

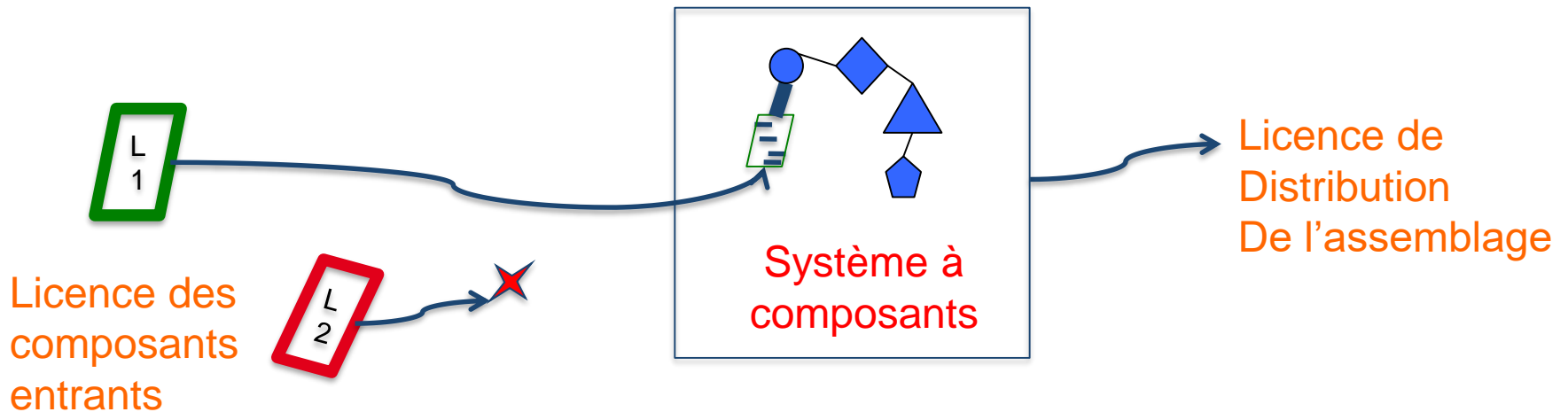
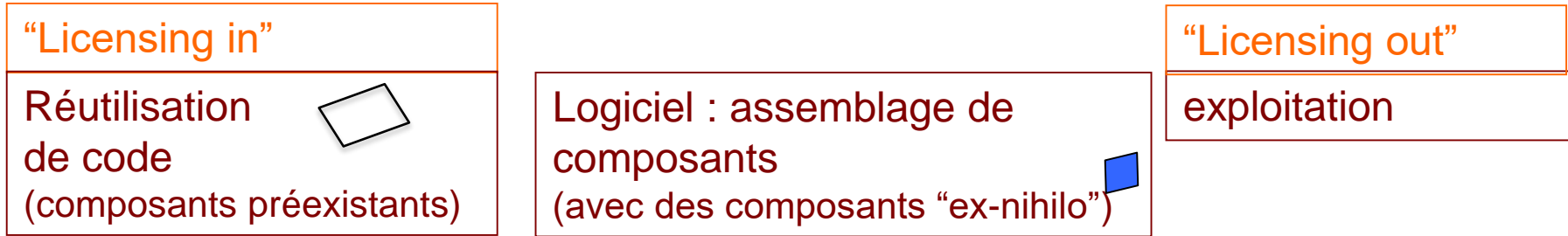
**LES France**

-

**Webex du GT Logiciel du 1<sup>er</sup> avril 2020**

**Licences Open Source : aspects juridiques et  
outils de traçabilité**

# La réutilisation de code du point de vue du juriste...

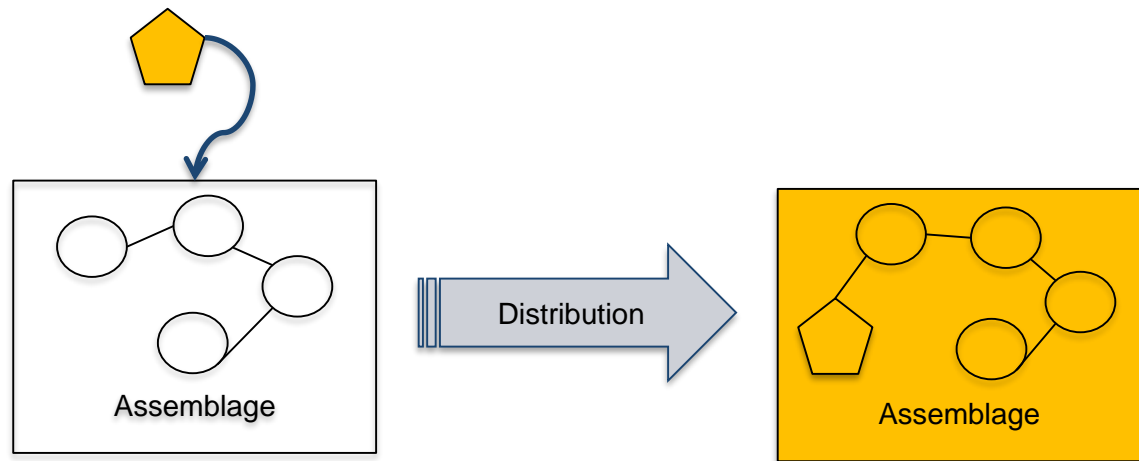
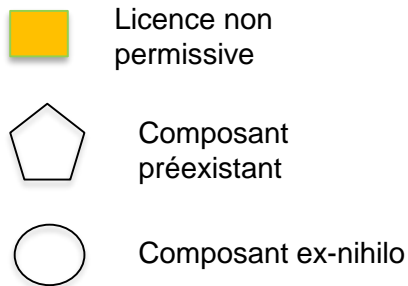


*Les conditions des licences « entrantes » autorisent-elle de distribuer l'assemblage sous la licence « sortante » ?*

# Les licences non permissives (copyleft fort)

- Le composant, modifié ou pas, ne peut être redistribué que sous la licence non permissive d'origine uniquement
- En cas d'intégration dans un assemblage, ce dernier ne peut être redistribué que sous cette même licence non permissive
- Exemples de licences non-permissives: GNU GPL, GNU AGPL, CeCILL, OSL






## Légende:

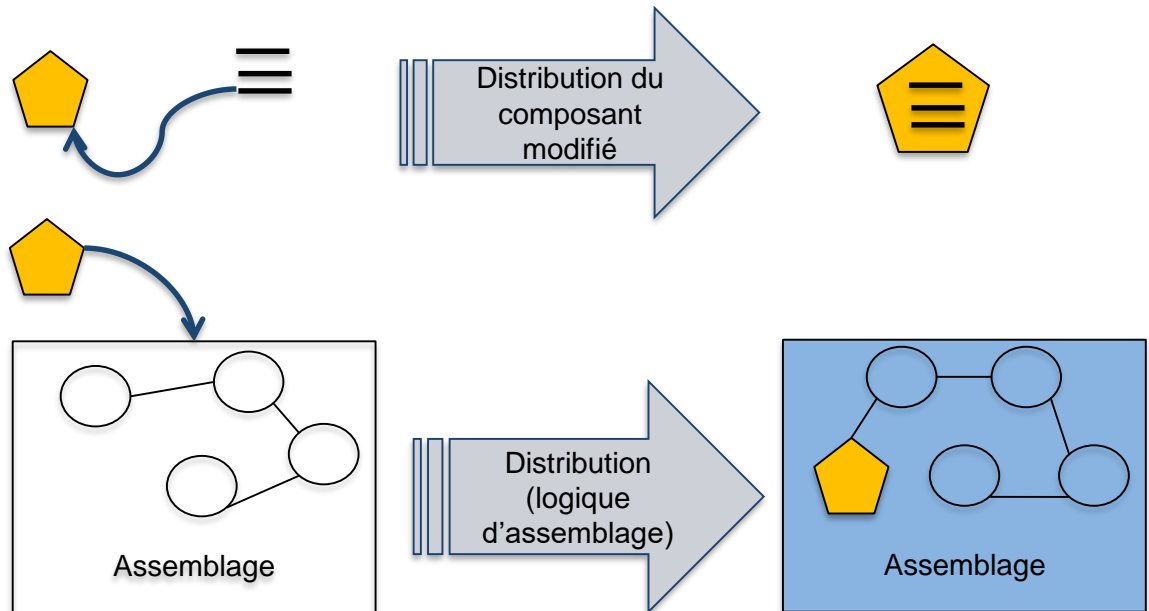


# Les licences permissives en composition ou en assemblage (copyleft atténué/héritaires)

- Le composant, modifié ou pas, ne peut être redistribué que sous la licence non permissive d'origine uniquement
- En revanche, si je « link »/intègre le composant dans un assemblage, je peux redistribuer ce dernier sous une autre licence
- Exemple de licences permissives en composition: GNU LGPL, CeCILL-C, Mozilla public license, Eclipse public license

## Légende:


-  Licence permissive en composition
-  Autre licence
-  Composant préexistant
-  modification
-  Composant ex-nihilo




# Les licences permissives en composition et dérivation


- Le composant modifié peut être redistribué sous une autre licence
- Si je « link »/intègre le composant dans un assemblage, je peux redistribuer ce dernier sous une autre licence également
- Exemple de licences permissives en composition et dérivation : BSD, Apache, CeCILL-B, MIT


## Légende:

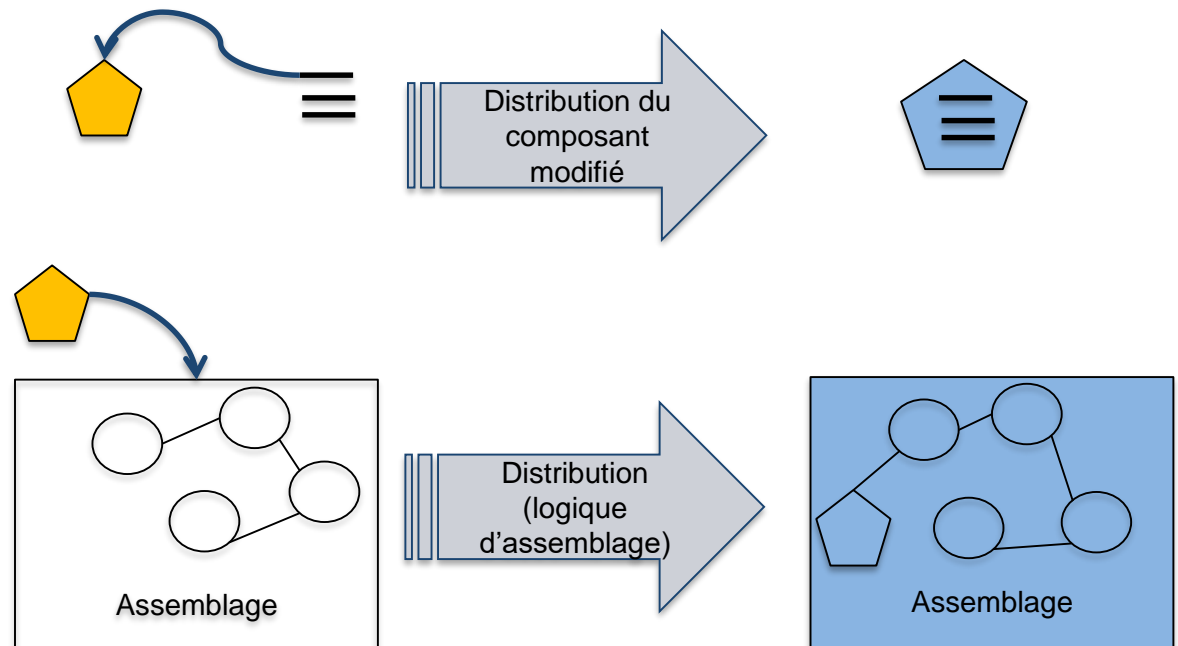
 Licence permissive en composition et dérivation

 Autre licence

 Composant préexistant

 modification

 Composant ex-nihilo



# Partie 1

**Allons plus loin: le diable se cache dans les détails...**

## Pourquoi dans le détail?

- Les catégories de licences indiquées permettent de disposer d'une grille de lecture simple
- Dans la pratique a mise en œuvre nécessite d'avoir une lecture plus poussée
- Cette seconde lecture doit s'appuyer sur des considérations de différentes natures:
  - Juridiques/contractuelles...
  - Mais aussi techniques...
  - Ou encore business
- La démarche n'est pas celle du juriste seul:
  - ✓ Une valeur ajoutée forte du couplage des compétences juridiques/développement/licensing-business

# L'élément déclencheur est-il là ?

- La présence de licences incompatibles dans un logiciel n'est pas obligatoirement gênante...
- Cela le devient dès lors qu'il devient nécessaire de respecter les obligations contradictoires des unes et des autres...
- Attention donc à l'élément « déclencheur » de ces obligations :
  - ✓ Très souvent, c'est la distribution
  - ✓ Pour un nombre très réduit de licences, la mise à disposition à distance via un serveur suffit (exp: GNU AGPL et OSL)
- Attention à l'élément déclencheur qu'on « élude »:
  - ✓ à la GNU GPL et à une exploitation en mode SaaS: on peut avoir besoin de distribuer préalablement une solution SaaS malgré tout!
  - ✓ Cas de la filialisation d'une activité
- A qui doit on faire bénéficier des droits prévus par la licence O-S, si l'élément déclencheur est au rendez-vous:
  - ✓ Mon licencié v. le reste du monde



# Les spécificités introduites par les clauses de certaines licences qui changent « la donne »

- La Cecill B est une licence qui se veut très permissive (de type BSD/MIT)
  - ✓ Elle a donc a priori vocation à être compatible avec n'importe quelle autre licence
- Cette liberté reste néanmoins soumise au respect d'une clause de citation spécifique qui s'avère très contraignante
  - ✓ Cette clause la rend incompatible avec la GNU GPL notamment
  - ✓ Elle la rend sans difficilement acceptable dans une distribution propriétaire

# Quand les contraintes techniques ont (peuvent avoir) une incidence sur la portée d'une licence

- Un exemple avec le cas de bibliothèques écrites en C++ diffusées sous licence GNU LGPL v2.1:
  - ✓ Une reprise « contrainte » de lignes de code des bibliothèques en C++ pour identifier les fonctions utilisées à la compilation
  - ✓ Un débat quant à savoir si cela doit être considéré comme une « dérivation » au sens de la GNU LGPL
    - Si on adopte une lecture d'application stricte et littérale de la GNU LGPL v2.1: oui!
    - Si on adopte une lecture en tenant compte de l'esprit de la licence: ça se discute...
- La FSF a finalement intégré dans la GNU LGPL v3 un article spécifiant ce cas en particulier
- Illustre un enjeu:
  - ✓ Comprendre a minima l'objet devant être analysé
  - ✓ Être capable de définir le périmètre de l'analyse

## Quand la compatibilité juridique ne suffit pas...

- La mise en œuvre des obligations d'une licence OS peut être possible sur le plan juridique... mais pas souhaitable sur un plan technique et business
- Un exemple avec les contraintes de mise en œuvre de la GNU LGPL:
  - ✓ Possible de lier un logiciel sous GNU LGPL à un logiciel soumis à une autre licence mais...
  - ✓ Si le licencié modifie le code sous GNU LGPL, il doit pouvoir le recompiler avec le logiciel soumis à une autre licence
- Dans les faits, cette conditions peut être mises en œuvre, mais parfois avec des conditions techniques/business qui ne sont pas souhaitables
  - ✓ Exemple: cas des logiciels en langage compilés dans le domaine de l'embarqué
- Pour y remédier, certains trouvent une «astuce » sur le plan juridique:
  - ✓ Licences propriétaires et risque de remise en cause des garanties
- D'autres se passent de GNU LGPL (ou essaient) pour des raisons présentées comme étant « business » (Eclipse par exemple)

# Parties 2: gouvernance de l'OS dans les projets et outillage

# Les enjeux liés à la gestion des licences OS s'inscrivent dans des contextes qui « donnent le tournis »

- Variabilité dans les niveaux d'échelle en ce qui concerne la taille des assemblages logiciels:
  - ✓ De l'infiniment petit (exemple : logiciels implémentant des workflows dans le biomédical)...
  - ✓ A l'infiniment grand (exemple: Linux...)
- Variabilité de la temporalité dans laquelle s'inscrivent ces problématiques:
  - ✓ Le logiciel ne s'inscrit pas toujours dans des cycles courts...
  - ✓ Temporalité pour aller de la POC à une solution dans la recherche
  - ✓ Temporalité liée à la maintenance (l'exemple des logiciels embarqués dans l'aéronautique)

# Industrialisation mais aussi éclatement des modes de production des logiciels

- Complexité des workflows et de la chaîne d'acteurs
  - ✓ « L'usine logicielle »
  - ✓ Filiales, intégrateurs, outsourcing...
  - ✓ Equipes mixtes/UMR dans la recherche publique
  - ✓ Contrats collaboratifs
  - ✓ Développements communautaires
- Rend la traçabilité des composants exogènes difficile: avant même d'analyser les licences d'un logiciel, comment identifier en premier lieu les licences en présence?

# L'apparition d'outil pour accompagner la gestion de l'OS dans les projets

- La façon traditionnelle de faire (et dont on ne peut pas se passer): la déclaration par les développeurs
- Depuis une décennie, on a vu émerger différents outils pour aider à l'audit des codes (et souvent une offre de prestation de service associée)
- Des outils de différents types:
  - ✓ Des outils qui ne viennent pas du monde de l'audit mais... qui peuvent donner des informations intéressantes
    - Les gestionnaires de dépendances de type Maven
  - ✓ Des outils de traçabilité:
    - *Licence-checkers*
    - Comparaison de code source
- Un constat cohérent avec la partie 1: ce n'est pas l'affaire du juriste/du juridique seul

# Intérêt et limites de l'outillage

- Des outils qui ont un intérêt dans une logique de passage à l'échelle (identification des informations pertinentes pour une analyse juridique d'un logiciel)...
- ... mais un défi en termes d'organisation:
  - ✓ Une appropriation des outils est nécessaire
  - ✓ Doit s'intégrer dans un processus et un workflow adaptés à une entité donnée
- sachant que l'outil « miracle » n'existe pas:
  - ✓ L'information reste à qualifier (attention aux faux positifs)
  - ✓ Les *licence checkers* ne sont efficaces que pour autant que les licences sont déclarées...
  - ✓ Les outils de comparaison de code source ont une qualité qui dépend (entre autres) de l'exhaustivité de leur base de connaissances



# Software Heritage et la bibliothèque universelle des codes sources disponibles

- Un projet à but non lucratif qui adresse des enjeux du monde scientifique, de l'éducation, de la culture mais aussi... de l'industrie
  - ✓ Le partage d'une base exhaustive des codes sources disponibles répond aussi à des enjeux de l'industrie
  - ✓ Intégration/étude d'un enjeu complémentaire lié à la provenance: quelle certitude peut on avoir quant à la provenance d'un code source donné?
    - Exemple: est-on sûr que la librairie X embarquée dans le logiciel est bien la version Y, avec correction d'une faille de sécurité, et non une version parallèle distribuée par la communauté Z?
    - Une question qui peut notamment avoir une incidence sur des questions de sécurité et donc sur les clauses de garantie et de responsabilité contrats liés à des logiciels