Xmodeling Studio :

Un outil pour définir des DSL exécutables

Léa Brunshwig, Eric Cariou, Olivier le Goaër,

Université de Pau / LIUPPA, France

ECSEL Joint Undertaking

European Commission | Horizon 2020 European Union funding for Research & Innovation

CIEL 2018 : 7ème Conférence en IngénieriE du Logiciel

14 Juin, 2018

Etat des lieux

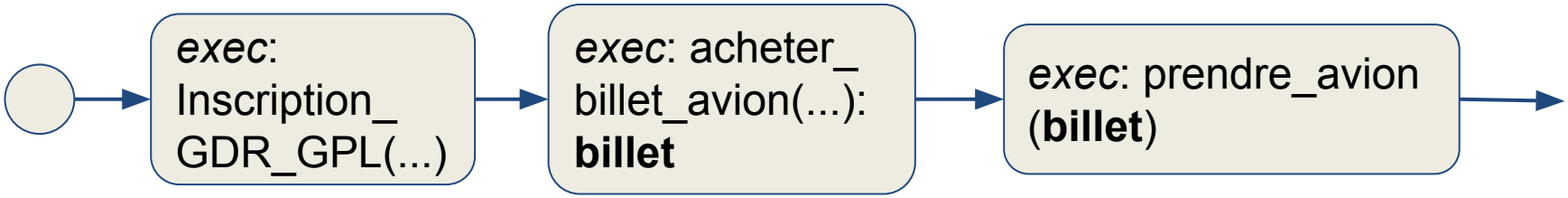
- Le rendez-vous manqué de l'IDM
  - Faible adoption par les entreprises « classiques » : combien de stages en IDM ? Combien d'emplois occupés ? ...
- Pourquoi ça coince ?
  - Les professionnels ne sont pas réfractaires aux notions d'abstraction, de DSL, ni aux technos innovantes, etc.
  - Ils sont juste en attente d'autres types de modèles, plus pragmatiques : qui s'intègrent à leurs SDK et outils habituels, leur bases de code existants, etc
- Approche plus pragmatique
  - « intégrer de l'IDM » dans du développement standard plutôt que d'espérer tout générer par les modèles

Application à la modélisation exécutable

- Modélisation exécutable (*X*modeling)
  - Application logicielle = comportement + métier
  - Comportement : défini par des modèles exécutables
    - Ex : machines à états, workflows, réseaux de Petri…
  - Métier : ensemble d'opérations (méthodes Java lambda ici)
- Intérêts
  - Définit le comportement à haut-niveau avec sémantique exéc.
  - Séparation claire comportement/métier
- Utilisation
  - Génération directe de code exécutable ou moteur d'exécution qui prend un modèle en entrée
  - Dans deux cas : gain car moteurs/générateurs indépendants du contenu métier des modèles

Challenges

- Créer des DSL exécutables avec moteur d'exécution ✓
  - On sait faire : Ecore, Java EMF, Kermeta, GEMOC ...
- Tisser les opérations métier sur le comportement ✗

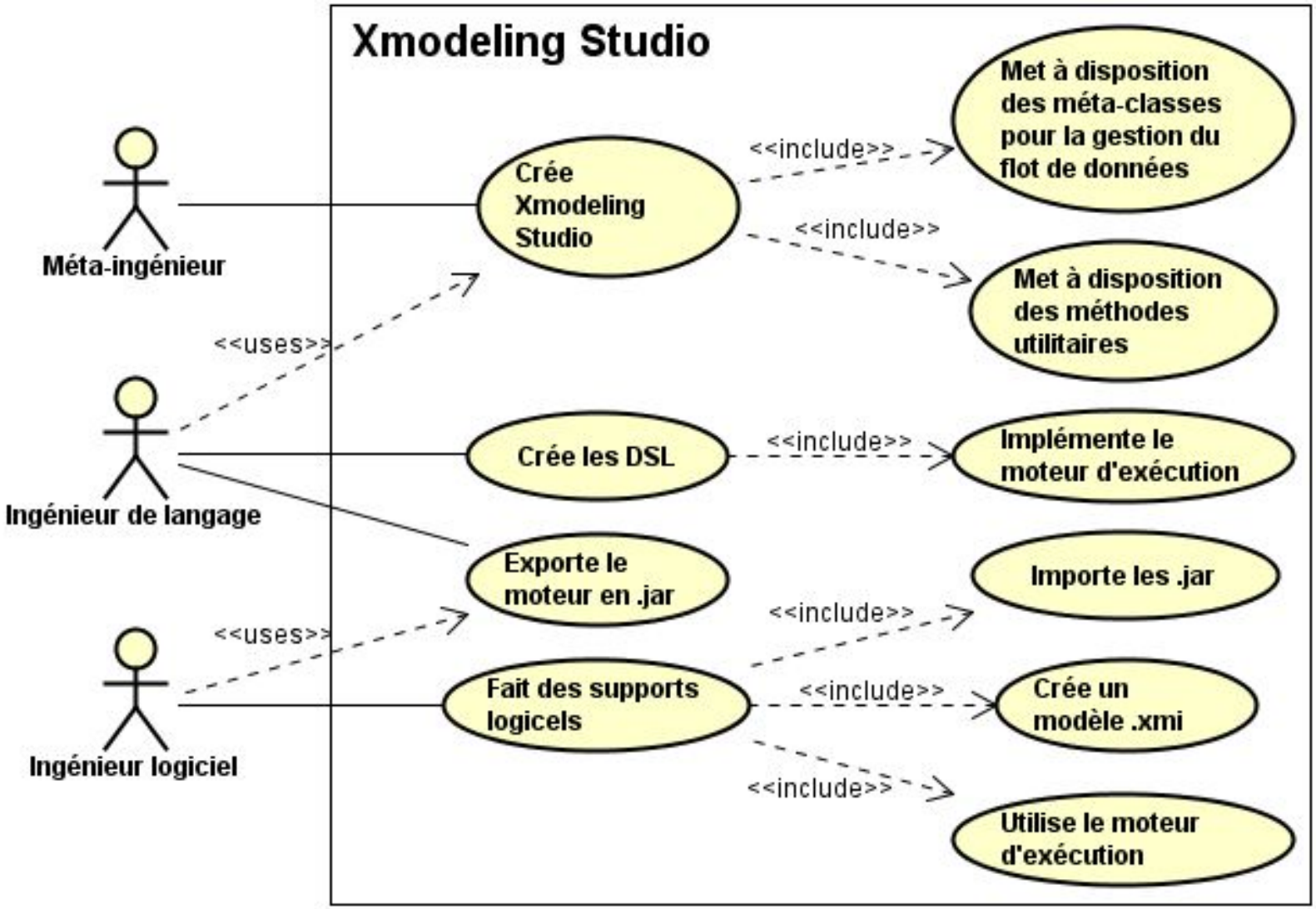


  - Moteur d'exécution est agnostique : indépendant du contenu métier du modèle, appelle les opérations activité par activité
  - Comment gérer des opérations de signature quelconque ?
  - Comment gérer le flot de données entre les opérations ?
- Intégration dans tout environnement Java-compliant ✗
  - Comment utiliser mon moteur et mes modèles pour faire tourner une application Android par exemple ?

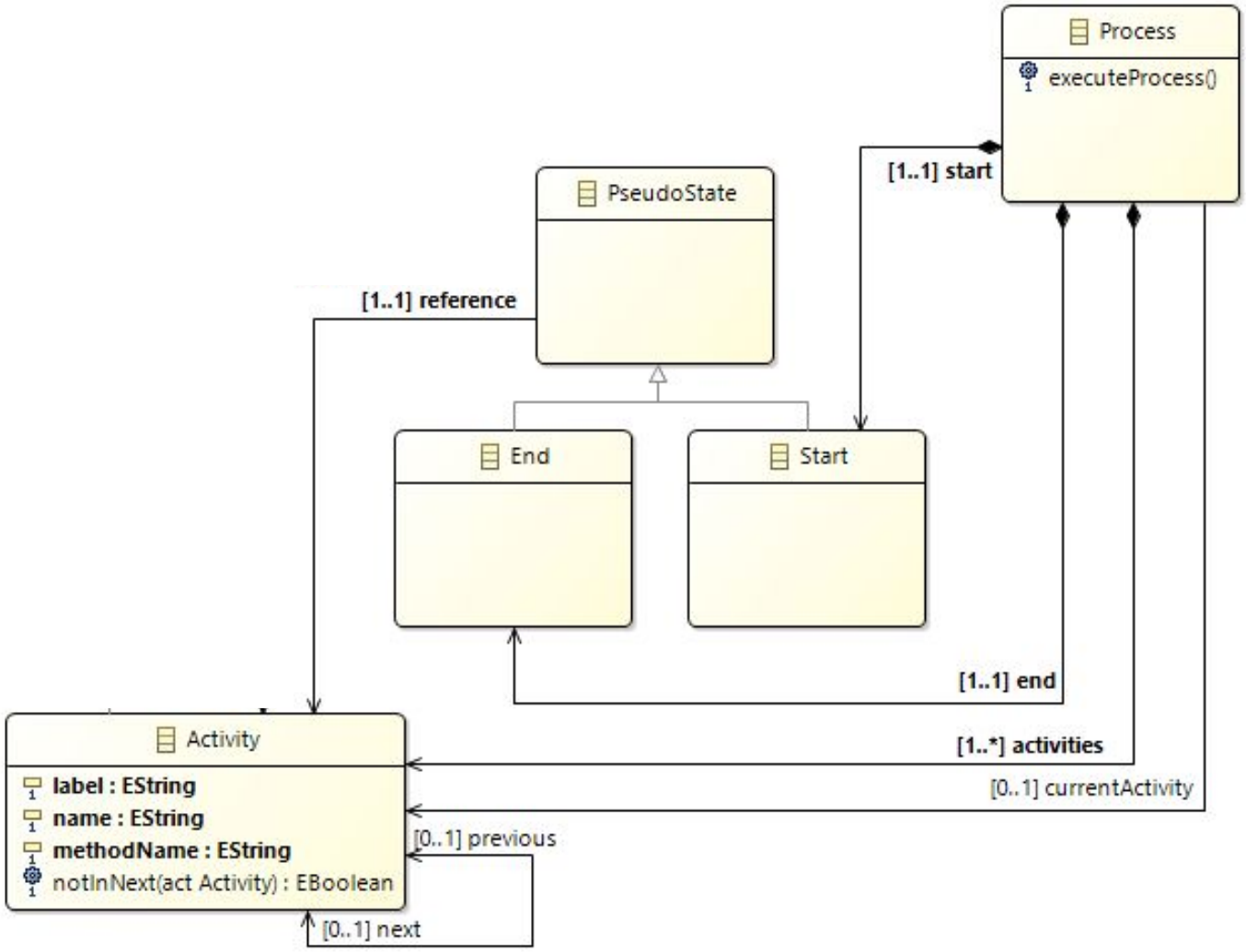
Xmodeling Studio

- Environnement de développement basé sur EMF
  - Pour assister les ingénieurs
- Trois niveaux d'ingénierie et d'acteurs
  - Développeur de Xmodeling Studio
    - Offre des éléments génériques aux autres ingénieurs
  - Ingénieur de langage
    - Définit un DSL exécutable et livre un moteur prêt à l'emploi
  - Ingénieur de développement
    - Définit des modèles conformes au DSL, associe des opérations métier et intègre le moteur d'exécution dans son projet particulier

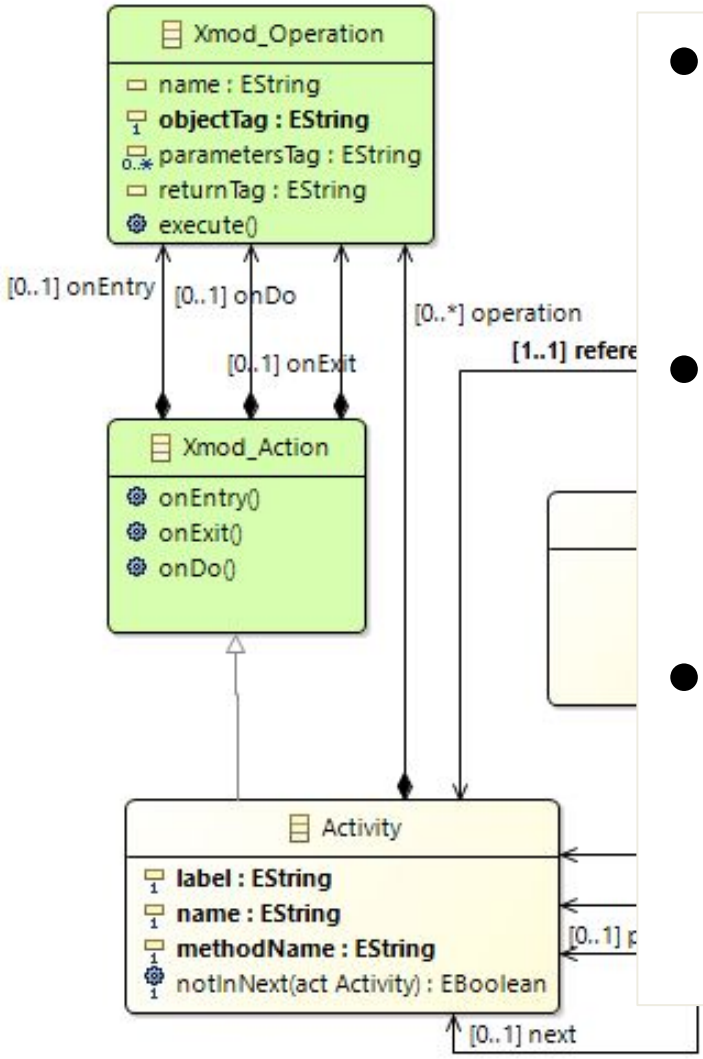
Xmodeling Studio (Use Cases)



Exemple : PDL


Process
executeProcess()
[1..1] start
PseudoState
[1..1] reference
End
Start
[1..1] end
Activity
label : EString
name : EString
methodName : EString
notInNext(act Activity) : EBoolean
[1..*] activities
[0..1] currentActivity
[0..1] previous
[0..1] next

Exemple : PDL étendu



- Par méta-transformation
  - Eléments avec comportement sont annotés
  - Ici ajout par héritage de la définition d'opérations aux activités
- Opération définie par
  - Tags sur les objets paramètres
  - Tag sur la valeur possible de retour
  - Tag sur l'objet sur lequel elle est appelée
- Coté Java
  - Une map associe un tag avec un objet Java
  - Code fourni pour exécuter automatiquement les opérations via ces tags

PDL : moteur exécution

- Code principal : méthode *executeProcess()* de *Process*

```
public void executeProcess() {
  // on recupere la premiere activite du processus
  Activity act = this.getStart().getReference();
  do {
    // execution des operations de l'activite
    // si elles sont definies
    act.onEntry();
    act.onDo();
    act.onExit();
    // on passe a l'activite suivante
    act = act.getNext();
    // fin de boucle si il n'y a plus d'activite
  } while (act != null);
}
```

- Nos classes fournies exécuteront directement le modèle chargé (instance *de Process*) avec opérations métier associées
  - Invocation dynamique de Java

PDL : utilisation moteur

- Exemple d'un modèle avec deux activités avec opérations appelées de signature avec des tags
  - *returnVal m1(nb) on metier*
  - *void m2(returnVal) on metier*

```
// on cree le contenu initial de la map avec les objets sur
// lesquels s'appliquent les operations metiers
HashMap<String, Object> map = new HashMap<>();
Metier metier = new Metier(...);
map.put("metier", metier);
map.put("nb", new Integer(12));
// on charge le contenu de notre modele avec notre
// classe utilitaire generee
Process proc = PDLXmodUtil.loadProcess("modele.xmi");
// on affecte la map a notre classe utilitaire generee
PDLXmodUtil.setMap(map);
// on execute le processus
proc.executeProcess();
```

Xmodeling Studio : utilisabilité

- Le DSL exécutable peut s'intégrer à tout code ou frameworks Java existants, sans contrainte
- Intégration aisée à un projet :
  - Ajouter 3 librairies (.jar) EMF (2 Mo)
  - Ajouter la librairie (.jar) du moteur du DSL
  - Fournir un modèle (.xmi) combinant le comportement et le métier de l'application

    Note : ce fichier peut être statique ou même dynamique !
- Testé avec succès pour développer une application mobile avec Android Studio utilisant le moteur PDL

Conclusion

- IDM pragmatique : intégrer l'IDM à du développement standard au lieu de vouloir le remplacer
- Application aux DSL exécutables
  - Xmodeling Studio permet d'intégrer l'aspect métier et le flot de données avec un DSL exécutable
  - Utilisable pour n'importe quel projet Java, pour n'importe quel DSL exécutable
- Perspectives
  - Guider encore d'avantage l'ingénieur de DSLs exécutables

Soon on www.pauware.com !