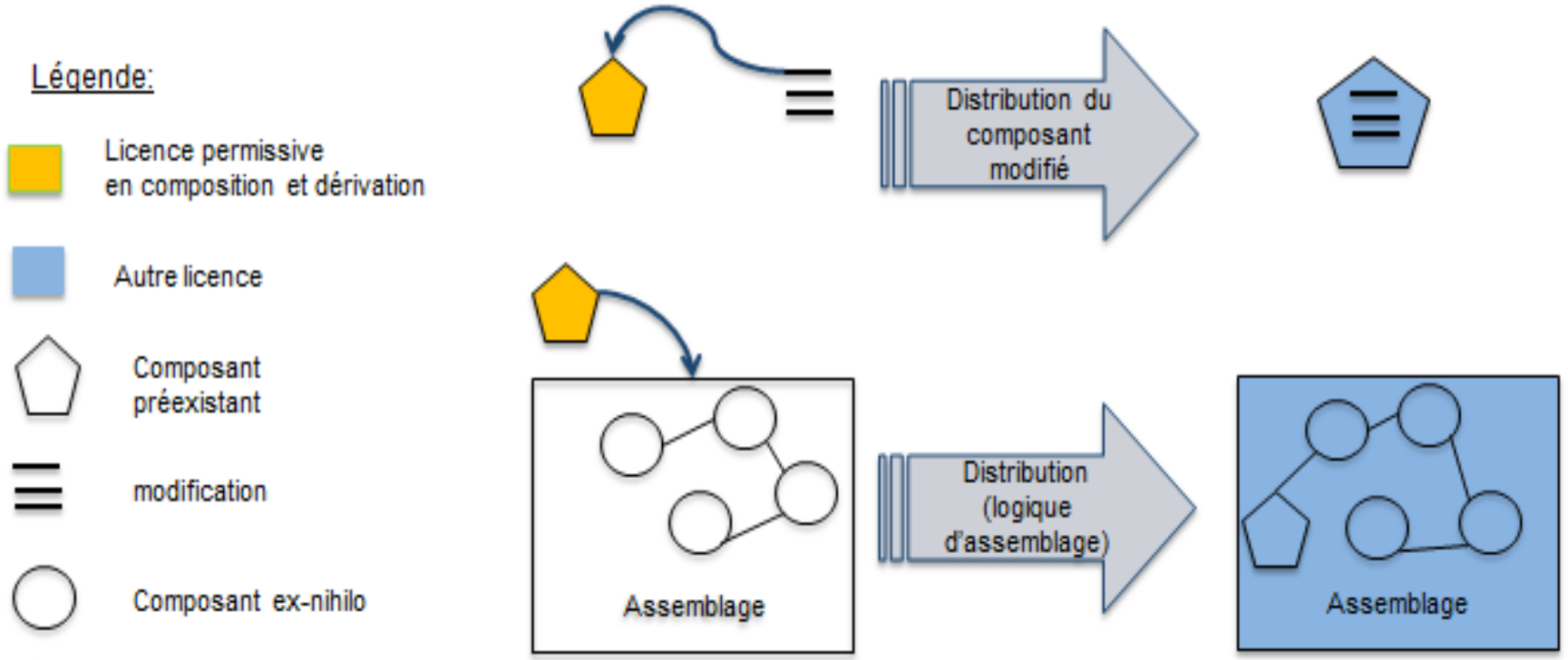
II - 10 Les licences libres permissives en composition: Exercice



II – 11 Les licences libres permissives en composition et en dérivation (1)

- N’impose aucune restriction quant à la redistribution du logiciel, même modifié (un logiciel peut ainsi passer sous une licence propriétaire);
- Exemple: BSD; CeCILL-B; Apache v2.0.
- Objectif: Permettre une diffusion large du logiciel.

II – 11 Les licences libres permissives en composition et en dérivation (2)



II – 11 Quelle philosophie du propriétaire du composant/logiciel derrière cette typologie ?

Type de licence	Philosophie du propriétaire	
Licence non permissive	*« Si vous redistribuez mon code dans votre logiciel, c'est avec les mêmes règles du jeu pour les deux et en partageant avec la communauté »*	*« Si vous redistribuez mon code dans votre logiciel ou si vous l'exploitez en mode SaaS, c'est avec les mêmes règles du jeu pour les deux et en partageant avec la communauté »*
Licence permissive en composition	*«Si vous redistribuez mon code (modifié ou pas) dans votre logiciel, il doit rester accessible et profiter à la communauté (« licensing back »); en revanche, vous choisissez vos règles du jeu pour le reste de votre logiciel »*	
Licence permissive en composition et en dérivation	*« Mon code, c'est cadeau, faites en ce que vous voulez! »*	

II – 12 Et une fois la licence choisie?

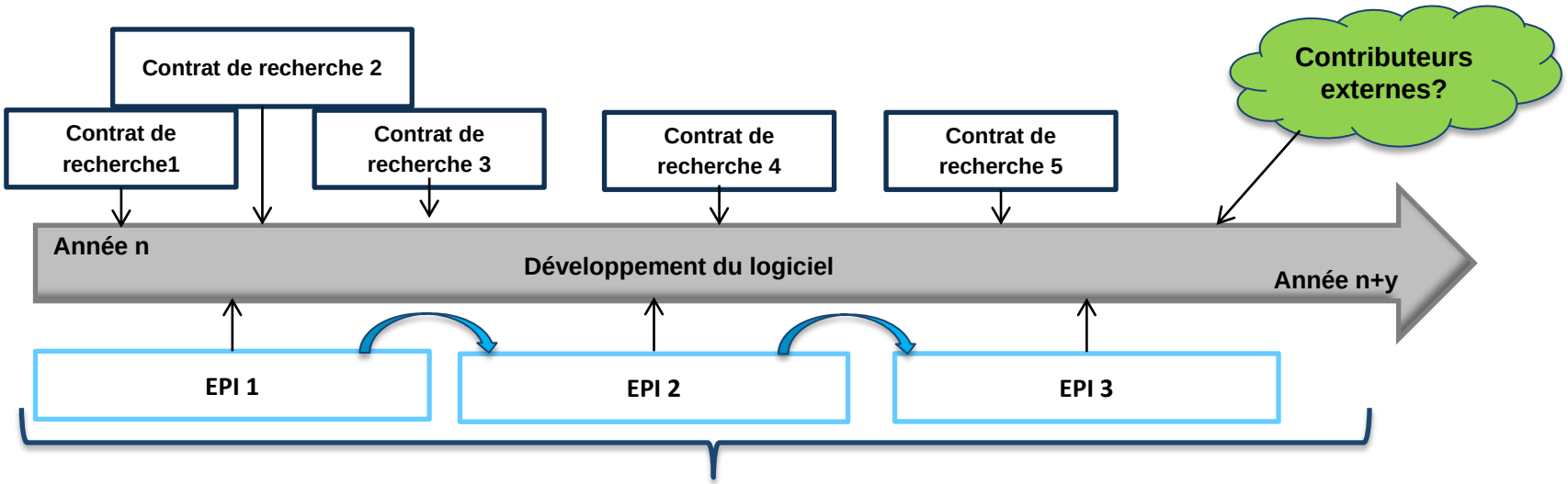
Inria est-il en mesure de distribuer/éditer le logiciel sous la licence envisagée?

En effet, dans la pratique, les choix dans l'organisation du développement d'un logiciel se traduisent souvent de la façon suivante:

- un développement collaboratif

- une réutilisation de code préexistante (notamment open source)

III – Incidence du développement en mode collaboratif



Equipes 100% Inria? Mixtes? Hébergeant des stagiaires ou autres personnes non salariés? Chercheurs changeant de statut?

III – 1 Le développement collaboratif : un élément générateur de copropriété

Type de mode collaboratif		Non-générateur de copropriété	Générateur de copropriété
Inexistant (un seul auteur)		x	
EPI 100% Inria (pluralité d'auteurs mais tous Inria)		x	
Equipe-projet commune			x
Au-delà d'une EPI	Contrat de partenariat de recherche		x
	Équipe hébergeant des ayants droits à titre personnel		x
	Existence de contributeurs externes		x

III – 2 Les contraintes/difficultés pratiques liées à l'identification des copropriétaires

→ cas des logiciels avec un grand nombre de « *committers* » à identifier, parfois difficiles à identifier (login avec nom « obscur », personne dont l'équipe ne se souvient plus)…

→ … un statut à déterminer (plus ou moins difficilement) pour chacun, pour identifier les ayants droits…

→ … à retrouver parfois plusieurs années après…

Sachant que ces copropriétaires peuvent s'avérer plus nombreux que prévu et d'une plus grande diversité (établissements publics, industriels, personnes physiques…)

→ Parfois, la copropriété n'est pas liées aux personnes des auteurs et leur statut, mais peut être prévue par voie contractuelle (clause d'un contrat de recherche prévoyant que le logiciel sera en copropriété, même si un partenaire n'est pas auteur)

III – 3 Les contraintes/difficultés pratiques liées à l'exploitation du logiciel en copropriété

→ Nécessite l'accord unanime de tous les copropriétaires sur le choix du mode d'exploitation du logiciel (y compris pour un changement de licence libre, par exemple), même si Inria est seul éditeur du logiciel

Au plus on est nombreux, au plus c'est compliqué de se mettre tous d'accord (même pour le choix d'une licence libre)!

→ Copropriétaires nombreux = coûts de transaction et de coordination importants

→ Copropriété parfois atypique à gérer (cas des logiciels avec des ayants droits à titre personnel, comme les stagiaires)

→ **Ces problèmes ont déjà dans le passé bloqué la valorisation de logiciels**

III – 4 (a) Pour résumer:

Certes, le développement collaboratif:

- Fait partie de quotidien (EPI communes, contrats…)
- Peut constituer une valeur ajoutée
- Peut être techniquement simple à mettre en œuvre (« il suffit » de donner des droits d'accès à un tiers contributeur sur la forge)

Mais:

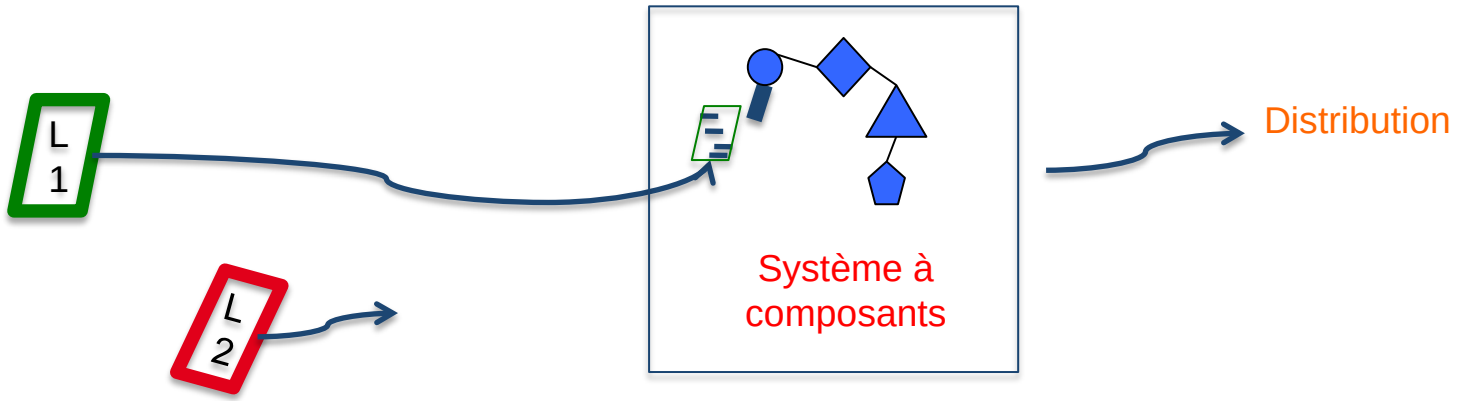
- il génère des contraintes plus ou moins importantes sur le logiciel et notamment, sa liberté d'exploitation
- Il est donc important de **discuter** avec le CPPI de **la feuille de route du logiciel**, afin d'identifier des solutions possibles pour limiter/aménager ces contraintes

IV – Incidence de la réutilisation de code

Réutilisation
de code
(composants préexistants)

Logiciel : assemblage de
composants
(avec des composants “ex-nihilo”)

exploitation



IV – 2 Nature du code préexistant

Il peut provenir d'autres projets d'Inria ou de projets tiers;

Ce code est souvent du code open source mais pas toujours:

- code propriétaire provenant d'un partenaire;

(Dans quel cas, attention aux conditions d'utilisation qui sont généralement restreintes!)

- code qui « a l'air » open source;

(Code soumis à une licence qui donne accès au code source mais sans être libre et qui par conséquent, n'est pas compatible avec une licence libre)

→ Ca peut devenir compliqué

Conclusion

→ Du fait de l'incidence des choix de développement en termes de propriété et de liberté d'exploitation, il est important que les équipes en aient conscience;

→ Ce n'est pas le rôle des équipes de traiter ses questions seules, par elles-mêmes…

→ Mais étant au cœur de la problématique, il est important de développer une bande-passante forte avec le SED/CPPI/juriste PI du centre, pour traiter ensemble et efficacement de ces questions