Modèles exécutables pour la simulation

Solution générique de lien modèle-code

Hugo RICHARD

Encadrants : Jean-Philippe BABAU, Éric CARIOU

Contexte et problématique

- Model-Driven Architecture (MDA) : tout modèle
- UML → fUML
  - Intégration de code existant ? intérêt de l'abstraction ? adoption par la communauté ?
- Approche mixte (modèle-code)
  - Structure modélisée, code en langage de programmation classique, appel du code depuis le modèle
- Apporter une nouvelle méthode de développement du logiciel

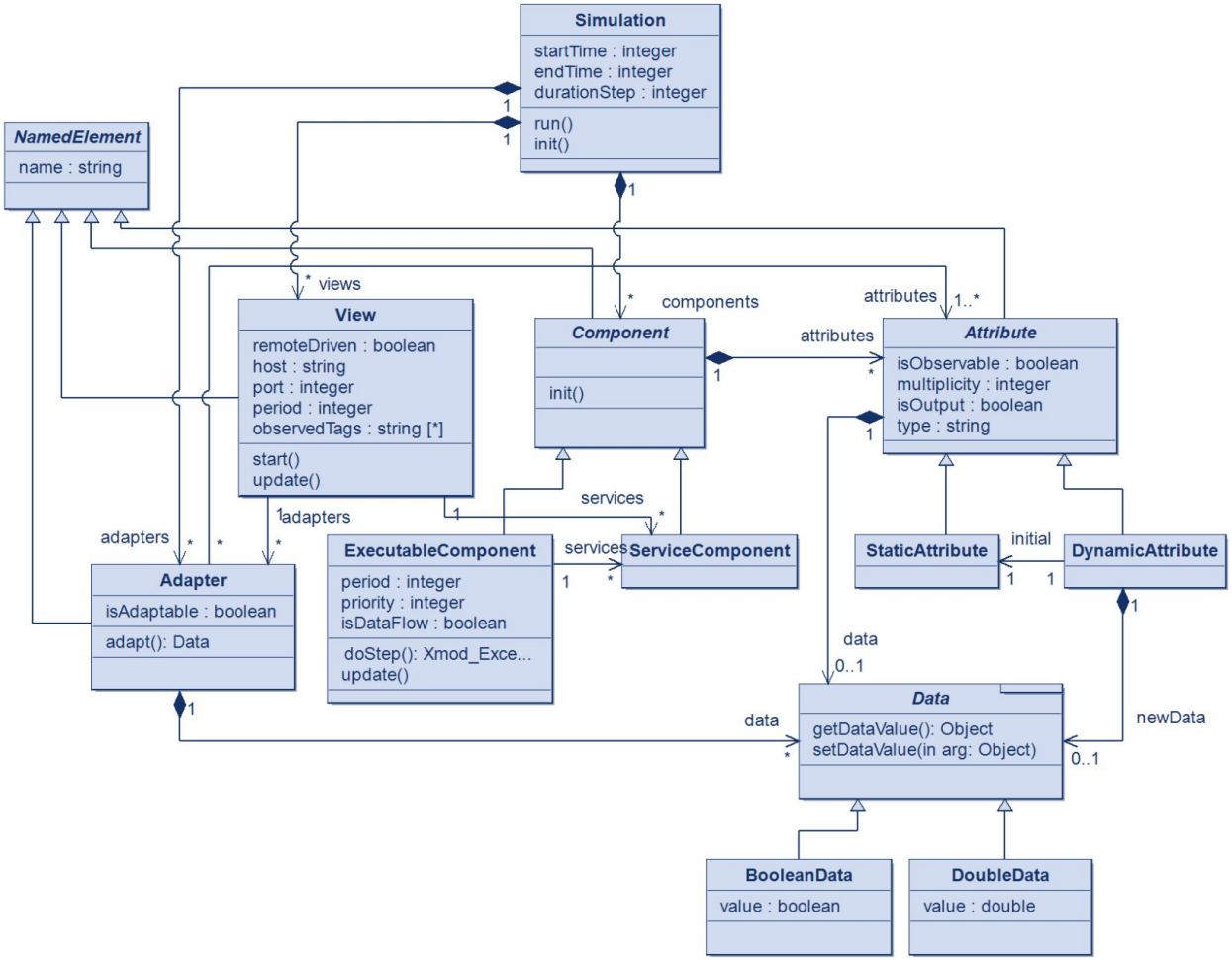
- **Solution *générique* qui s'appliquerait à n'importe quel xDSL ?**
- Extension de la première version de Xmodeling Studio

Étude de cas

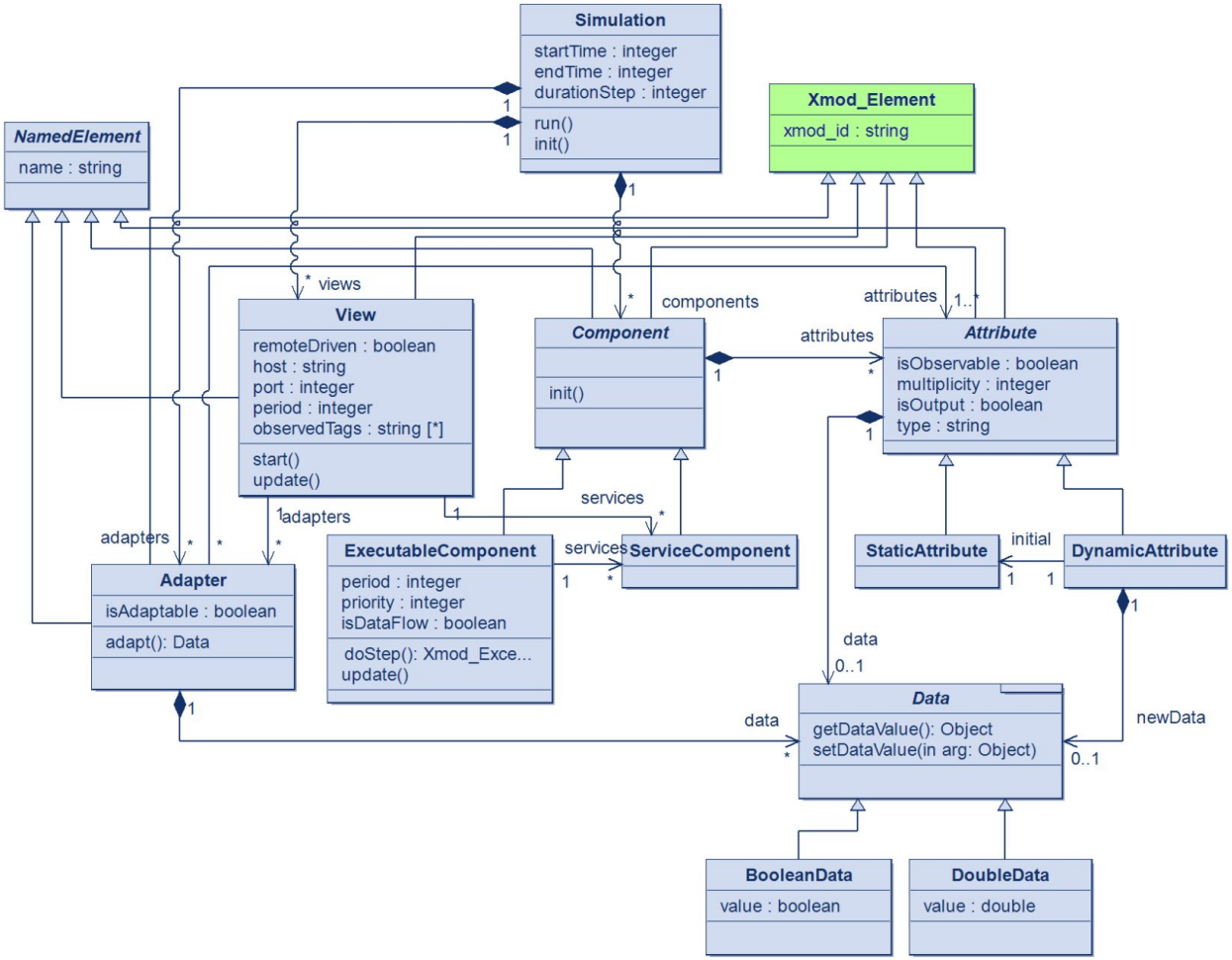
- Développement d'un simulateur de drones
- Code existant : fonctions de calculs, trajectoires (simulation)

- Ingénieur de langage
  - Méta-modèle
  - Moteur d'exécution
- Ingénieur logiciel
  - Modèle
  - Code métier

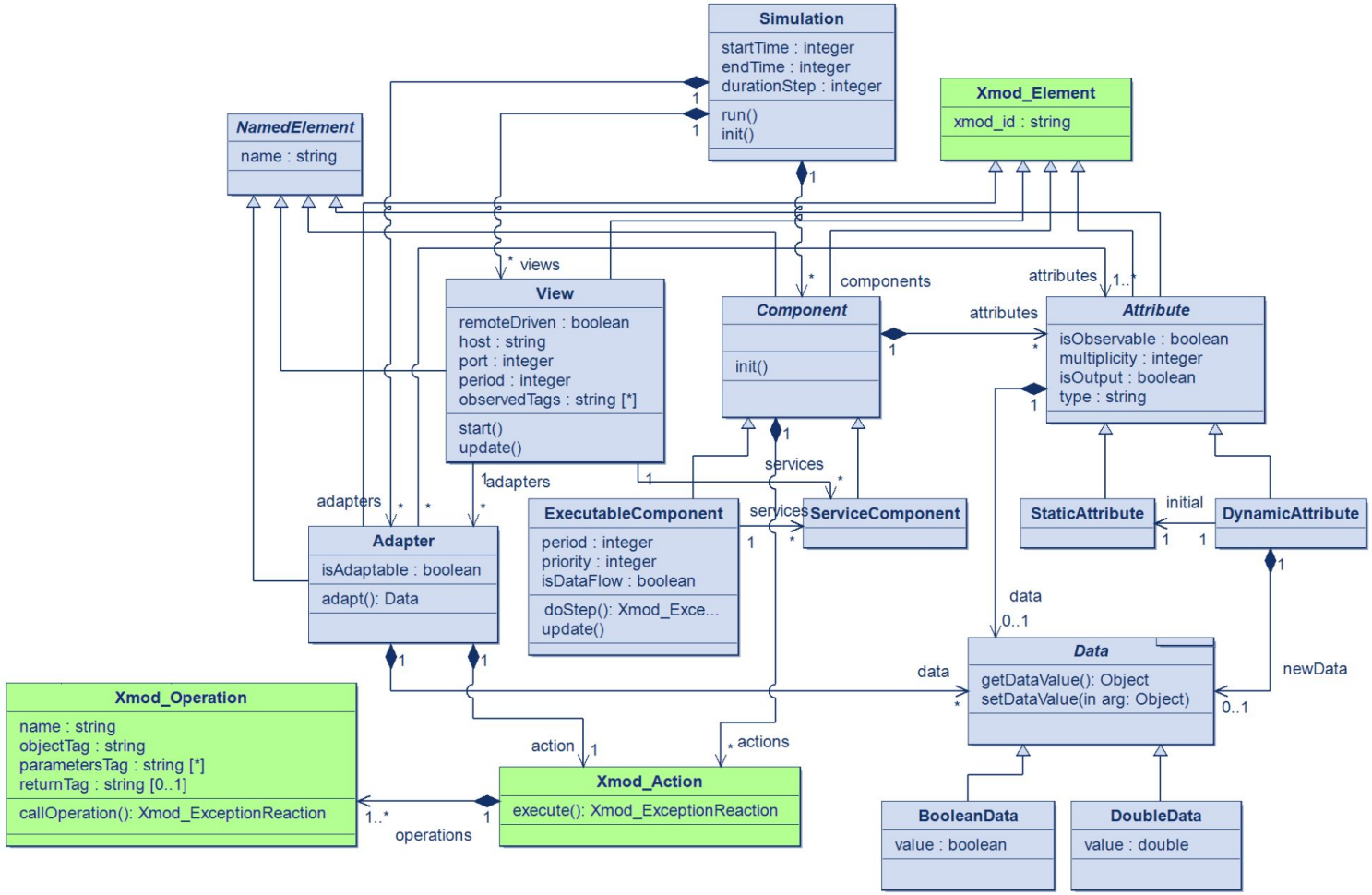
Ingénieur de langage : Définition du méta-modèle



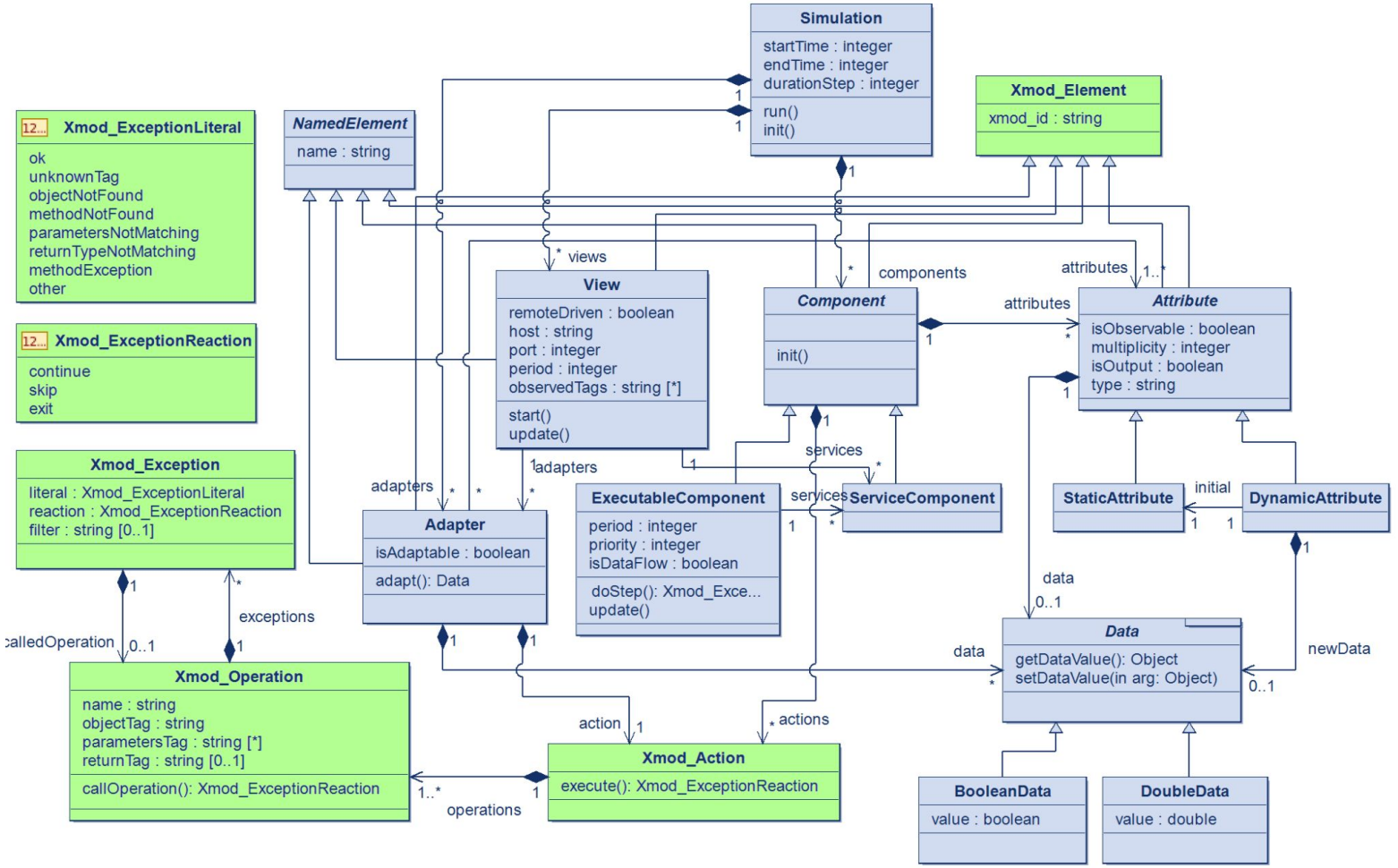
Ingénieur de langage : Définition du méta-modèle



Ingénieur de langage : Définition du méta-modèle

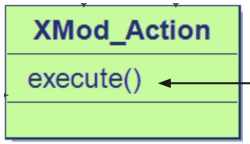


Ingénieur de langage : Définition du méta-modèle



Ingénieur de langage : Développement du moteur d'exécution

- Fonctionnement du logiciel dans son ensemble
- Pas de modification après compilation
- Raccordement avec les méta-classes Xmod




```java
// class Simulation
public void simulate() {
    List<ExecutableComponent> ecs = new ArrayList<>();
    for (Component c : components) {
        if (c instanceof ExecutableComponent) ecs.add((ExecutableComponent) c);
    }
    ecs.sort(Comparator.comparingLong(ExecutableComponent::getPriority));

    for (View v : views) v.start();

    for (long i = startTime; i < endTime; i++) {
        for (ExecutableComponent ec : ecs) {
            if (i % ec.getPeriod() == 0) {
                ec.doStep();
                ec.update();
            }
        }
    }
}

// class ExecutableComponent
public XMod_ExceptionReaction doStep() {
    return action("doStep").execute();
}
```

Ingénieur logiciel : Modélisation

- Création d'un modèle de drone muni d'une **position**, d'un niveau de **batterie** et une **vue**

- Paramétrage de l'appel du code métier

- 3 tags
- Gestion des valeurs de retour
- Gestion des exceptions

- Vérification statique (XText)
- Vérification dynamique

```
Simulation xmod_id:sim (0,10,1000) {
    Executable xmod_id:drone (0,1) {
        static Double[3] positionInitial = (0.0, 3.0, 0.0);
        dynamic observable Double[3] position : positionInitial;

        static Double batteryCapacity = 6.0;
        dynamic Double batteryLevel : batteryCapacity;

        Action doStep {
            call calcNextPos("model:drone.attributes[position].data.value",
                "java:Double[3](2.0,0.0,0.0)", "model:sim.durationStep")
                on "ext:calcs" returns "model:drone.attributes[position].newData.value"
            onError methodException("NoChange") then continue
            onError other call handleException("ext:context.exception")
                on "ext:calcs" then exit;
        }
    }

    View xmod_id:unityView {
        host "localhost"
        port 2223
        period 1000
        observe "drone.position", "drone.batteryLevel";
    }
}
```

Nom de la méthode

Paramètres

Objet d'appel

Retour

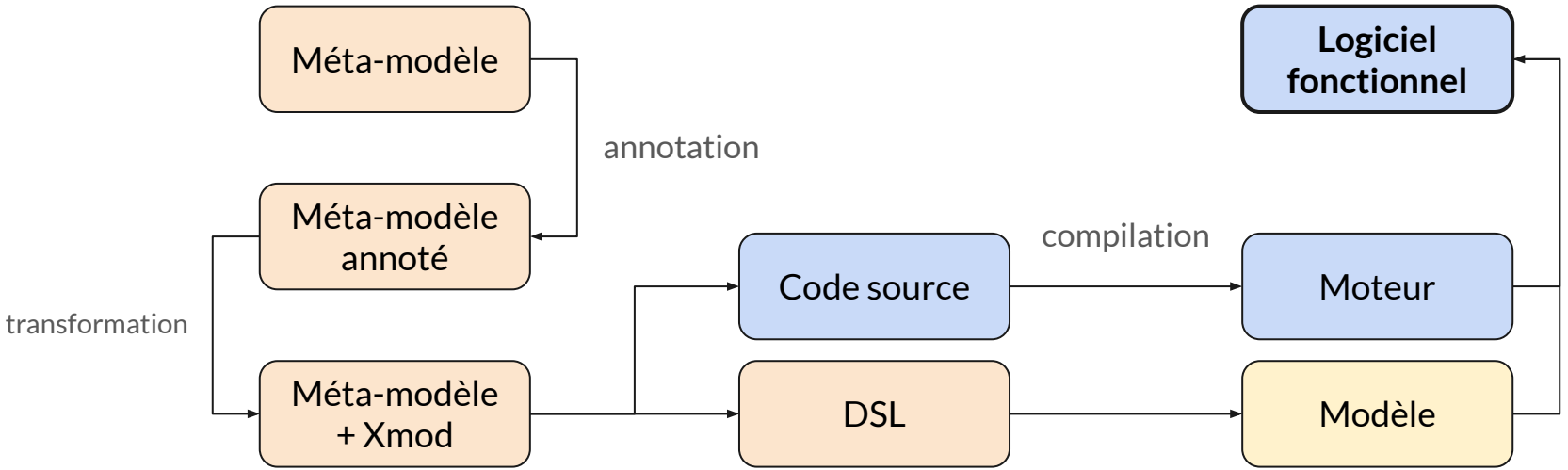
Ingénieur logiciel : Code métier

- Fonctions métier
  - Calculs
  - Trajectoires
  - Comportement

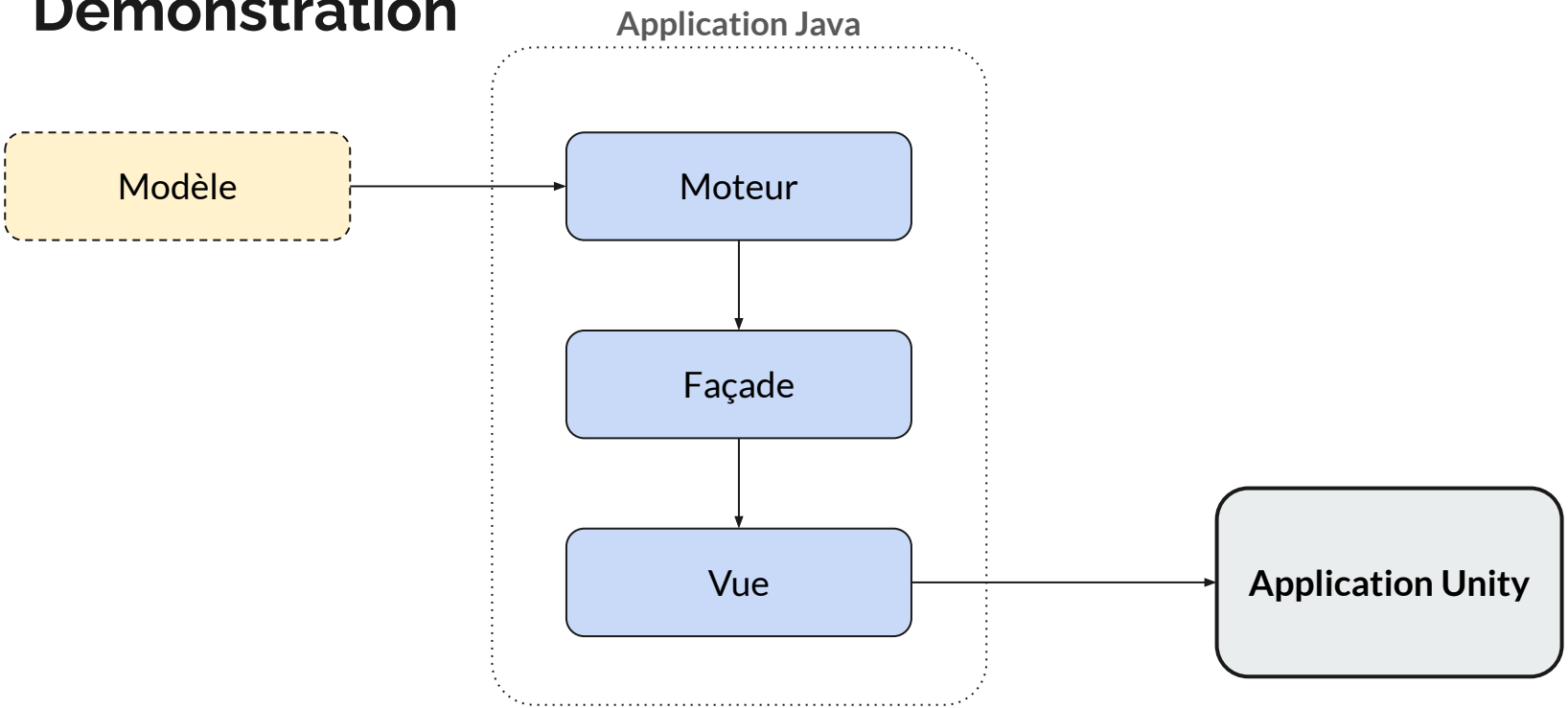
- Ajout de nouveaux algorithmes
- Modification des existants

```java
/**
 * Calculates the next battery level after a given duration.
 *
 * @param level      of battery in MJ
 * @param speed      in m/s
 * @param mass       in kg
 * @param efficiency the engine efficiency (between 0 and 1)
 * @param duration   in second
 * @return the next battery level in MJ
 */
public double calcNextBatLevel(double level, double speed,
                               double mass, double efficiency,
                               long duration) {
    double power = ExtUtils.calculatePower(efficiency, mass, speed);
    double consumedEnergy =
            ExtUtils.energyConsumption(power, (duration / 1000.0))
                    / 1_000_000; // converted in MJ
    return Math.max(level - consumedEnergy, 0);
}
```

Développer avec Xmodeling Studio



Démonstration



Conclusion et perspectives

- Modification facilitée du modèle et du code métier

- Léger : pas de génération de code
- Non intrusif : pas de modification du code existant

- Typage côté modèle

Références

[1] Loïc Salmon, Pierre Yves Pillain, Goulven Guillou, Jean-Philippe Babau: NAVIDRO, a CARES architectural style for configuring drone co-simulation. ACM Trans. Embed. Comput. Syst. 23(3): 39:1-39:26 (2024)

[2] Eric Cariou, Olivier Le Goaër, Léa Brunschwig, Franck Barbier: A generic solution for weaving business code into executable models. EXE Workshop at MoDELS 2018, CEUR Workshop Proceedings, vol. 2245, October 2018

[3] Blochwitz, T. et al. : *The Functional Mockup Interface for Tool-independent Exchange of Simulation Models*. Modelica Conference (2011)

[4] OMG : *Semantics of a Foundational Subset for Executable UML Models* (2011–2021)