UNIVERSITÉ DE NANTES

Génération de contrôleurs d'automates depuis des modèles UML

Raffinement de modèles de contrôle

MSR 2019

Pascal André, Yannis Le Bars

LS2N, Université of Nantes, France

Introduction

- **Contexte**
  Systèmes logiciels de contrôle d'automatismes

quèsaco ?

Contexte

Systèmes logiciels de contrôle d'automatismes

quèsaco ?

- Matériel + logiciel
- Contrôle (automates)
- Interaction et communication (messages)

Domaine large

- Systèmes distribués
- Logiciel embarqué
- Objects connectés

application + modeste

Point de vue du logiciel

Développement
Maintenance
Qualité

prépondérant

Source https://trends.directindustry.fr/pilz/project-7550-121067.html

Problèmes

Diversité (dispositifs, API)
Adaptation,
Evolution continue

Source: http://reseau.fing.org/blog/
Image Source: https://www.reallusion.com/event/2010/photoshop/

Introduction

- **Contexte**
  Systèmes logiciels de contrôle d'automatismes
- **Motivations**
  Appliquer de bonnes pratiques du génie logiciel

Motivations générales

Appliquer de bonnes pratiques du génie logiciel

- Modularité — Composants, services
  - Encapsulation
  - Passage à l'échelle
  - Réutilisation

Principes de conception logicielle

- Abstraction — Ingénierie des modèles
  - Raisonner au niveau des modèles (séparation des aspects)
  - Automatiser partiellement le développement logiciel
  - Traçabilité

- Qualité — Méthodes formelles
  - Modèle formel de services et contrats
  - Vérification (correction, sûreté, vivacité…)

☞ *Focus IDM*

Introduction

- **Contexte**
  Systèmes logiciels de contrôle d'automatismes
- **Motivations**
  Améliorer la production de code d'application de contrôle en s'inspirant de bonnes pratiques du Génie logiciel

  Focus
  - Ingénierie des modèles

- **Objectif**

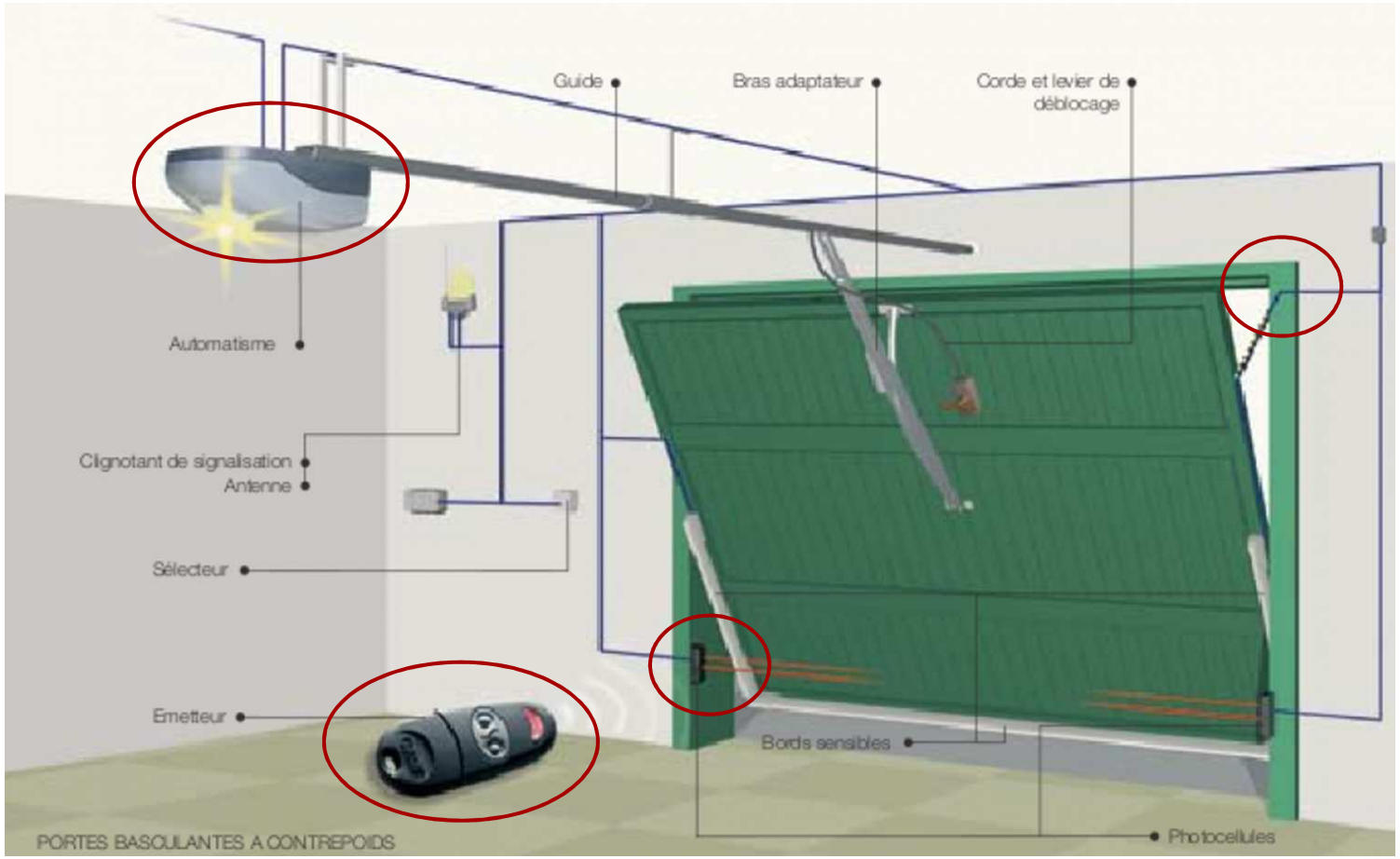
☞contribuer à la rationalisation du processus par un cadre structurant

Plan de l'exposé

- Introduction
  - ☞ contribuer à la rationalisation du processus par un cadre structurant
- Illustration : un exemple simple de domotique
- Raffinement du modèle au code
- Expérimentations
- Conclusion

Un exemple de domotique

Porte de garage

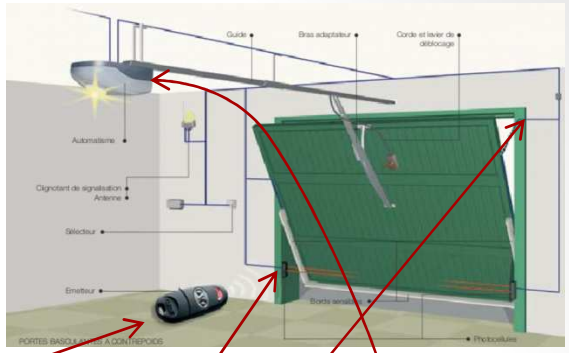


☞ *Modèle logiciel*

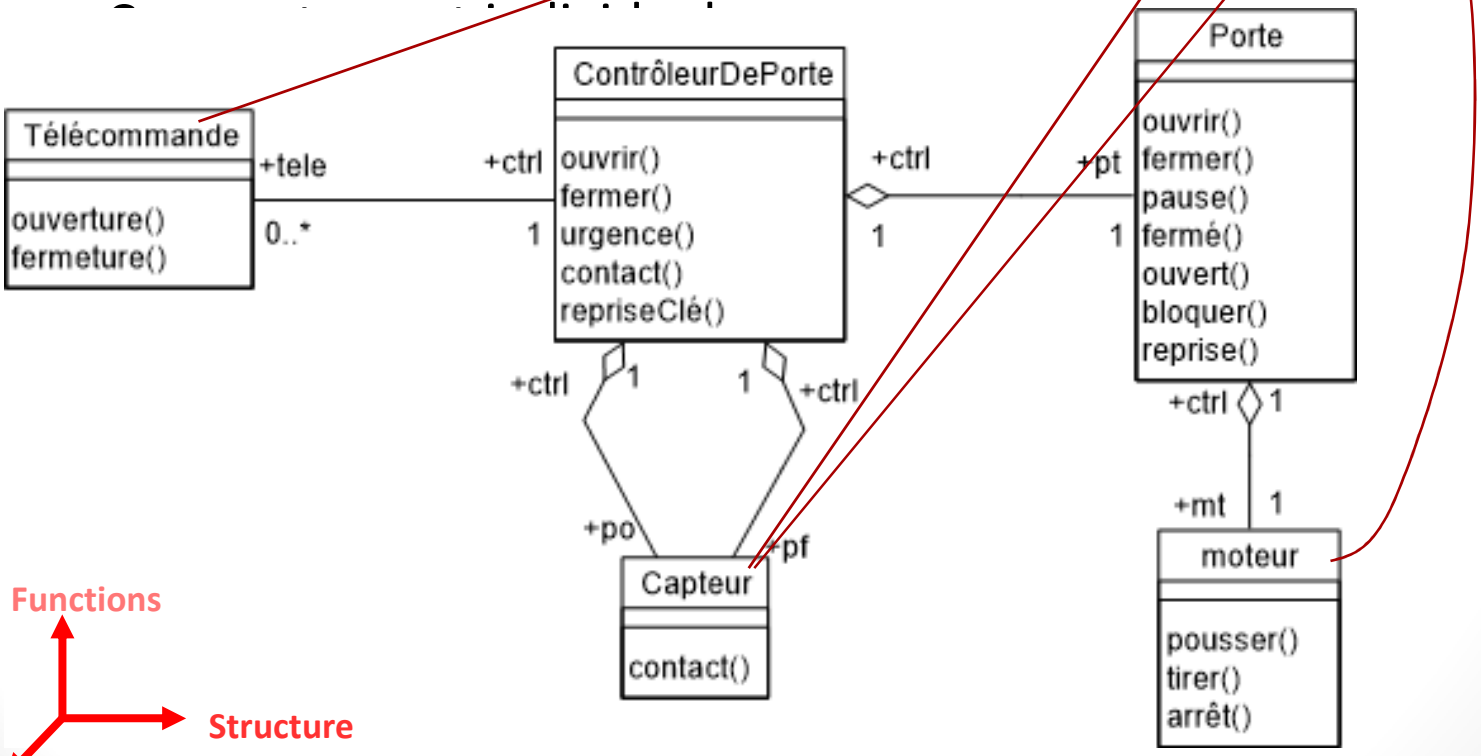
Source: https://www.bricozor.com/automatisme-portes-garages-serie-ver-24-4400-came.html

Un exemple de domotique

Porte de garage



- Langage de modélisation expressif
  - Structure
    Diagramme de classes (UML, SysML…)

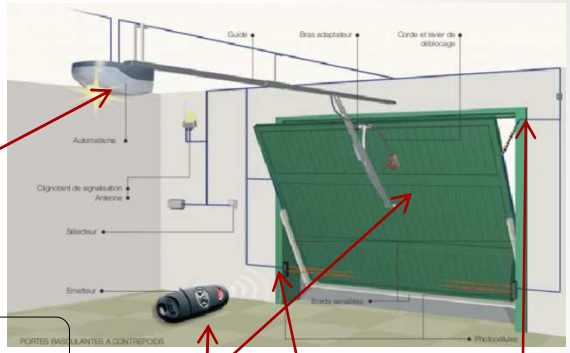


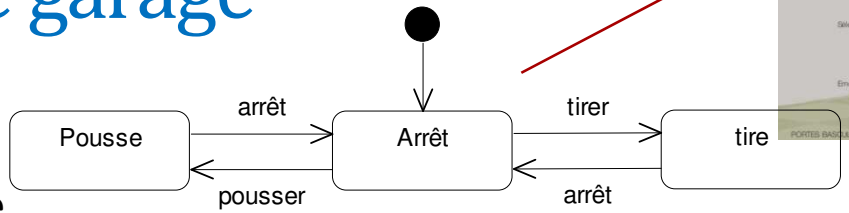
Notation UML

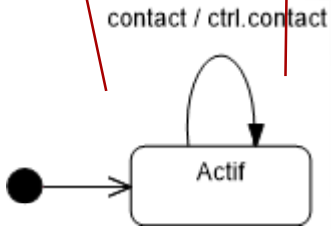
Un exemple de domotique

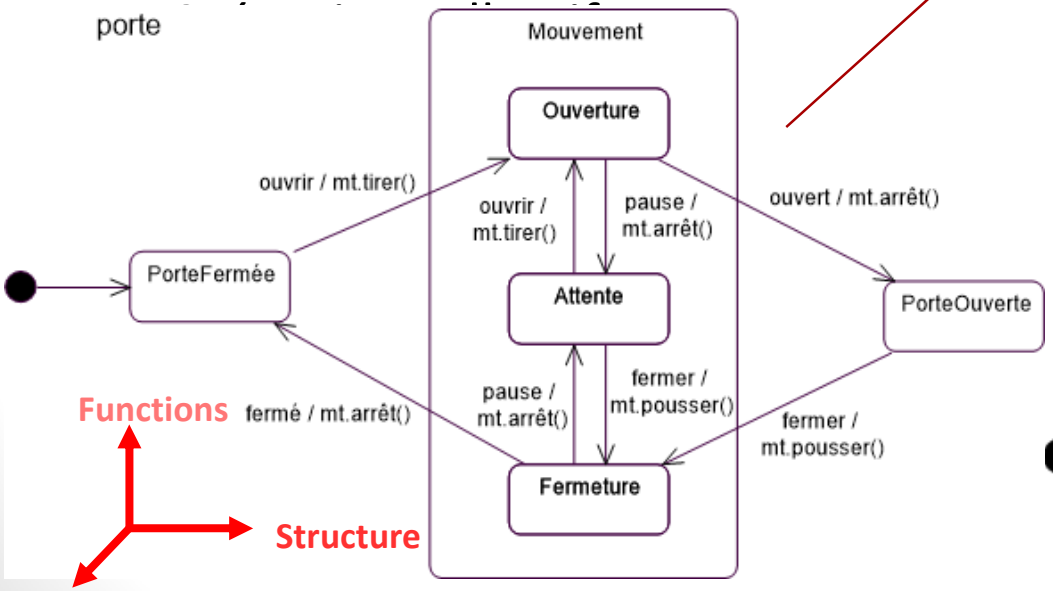
Porte de garage

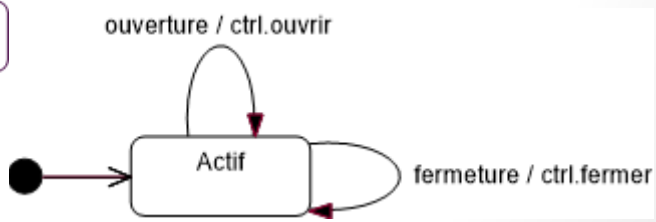
- Modèle
  - Structure
    Diagramme de classes
  - Comportement individuel
    Diagrammes états-transitions (statecharts)





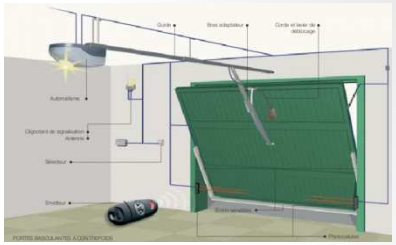




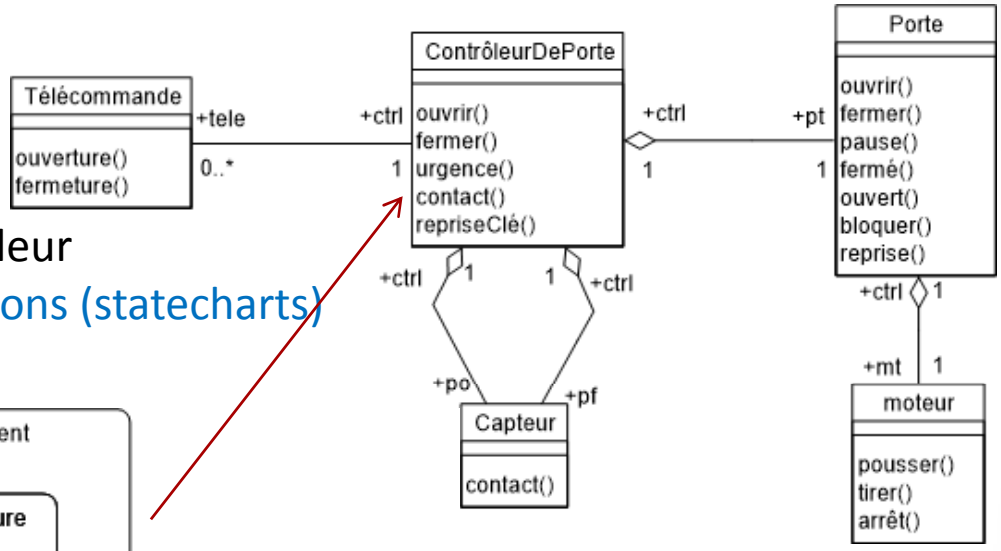


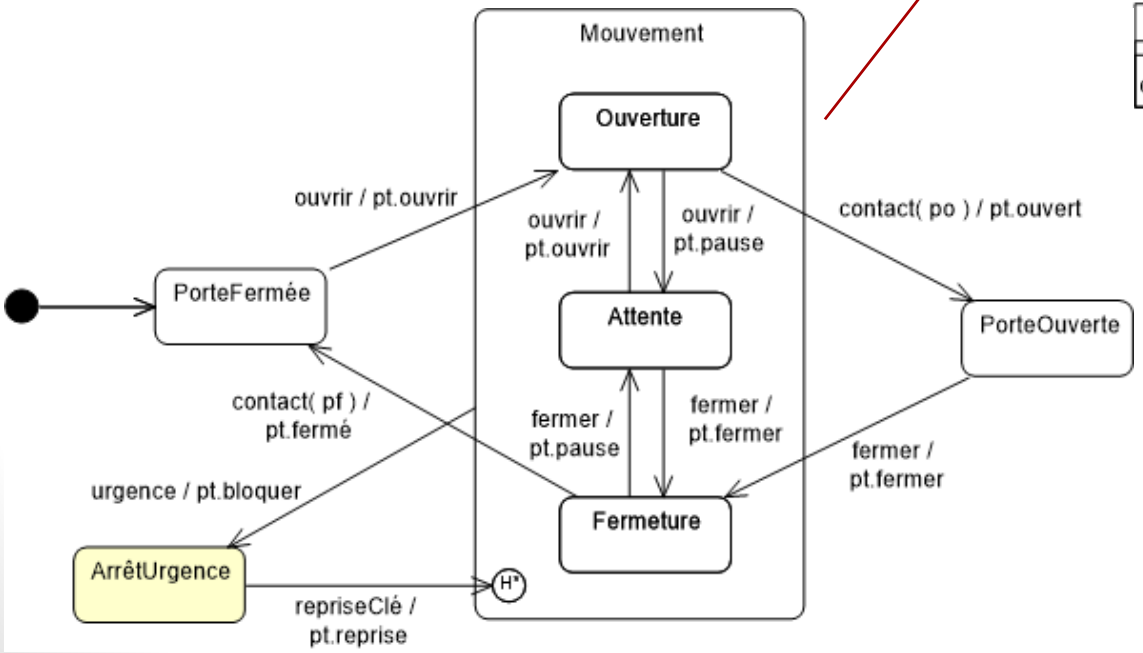
Notation UML

Un exemple de domotique
Porte de garage



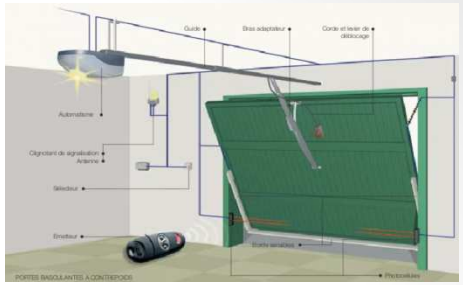
- Modèle multi-vues
  - Structure
    Diagramme de classes
  - Comportement du contrôleur
    Diagrammes états-transitions (statecharts)



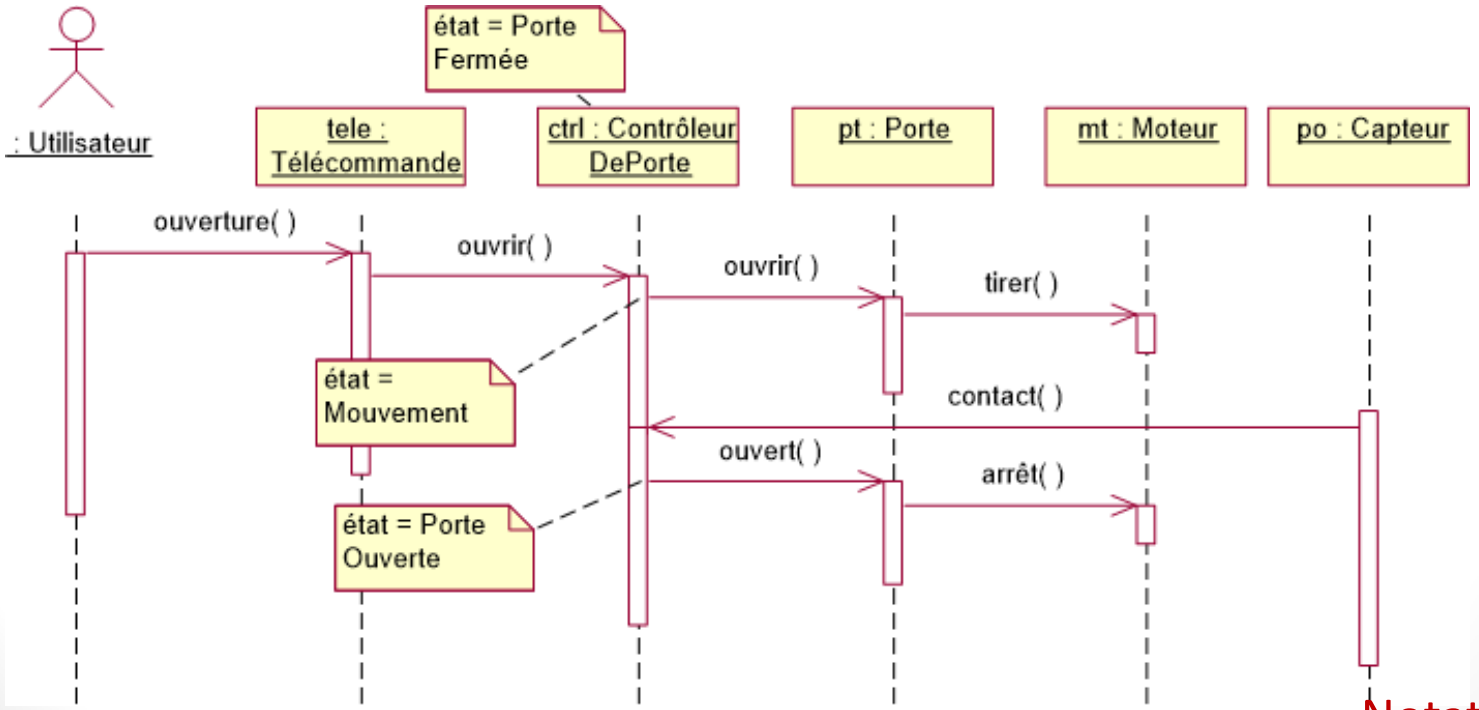


Implicite :
envois de messages

Un exemple de domotique
Porte de garage



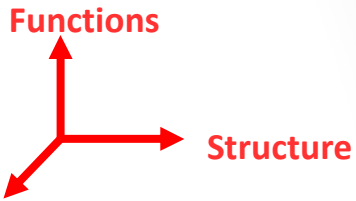
- Modèle multi-vues
  - Structure
    Diagramme de classes
  - Comportement individuel
    Diagrammes états-transitions (statecharts)
  - Scénarios collectifs (Diagrammes de *séquences*)



Notation UML

Un exemple de domotique
Porte de garage

- Modèle d'analyse multi-vues
  - Structure
    Diagramme de classes
  - Comportement individuel
    Diagrammes états-transitions (dyn)
    Diagrammes d'activités (fct)



Types =>

modèles de l'application

Implantation ?

  - Scénarios collectifs

    Diagramme de *séquences*

    Diagrammes de collaboration

    Diagrammes d'objets

Instances =>

scénarios de test de l'application

Plan de l'exposé

☞contribuer à la rationalisation du processus par un cadre structurant

- Introduction
- Illustration
- Raffinement du modèle au code
  - Développement manuel
  - Génération de code
  - Transformations de modèles
- Expérimentations
- Conclusion

Plan de l'exposé

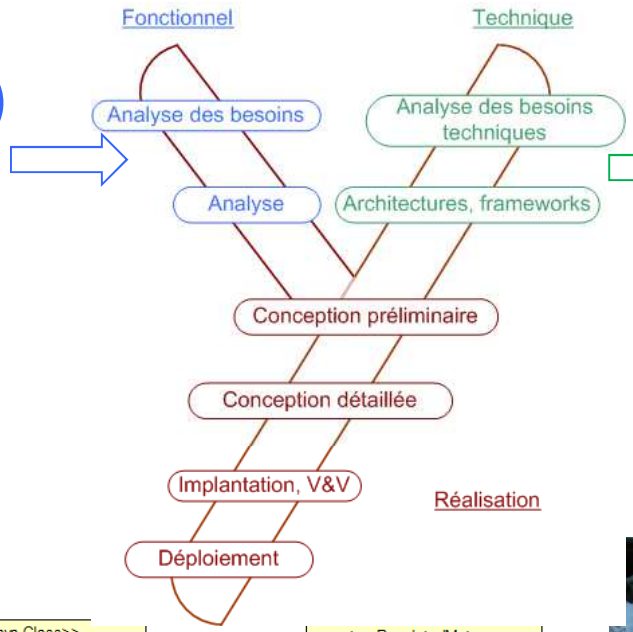
☞contribuer à la rationalisation du processus par un cadre structurant

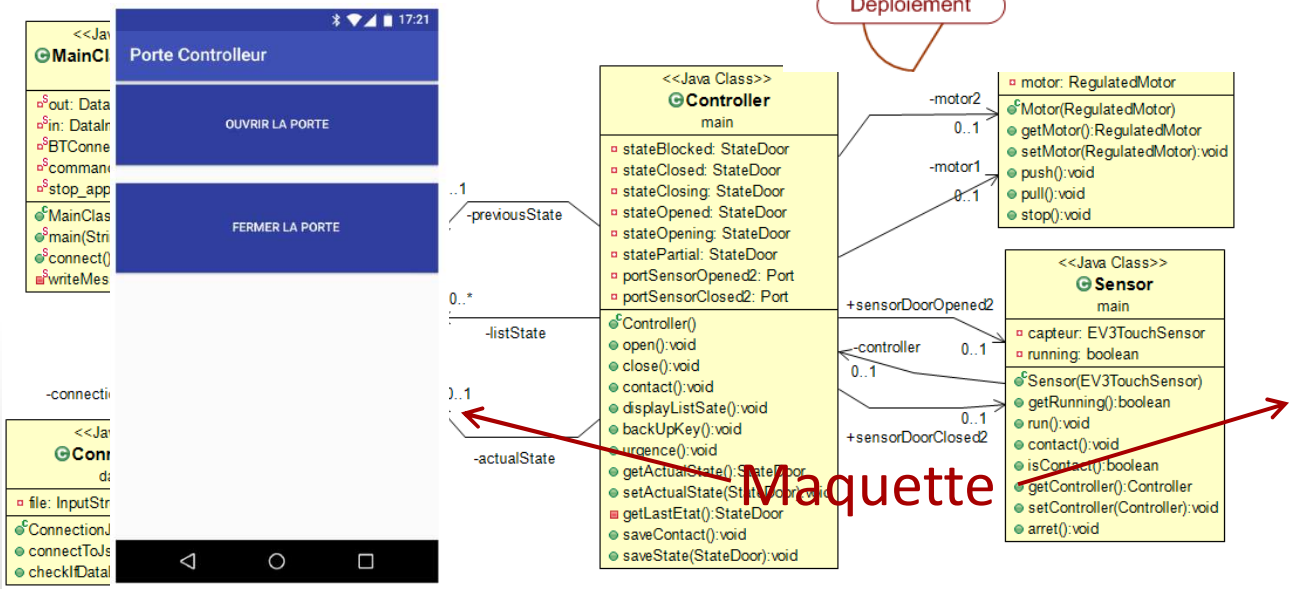
- Introduction
- Illustration
- Raffinement du modèle au code
  - Développement manuel (conception et programmation)
  - Génération de code
  - Transformations de modèles
- Expérimentations
- Conclusion

Raffinement du modèle au code

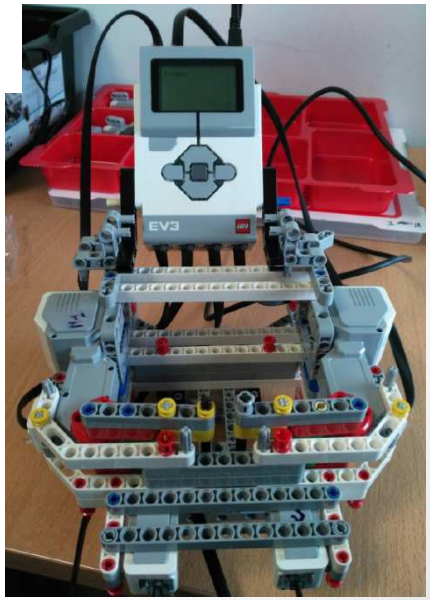
Développement manuel

- Modèle logique (UML)
- Conception / Codage
- Test





Maquette

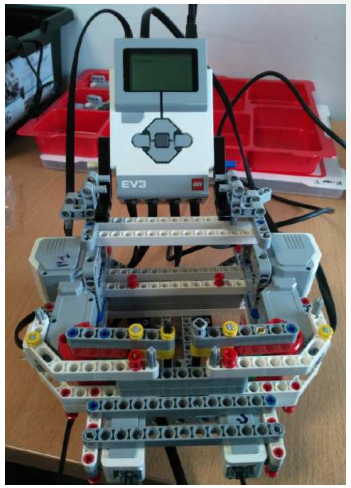


Raffinement du modèle au code
Développement manuel

- Processus (détails)
  - Montée en compétence fonctionnelle (modèle sert de référence)
  - Montée en compétence technique (Lejos, Android)
  - Décisions de conception
    - communication (sockets, bluetooth)
    - Automates
    - IHM (android)
  - Décisions de programmation
    - branchement API Lejos
    - branchement android
    - Implantation communication (authentification)
    - Etc.

Raffinement du modèle au code
Développement manuel

- Processus (détails)
  - Montée en compétence fonctionnelle
  - Montée en compétence technique (Lejos, Android)
  - Décisions de conception
  - Décisions de programmation



- Résultat opérationnel
  - Le modèle est une référence mais pas une abstraction stricte
  - L'expertise influe sur le résultat
  - La qualité "capacité fonctionnelle » prime
  - Le raffinement de communication est un problème à part

Automatisation ?

Plan de l'exposé

☞contribuer à la rationalisation du processus par un cadre structurant

- Introduction
- Illustration
- Raffinement du modèle au code
  - Développement manuel
  - Génération de code (traduction, exécution)
  - Transformations de modèles
- Expérimentations
- Conclusion

Raffinement du modèle au code

Génération de code

- AGL / Editeurs UML
- Couverture : statique / comportemental (dynamique + fonctionnel)
- Cible : générique / spécifique à une plateforme
- Intégration : one shot / roundtrip
- Résultat : simulation / application ☞ *Le générateur n'invente pas, il faut que les modèles d'entrée soient riches.*

TABLE 1 – Comparaison de quelques outils avec générateurs de code

	Star UML	Papyrus	Yakindu	Modelio	VisualParadim	IBM rational rhapsody
Version UML	2.0	2.5	-	2.4.1	2.0	2.4.1
DC	√	√	-	√	√	√
DET	-	-	√	√[1]	√	√
Operations	-	√	-	RndTrip	RndTrip	√
MOM	-	-	-	-	-	-
API Mapping	-	-	-	-	-	-
Round-trip	-	-	-	√	√	√
Licence[d]	F, C	O	F, C	O	C	C

☞ *Problème de la distance entre les abstraction et les frameworks techniques*

Plan de l'exposé

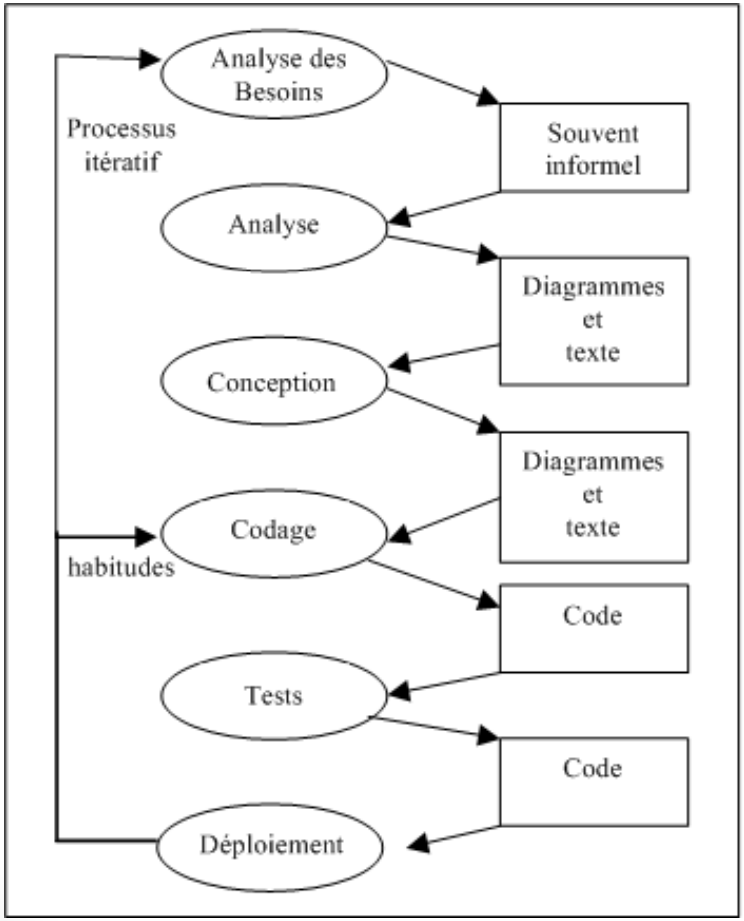
☞contribuer à la rationalisation du processus par un cadre structurant

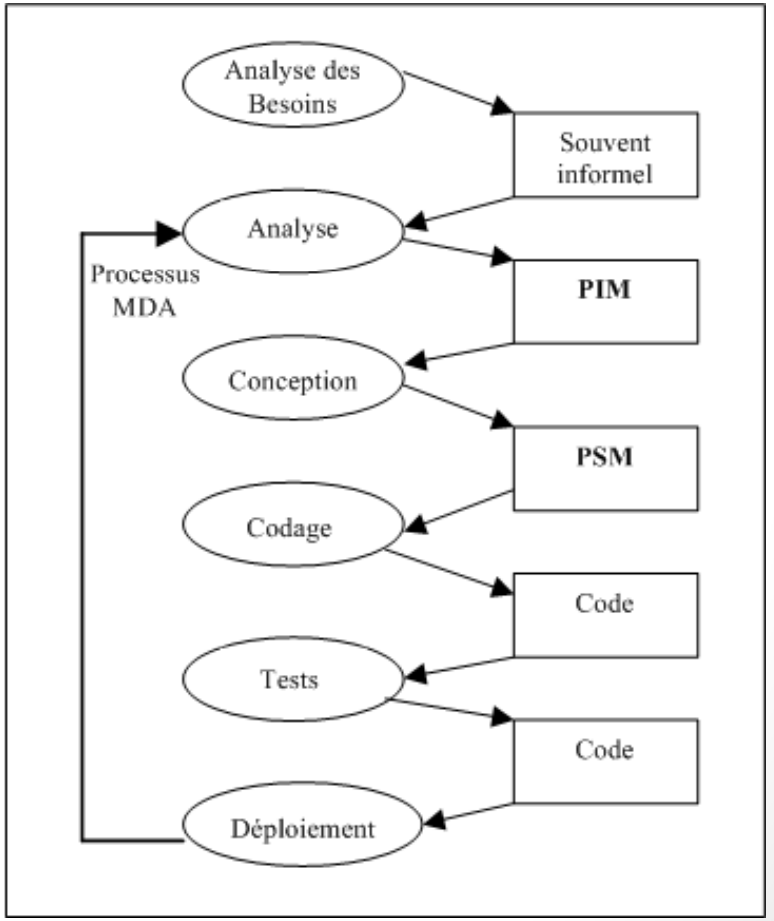
- Introduction
- Illustration
- Raffinement du modèle au code
  - Développement manuel
  - Génération de code
  - Transformations de modèles (MDA)
- Expérimentations
- Conclusion

Raffinement du modèle au code

Transformations de modèles

- Principe (MDA)

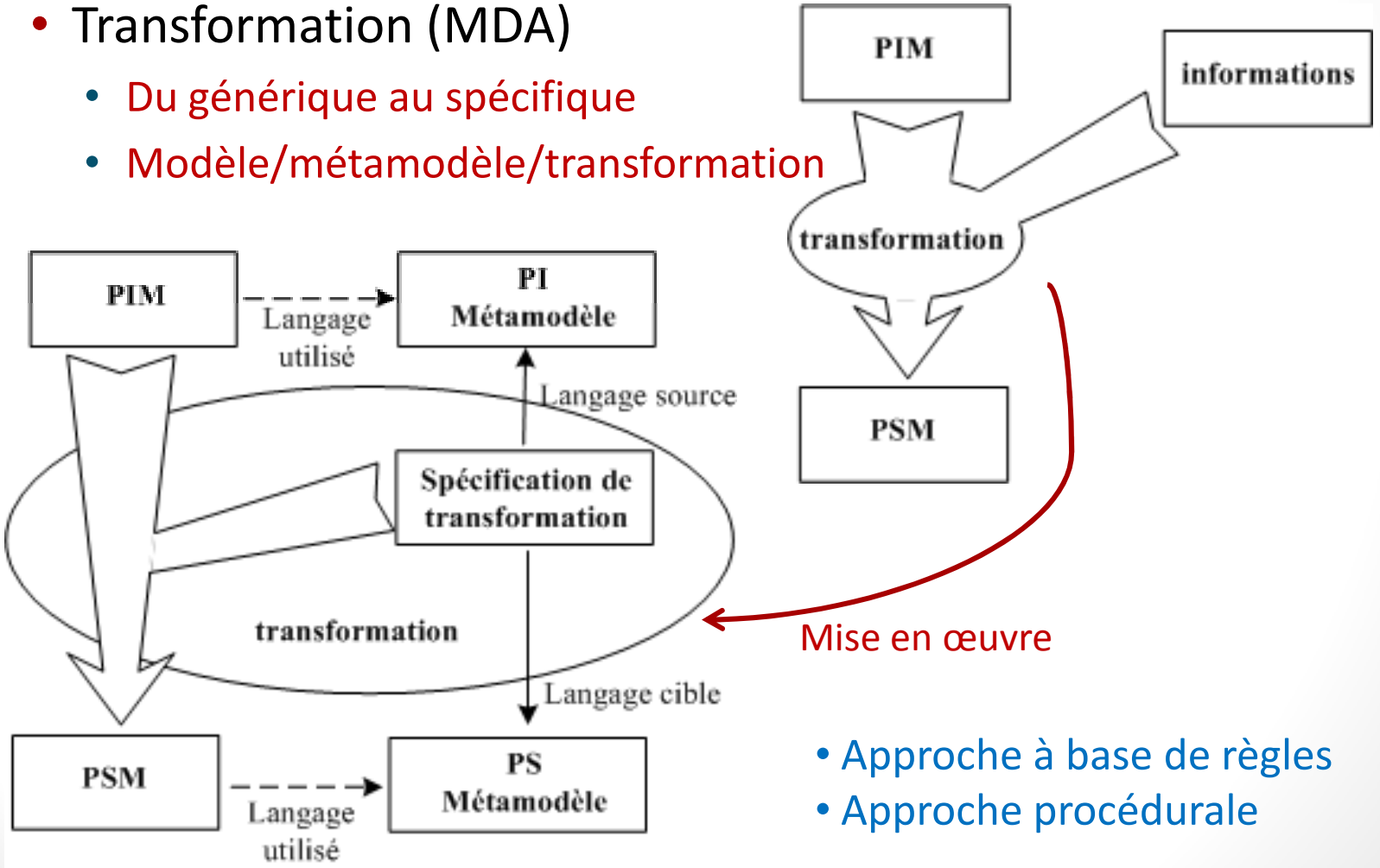




Raffinement du modèle au code

Transformations de modèles

- Transformation (MDA)
  - Du générique au spécifique
  - Modèle/métamodèle/transformation

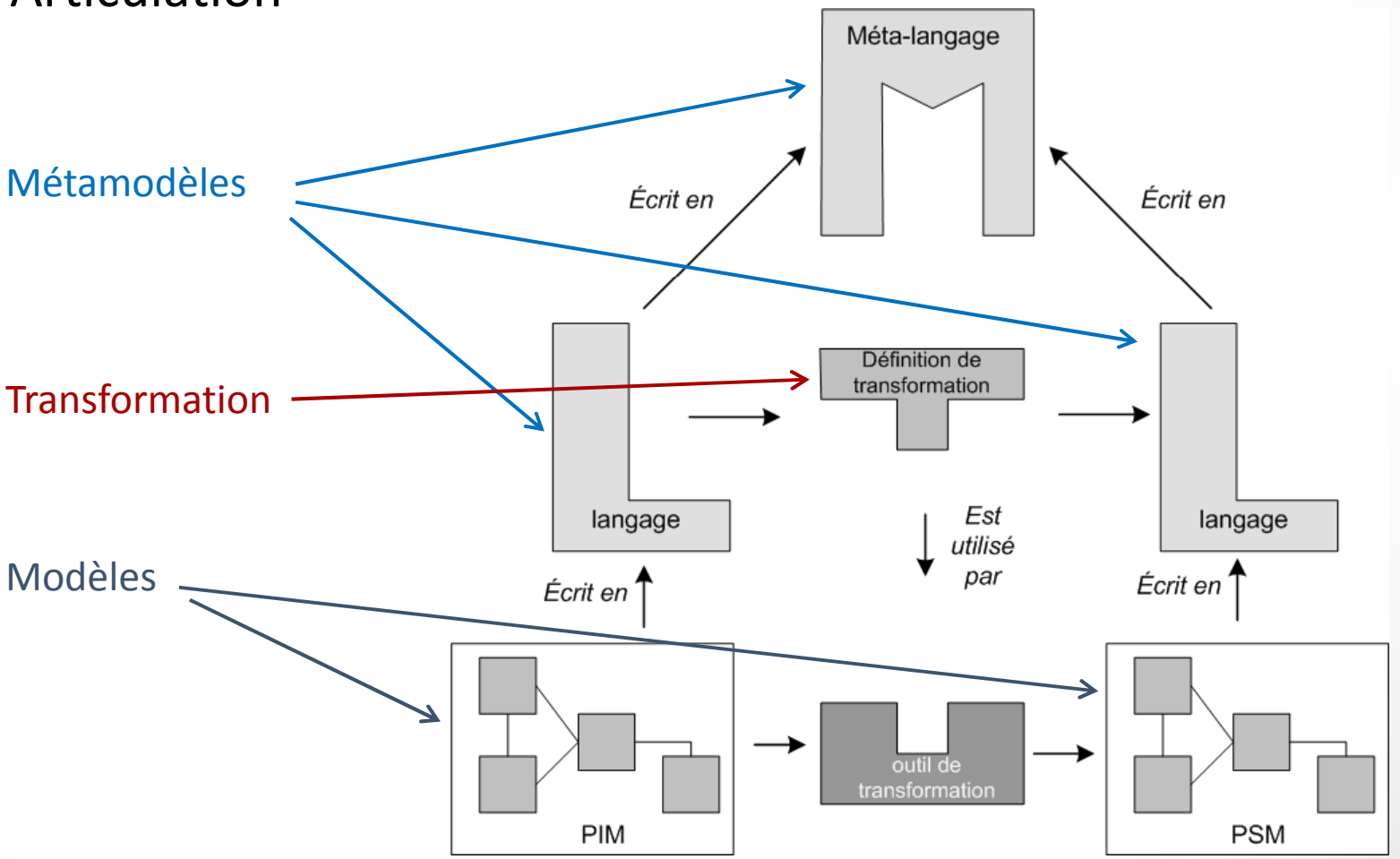


- Approche à base de règles
- Approche procédurale

Raffinement du modèle au code

Transformations de modèles

- Articulation



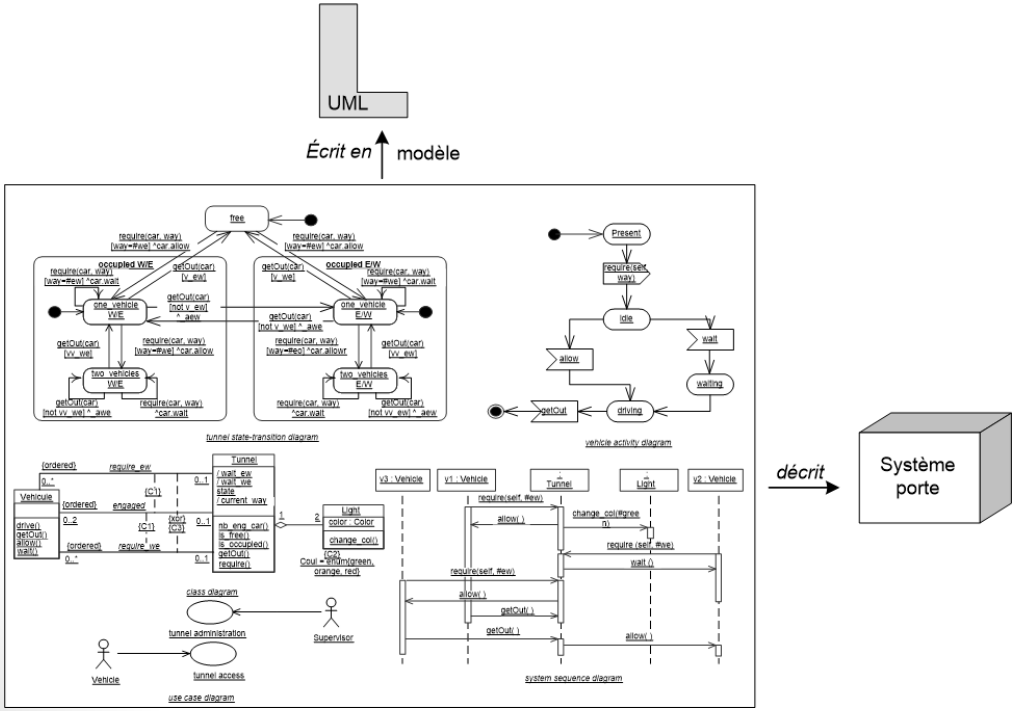
Raffinement du modèle au code

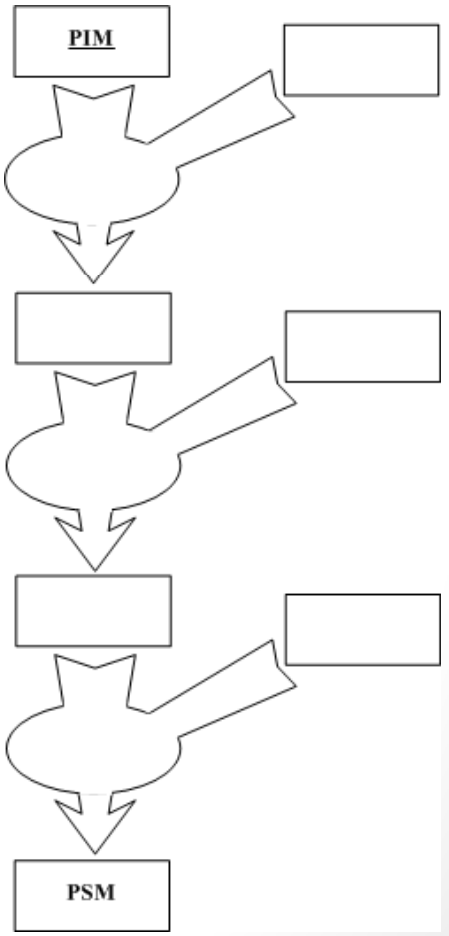
Transformations de modèles

- Rapprochement progressif avec le framework technique

  Processus de transformations

- Problème complexe
  - Hétérogénéité du modèle d'analyse





Raffinement du modèle au code

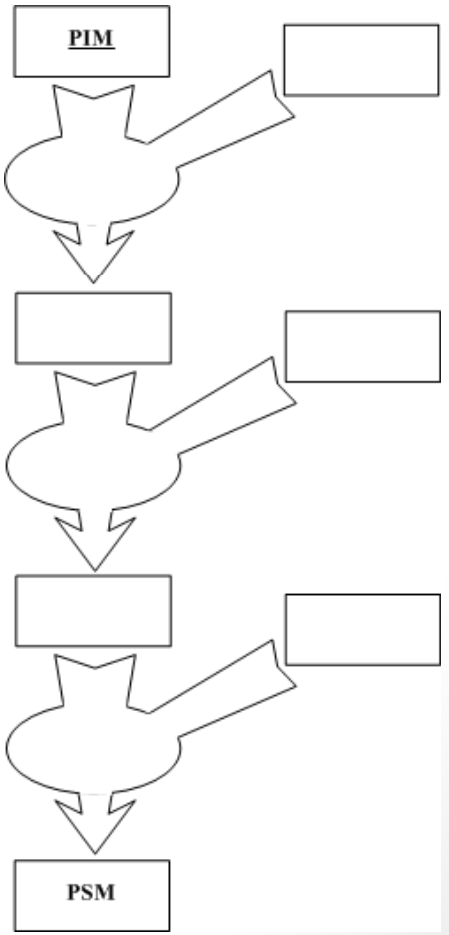
Transformations de modèles

- Rapprochement progressif avec le framework technique

  Processus de transformations

- Problème complexe
  - Hétérogénéité du modèle d'analyse
  - Distance entre le modèle d'analyse et le modèle technique (middleware, paradigmes de codage…)
  - Zoologie de langages (DSL) par processus de raffinement

implicite en manuel



- Structuration hiérarchique
  - Transformations
  - Processus

☞*expérimenter*

Plan de l'exposé

☞contribuer à la rationalisation du processus par un cadre structurant

- Introduction
- Illustration
- Raffinement du modèle au code
  - Développement manuel
  - Génération de code
  - Raffinement par transformations de modèles
- Expérimentations

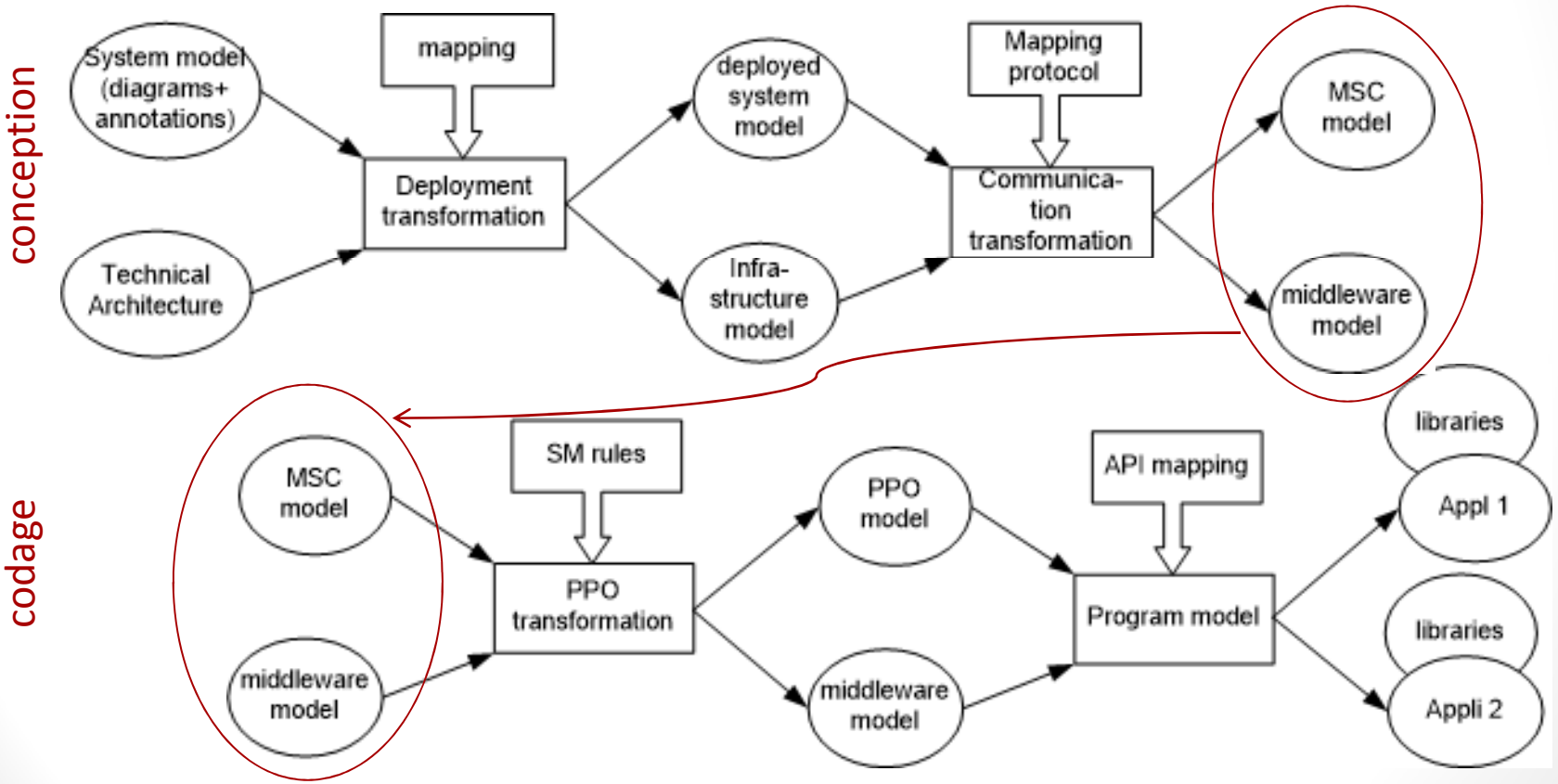
Projets étudiants

- Conclusion

Expérimentations

Transformations de modèles

- Processus de (macro-) transformations



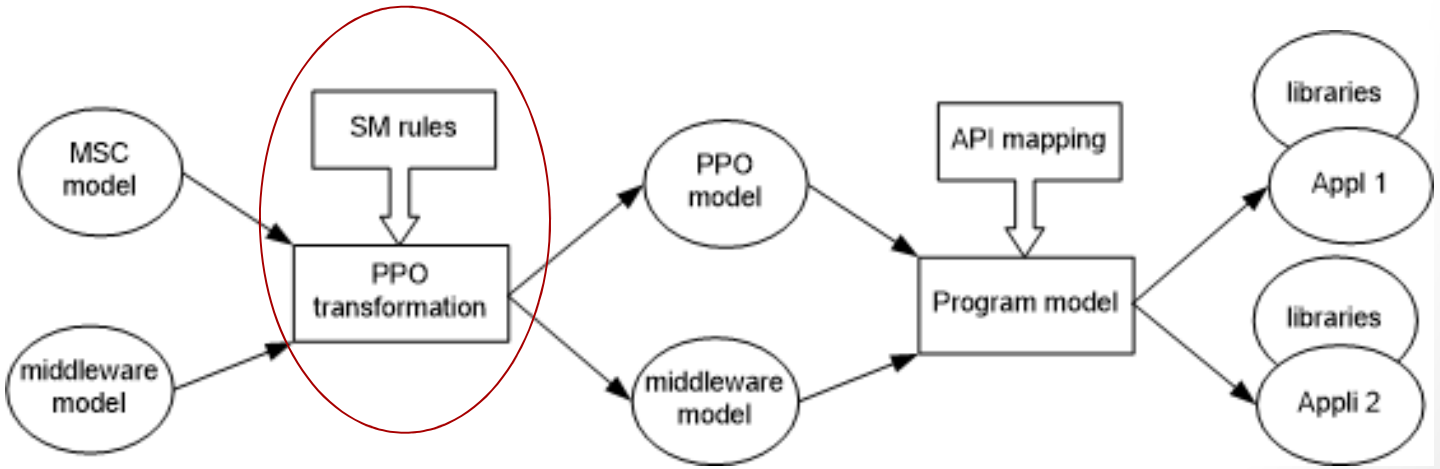
- Une macro-transformation est un processus de transformations de plus bas niveau.

Expérimentations

Transformations de modèles

- Exemple de transformation

  Modèle UML (avec statecharts) ⇨ Modèle PPO (Java)

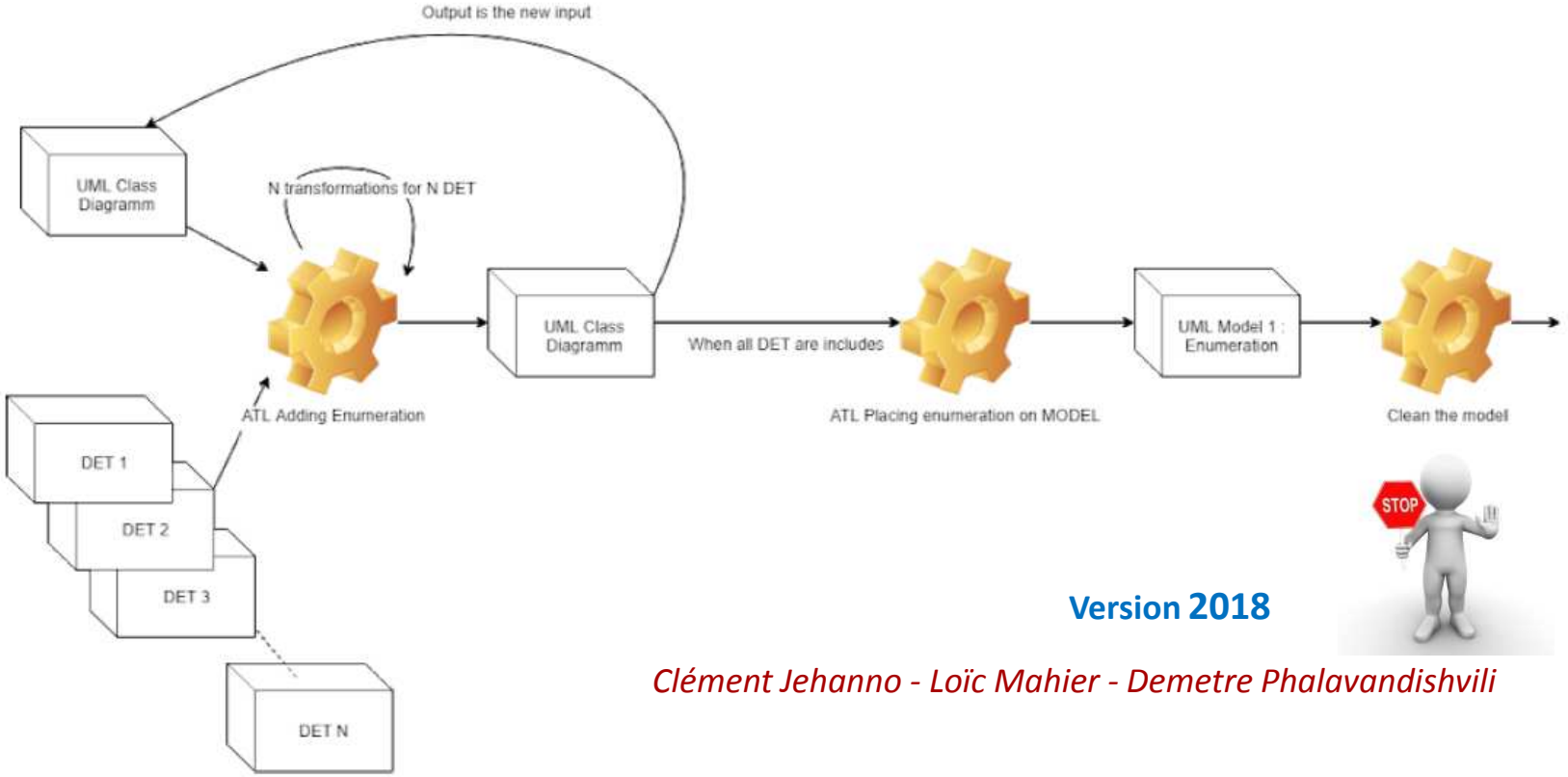


Expérimentations

Transformations de modèles

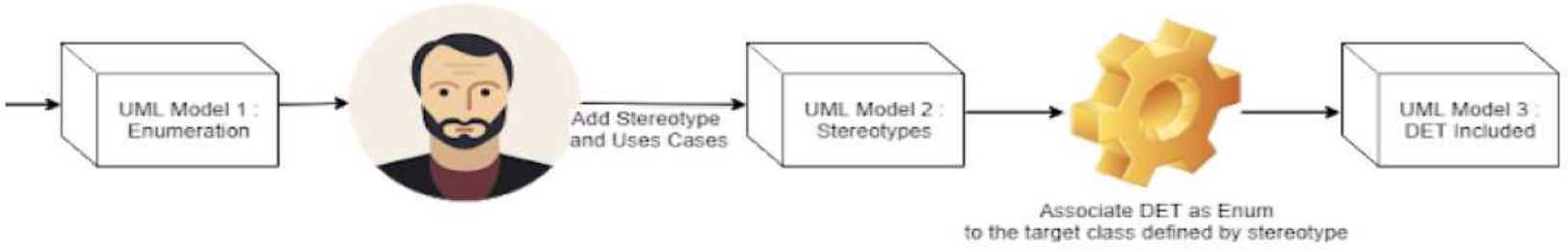
Classe UML (avec statecharts)

⇨ Classe Java



Version 2018

Clément Jehanno - Loïc Mahier - Demetre Phalavandishvili

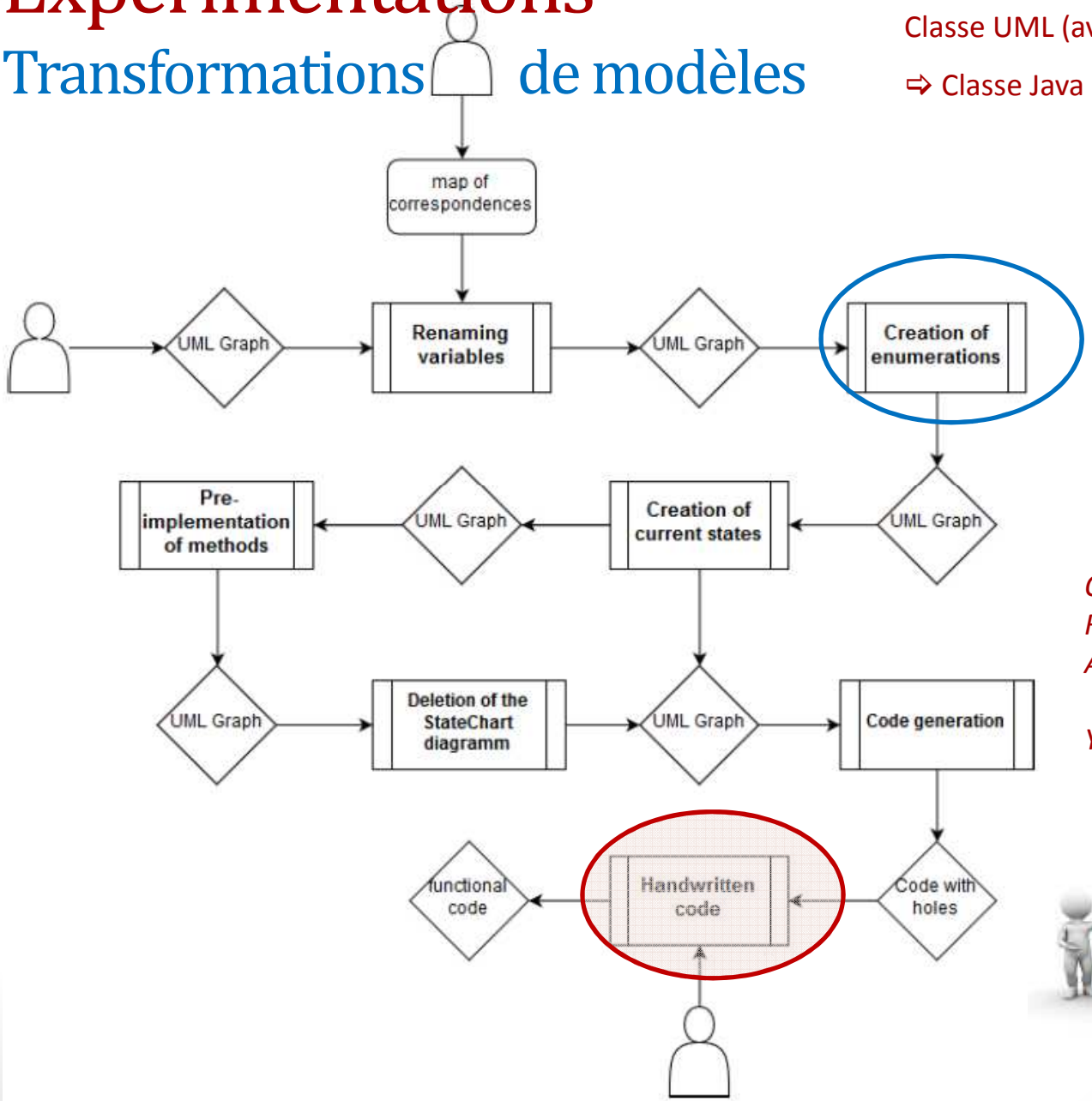


Expérimentations

Transformations de modèles

Classe UML (avec statecharts)

⇨ Classe Java



Version 2019

Oussama EL KOURRI
Ronan GUEGUEN
Antoine GODET

Yannis Le Bars

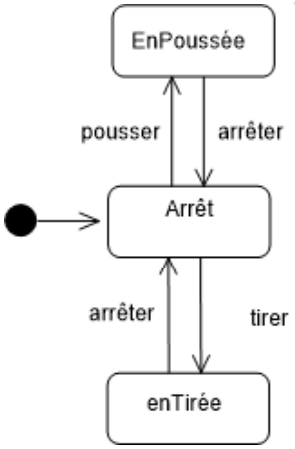
Expérimentations

Transformations de modèles

Classe UML (avec statecharts)

⇨ Classe Java

- Exemple de transformation ATL – énumérer les états



platform:/resource/Domodoor/ModeleBase/sortie1.uml
- <Model> Modele
  - <Class> Moteur
    - <Property> ctrl : Porte
    - <State Machine> Moteur
    - <Operation> pousser ()
    - <Operation> tirer ()
    - <Operation> stoppe ()
    - <Operation> arreter ()
    - <Operation> tire ()
    - <Operation> pousse ()
    - <<Create>> <Operation> Moteur ()
    - <Enumeration> StateMoteur
  - <Class> Capteur
  - <Class> Porte
  - <Class> Controleur
  - <Class> Telecommande
  - <Association> A_ctrl_tele
  - <Association> A_ctrl_mt
  - <Association> A_ctrl_pt
  - <Association> A_ctrl_co
  - <Association> A_ctrl_cf

platform:/resource/Domodoor/ModeleBase/sortie2.uml
- <Model> Modele
  - <Class> Moteur
    - <Property> ctrl : Porte
    - <State Machine> Moteur
    - <Operation> pousser ()
    - <Operation> tirer ()
    - <Operation> stoppe ()
    - <Operation> arreter ()
    - <Operation> tire ()
    - <Operation> pousse ()
    - <<Create>> <Operation> Moteur ()
    - <Enumeration> StateMoteur
      - <Enumeration Literal> Arret
      - <Enumeration Literal> enTiree
      - <Enumeration Literal> enPoussee
  - <Class> Capteur
  - <Class> Porte
  - <Class> Controleur
  - <Class> Telecommande
  - <Association> A_ctrl_tele
  - <Association> A_ctrl_mt

Expérimentations

Bilan

- Simple dans le principe mais difficile à mettre en œuvre
  - Tâtonnements (macro- et micro- transformations)
  - Trop petites transformations (biais ATL ?)
  - Distance importante avec les concepts Java
  - Décision unitaire vs approche systématique

- Facteurs
  - Peu d'expertise ATL
  - Pas de cible précise
  - Remise en cause de choix de mise en œuvre

Exemple 1

automate simple

- Deux états ⇨ booléen et alternative
- Peu d'états ⇨ types énumérés et alternatives
- Comportement spécifique par états ⇨pattern State
- Beaucoup d'états ⇨ tables de transition

automate hiérarchique

- États séquentiels ⇨ pattern State
- États concurrents ⇨ produits d'états
- États historique, erreurs, sortie, entrée…

Expérimentations

Bilan

- Simple dans le principe mais difficile à mettre en œuvre
  - Tâtonnements (macro- et micro- transformations)
  - Trop petites transformations (biais ATL ?)
  - Distance importante avec les concepts Java
  - Décision unitaire vs approche systématique

- Facteurs
  - Peu d'expertise ATL
  - Pas de cible précise
  - Remise en cause de choix de mise en œuvre

Exemple 2

Communication tablette / EV3
- Bluetooth
- Wifi
- Transformation de protocoles ⇨ complex

Communication capteurs / EV3 / Actionneurs
- API

Les expérimentations n'ont pas abouti…

Plan de l'exposé

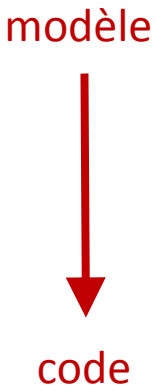
☞contribuer à la rationalisation du processus par un cadre structurant

- Introduction
- Illustration
- Raffinement du modèle au code
  - Développement manuel
  - Génération de code
  - Raffinement par transformations de modèles
- Expérimentations
- Conclusion

Conclusion

Synthèse

- Raffinement nécessaire
  - développer mais surtout faire évoluer des applications
  - améliorer la qualité par vérification des modèles
    (test du code coûteux et non suffisant)
  - rationaliser et éviter l'expertise indivisuelle (capitaliser)
  - réutiliser les composants mais aussi le savoir-faire

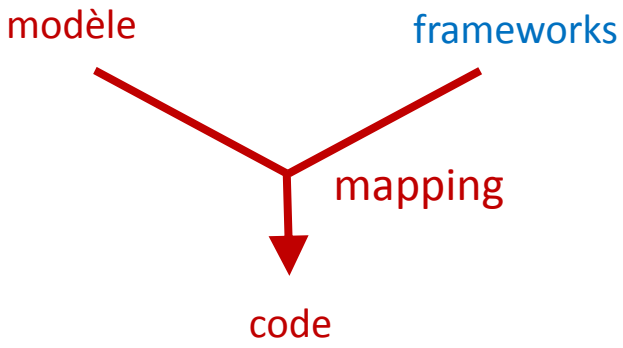


- Le problème reste épineux
  - expérimentations décevantes avec des langages de modélisation expressifs
  - outils insuffisants
  - peu exploré de manière générique dans la littérature
  - processus peu rationnel
    (concevoir et programmer =ingénierie + expérience individuelle)
  - multiplicité des environnements techniques cibles
    - …

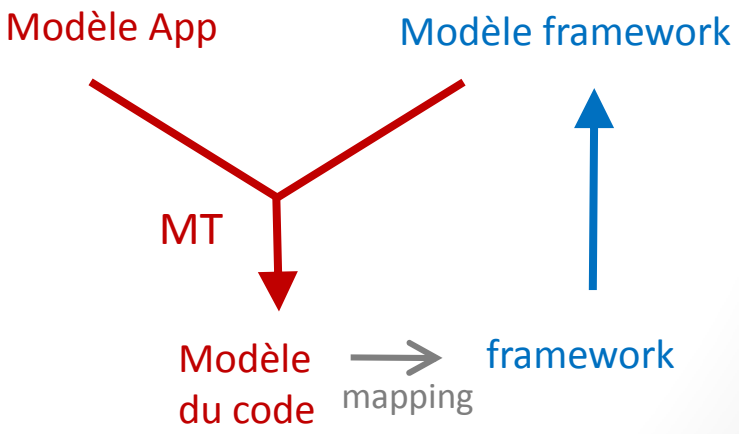
Conclusion

Points clés

- Changer la vision des transformations
  - adaptation vs raffinement



- Conserver la vision modèle
  - Tout est modèle
    (le modèle final du code induit
    la génération du code source)
  - rétro-ingénierie
    pour faciliter les transformations



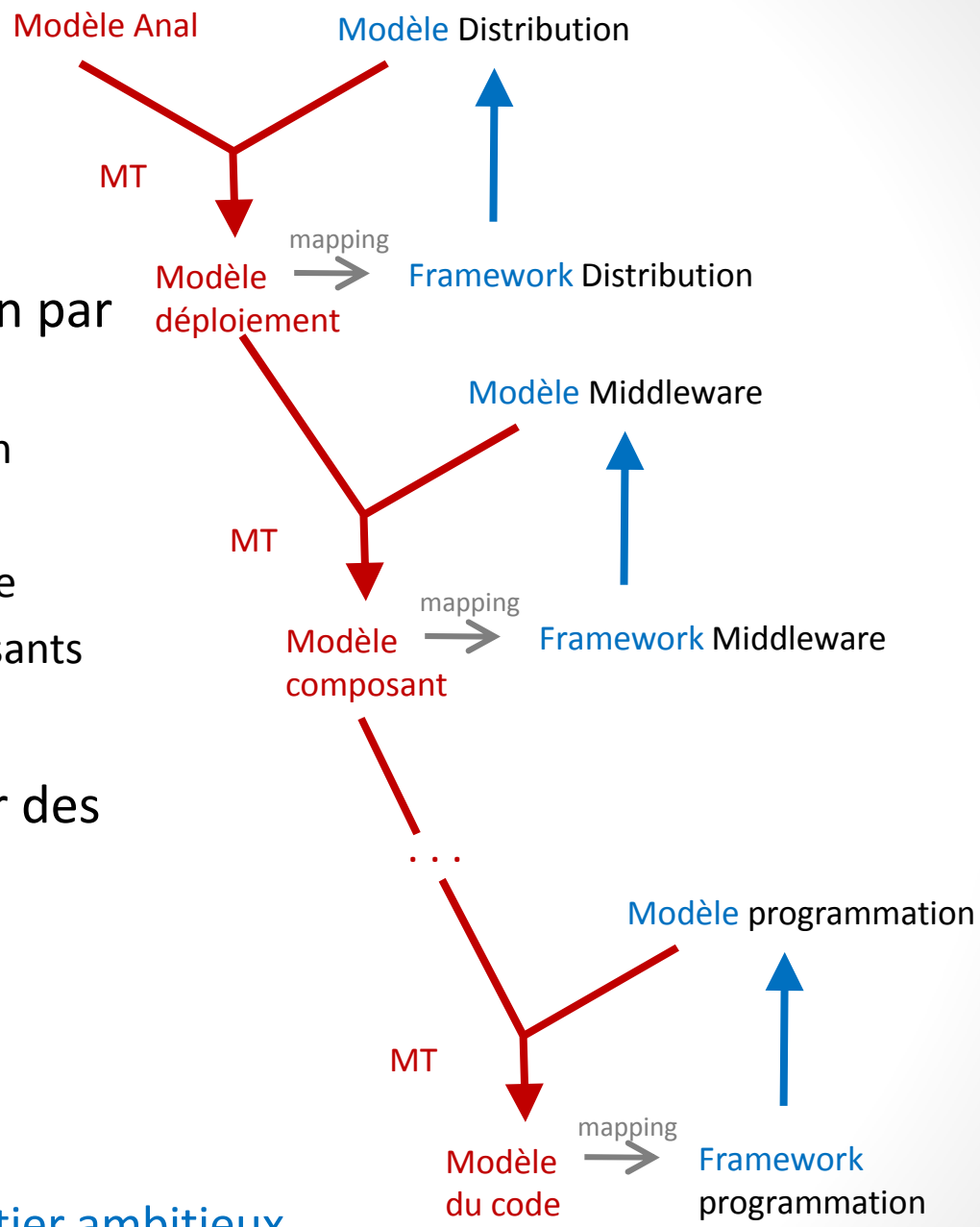
- Frameworks
  - réutilisation
  - adaptation
  - limiter les décisions
  - maîtrise technique

Conclusion

Points clés

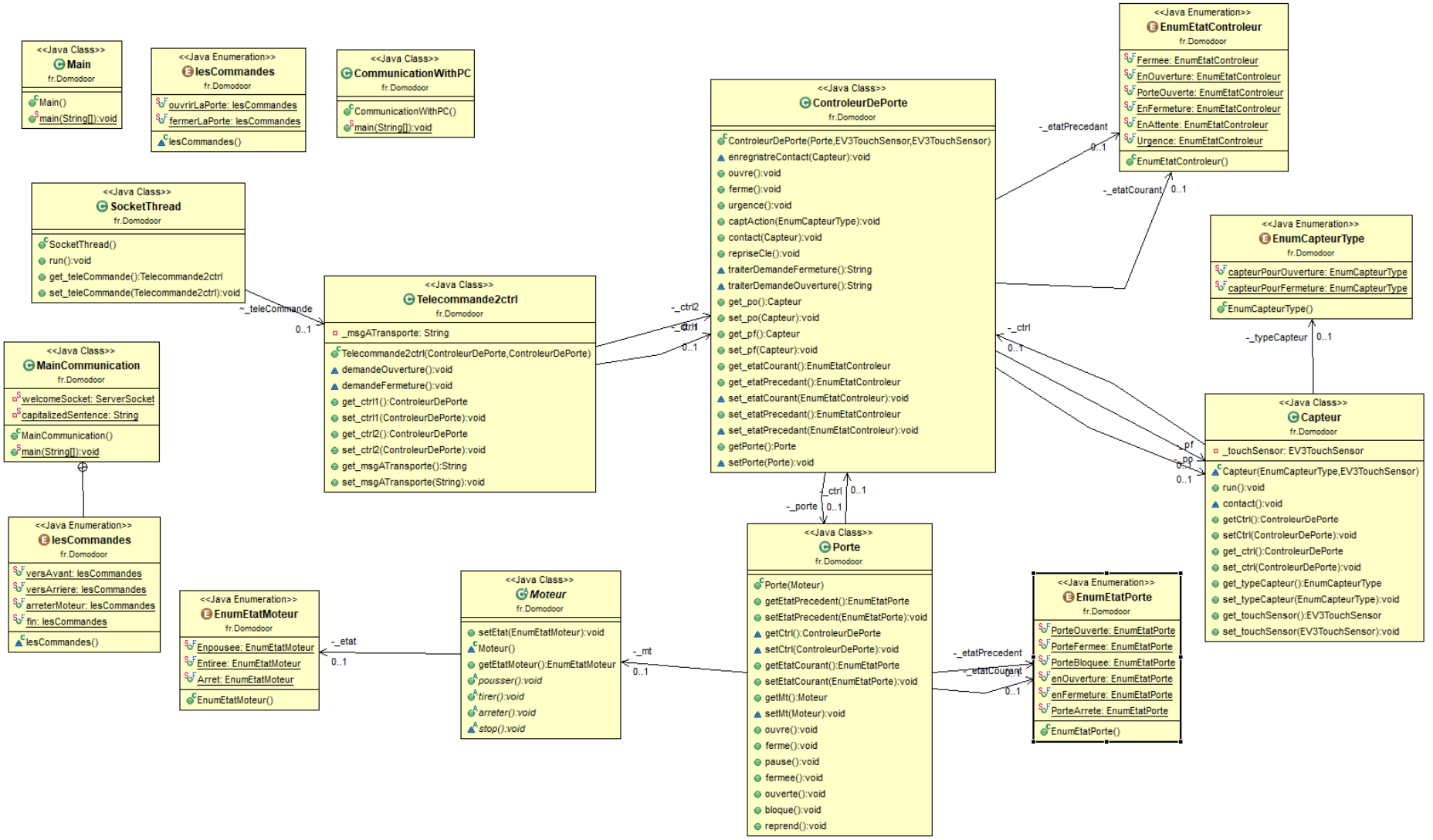
- Décisions de conception par ordre d'impact
  - Distribution et répartition
  - Middleware
  - Environnement technique
  - Bibliothèques de composants
- Parallélisme/tissage sur des aspects orthogonaux
  - Données
  - Concurrence
  - IHM
  - réseaux

Chantier ambitieux



Merci de votre attention

Raffinement de modèles de contrôle



- Rétro-ingénierie (déploiement lejos)