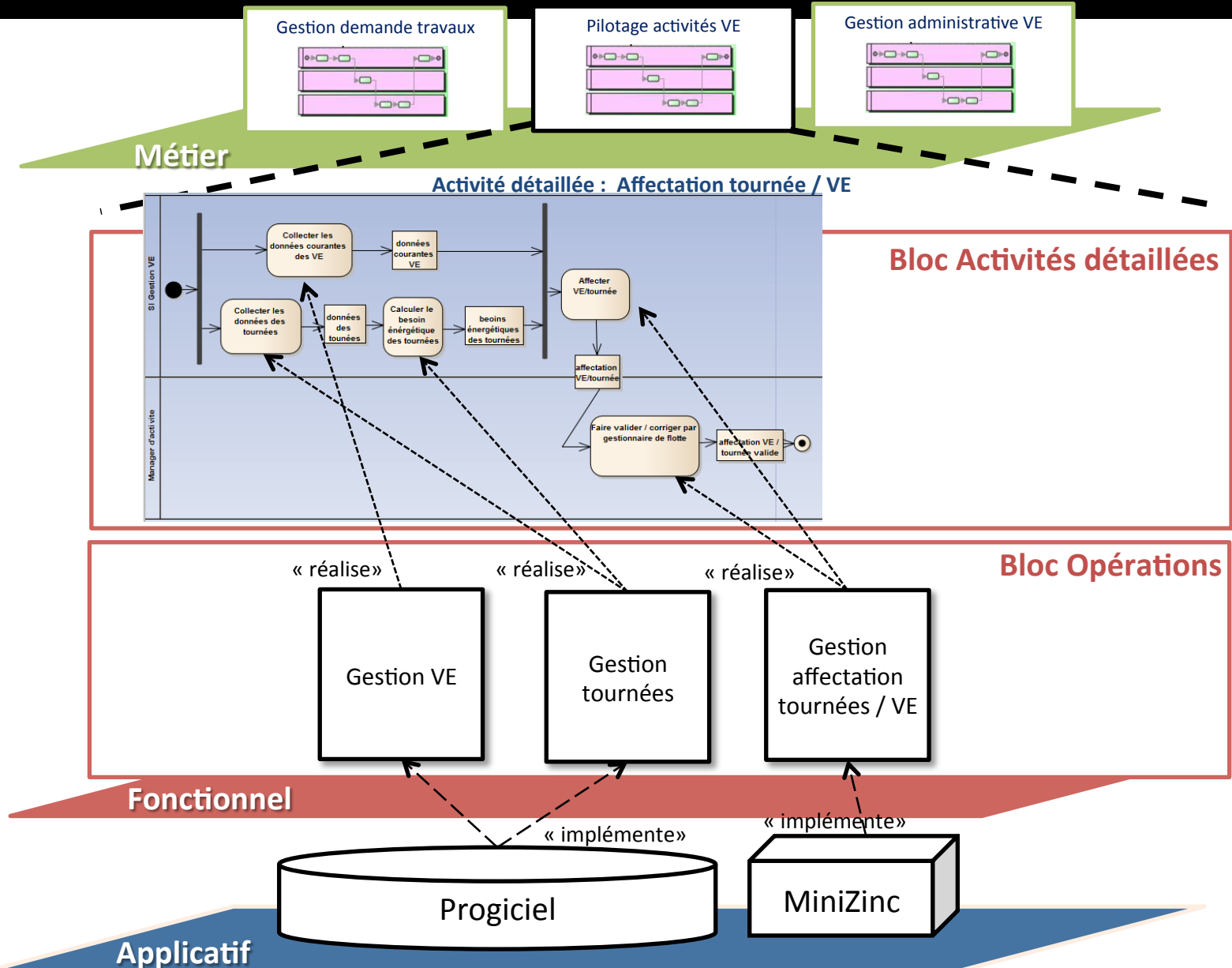
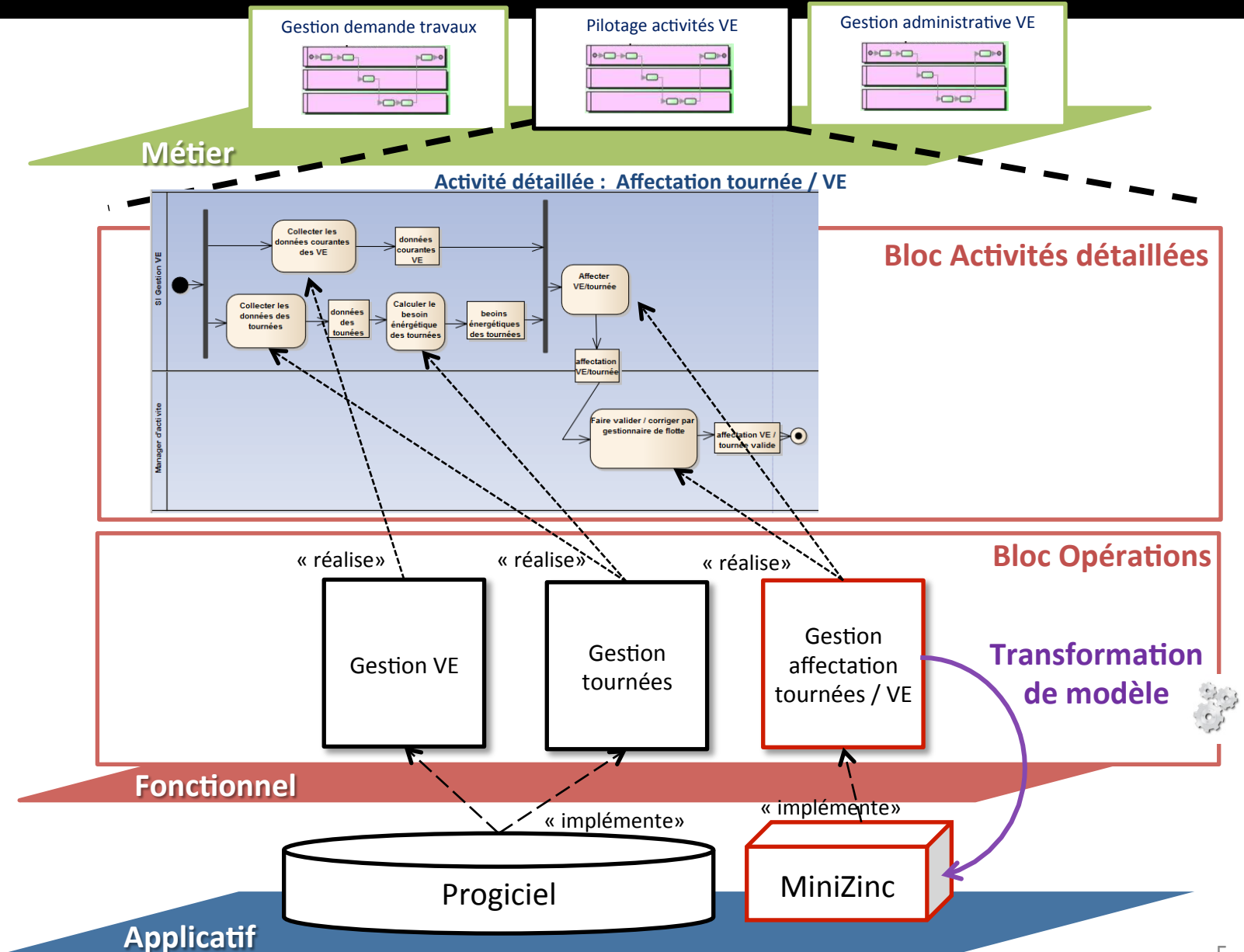


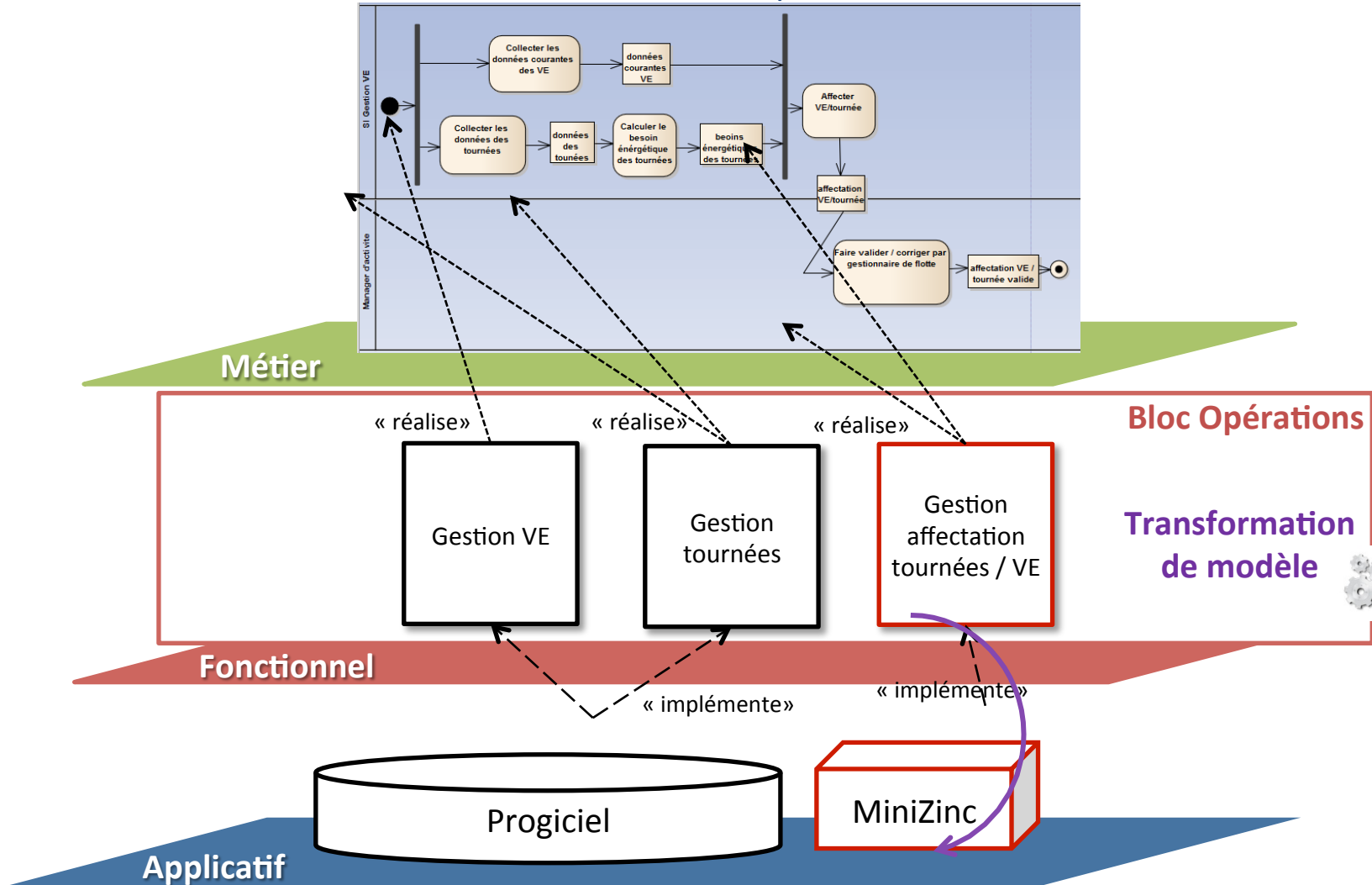
# Mise en œuvre pour le cas métier



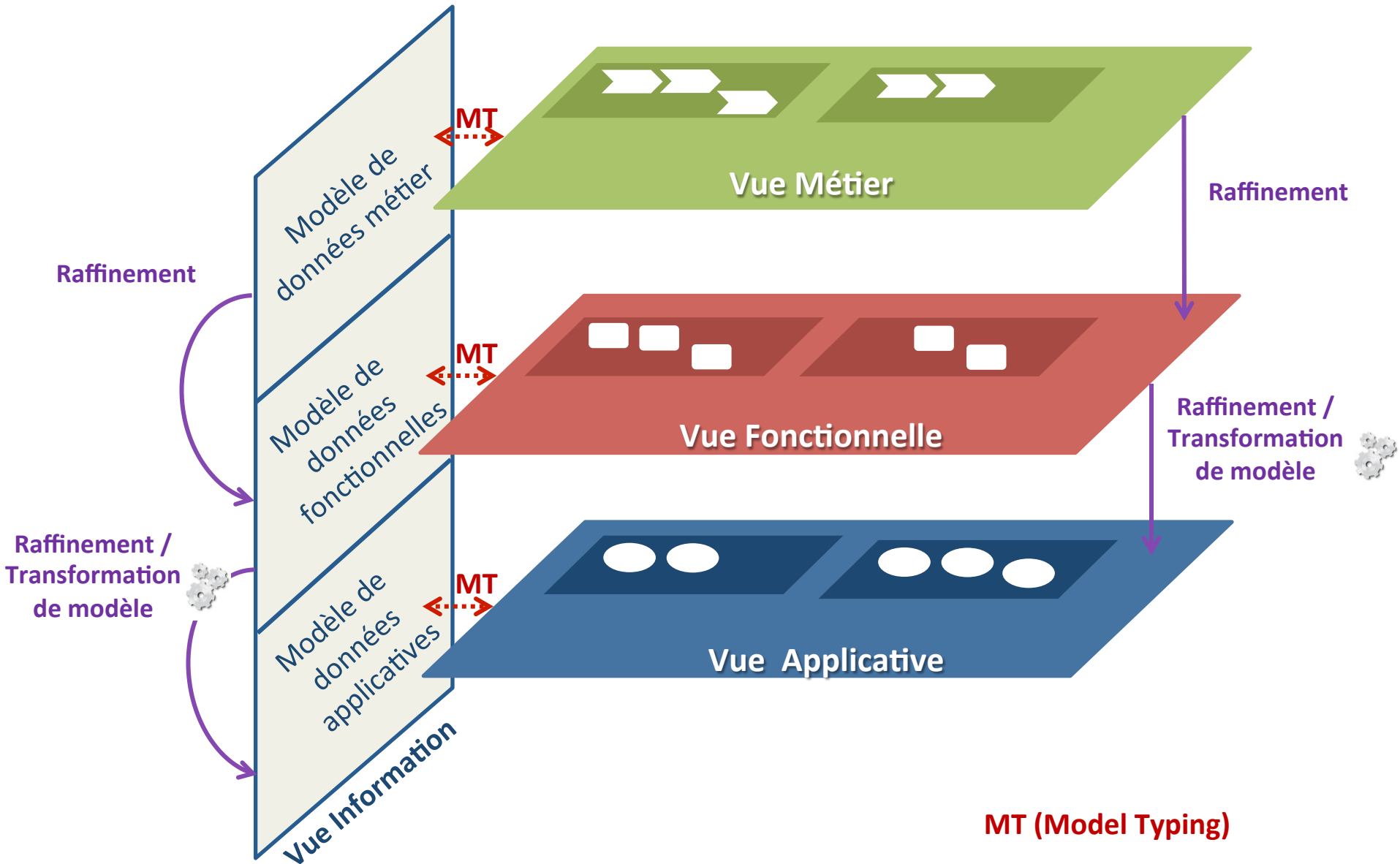
# Mise en œuvre pour le cas métier



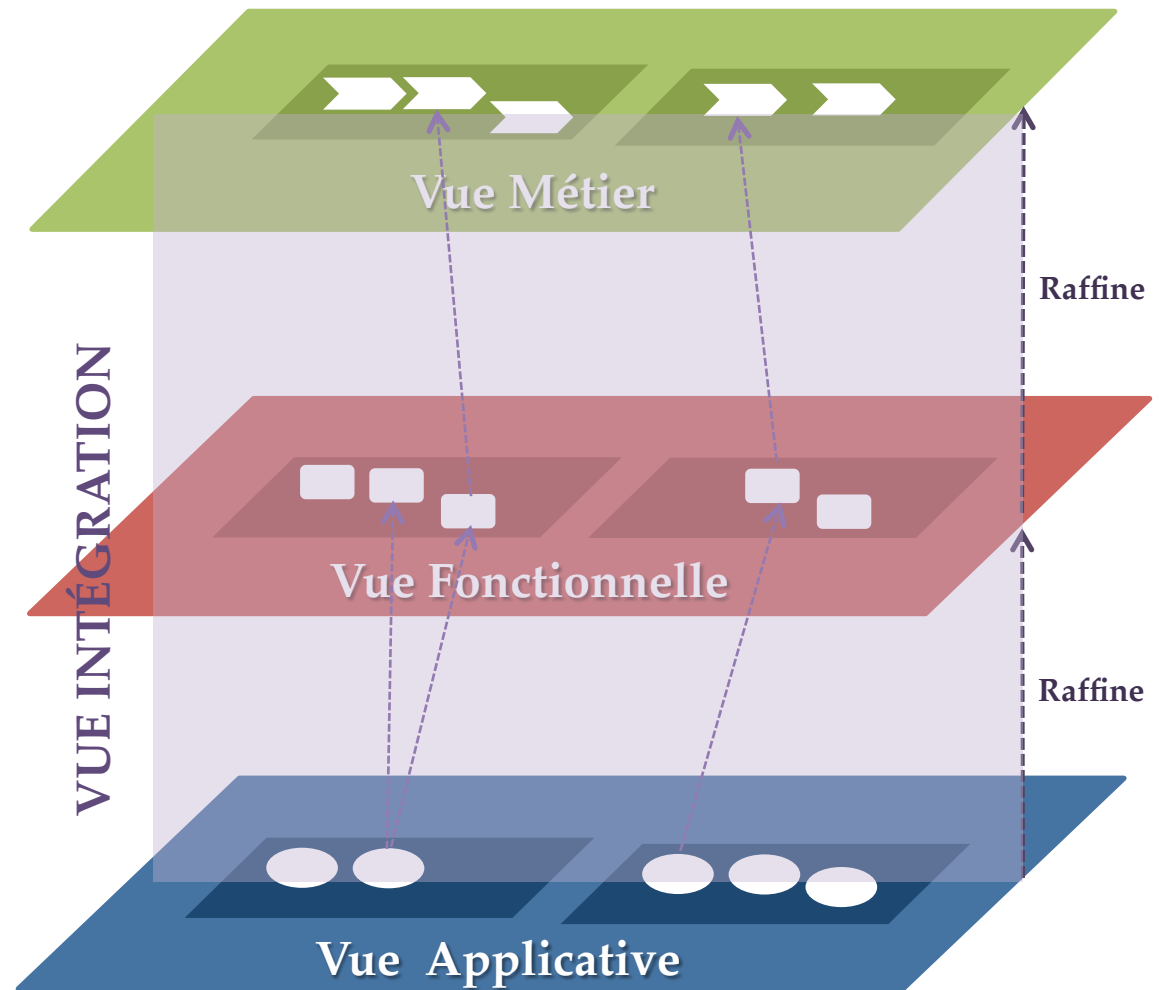
## Activité détaillée : Affectation tournée / VE



# Démarche

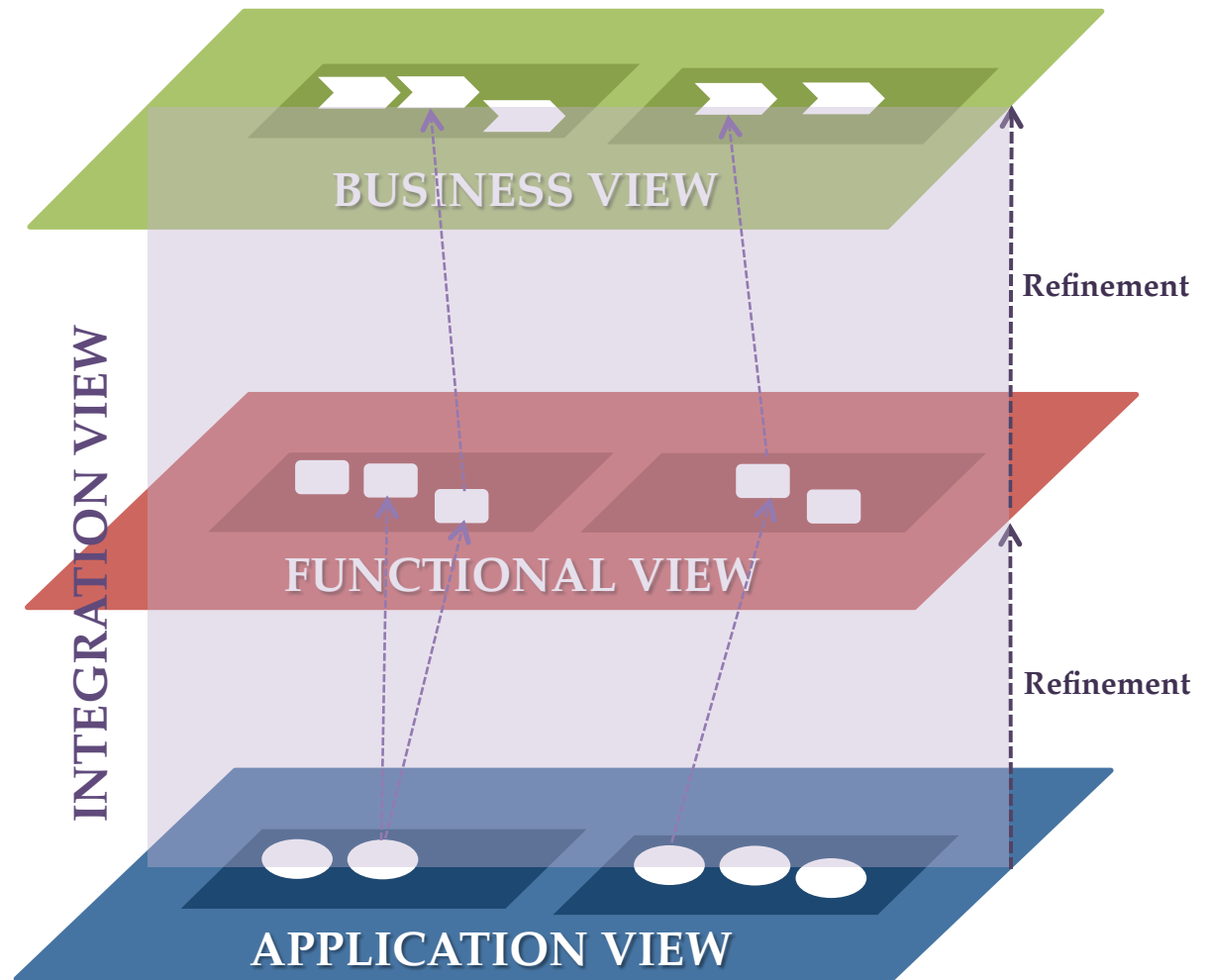


# Démarche





# Démarche



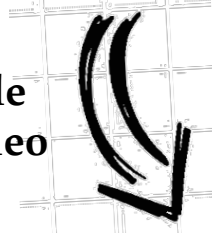
### Gestion affectation

```
context Tournee
inv:    self.VE.autonomie * self.VE.niveauBatterie > self.distanceParcourue

context Tournee
inv:    self.VE <> undefined xor self.VT <> undefined

context Tournee::kmElec(): int
body:   (Tournee::allInstances()->collect(t.VE <> undefined |t.distanceParcourue))->sum()
```

Transformation de  
modèle avec Acceleo



### MiniZinc

```
constraint forall (i in Tournees) (
%affectation de vehicule electrique si l autonomie l autorise
constraint forall(i in Tournees, j in VehiculesElec)
(tourneeVehicule[i]= identifiantVE[j] -> (autonomie[j]*niveauBatterie[j] > distanceTournee[i] /\
kmElec[i]=distanceTournee[i]));

% affectation d un vehicule thermique et dans ce cas kmElec est nul
constraint forall(i in Tournees, k in VehiculesTherm)
(tourneeVehicule[i]= identifiantVT[k] -> kmElec[i]=0);

%maximiser le nombre de km de tounrnee fait par les vehicules electriques
solve maximize sum(i in Tournees) (kmElec[i]);
```

## Functional view

### Allocation function

```
context Tour
inv : self.EV.autonomy * self.EV.batteryLevel > self.neededEnergy

context Tour
inv : self.EV <> undefined xor self.CV <> undefined

context Tour :: elecKm() : int
body : (Tours::allInstances() -> collect(t.EV <> undefined | t.distance)) ->
sum()
```

## Model transformation with Acceleo



## Application view

### MiniZinc Module

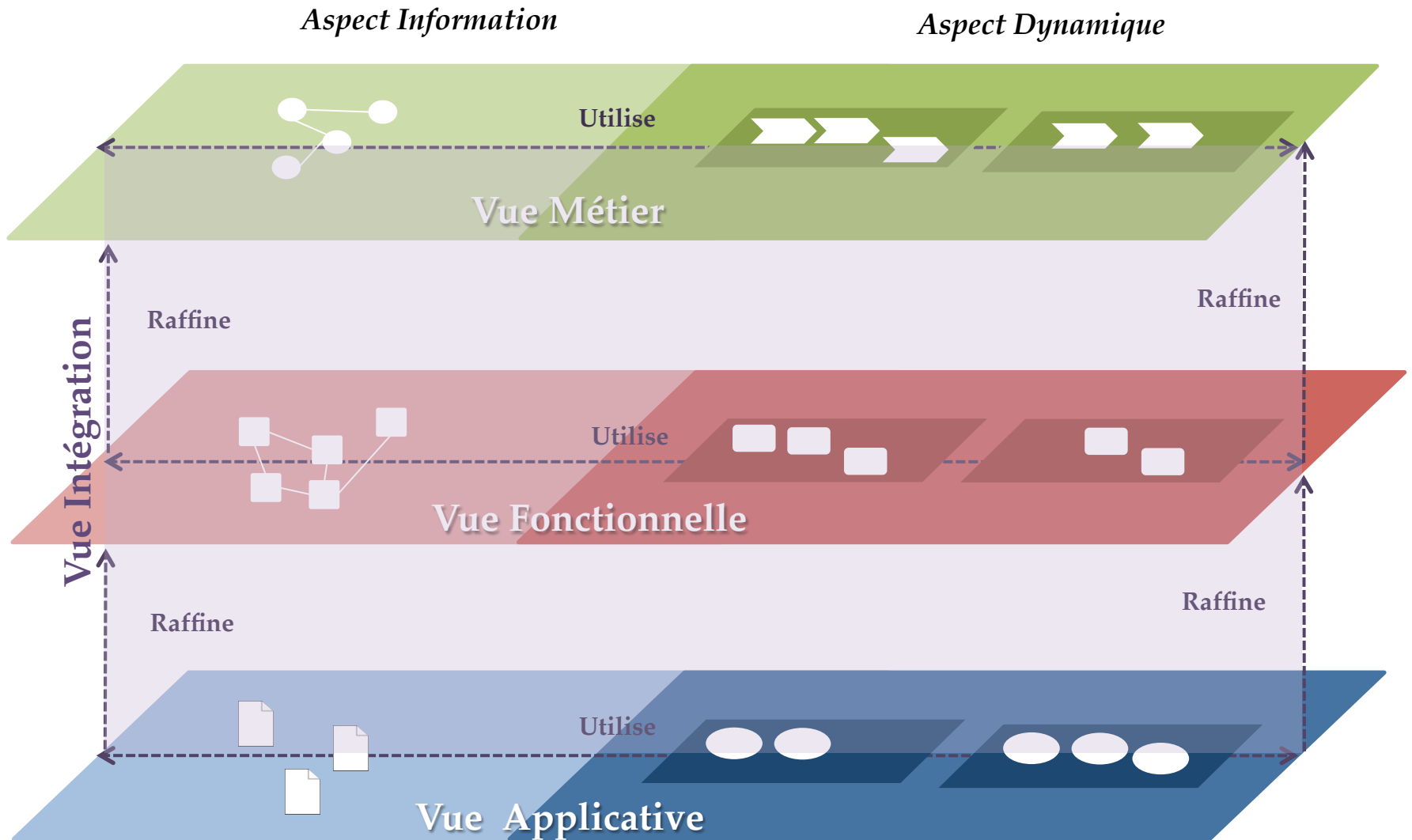
```
%a vehicle is affected to only one tour
constraint alldifferent(allocation);

%allocate an electric vehicle if the autonomy permits it
constraint forall(i in Tours, j in ElectricVehicles)
(allocation[i]= id_EV[j] -> (autonomy[j] > neededEnergy_Tours[i]
/\ kmElec[i]= distance_Tours[i]));

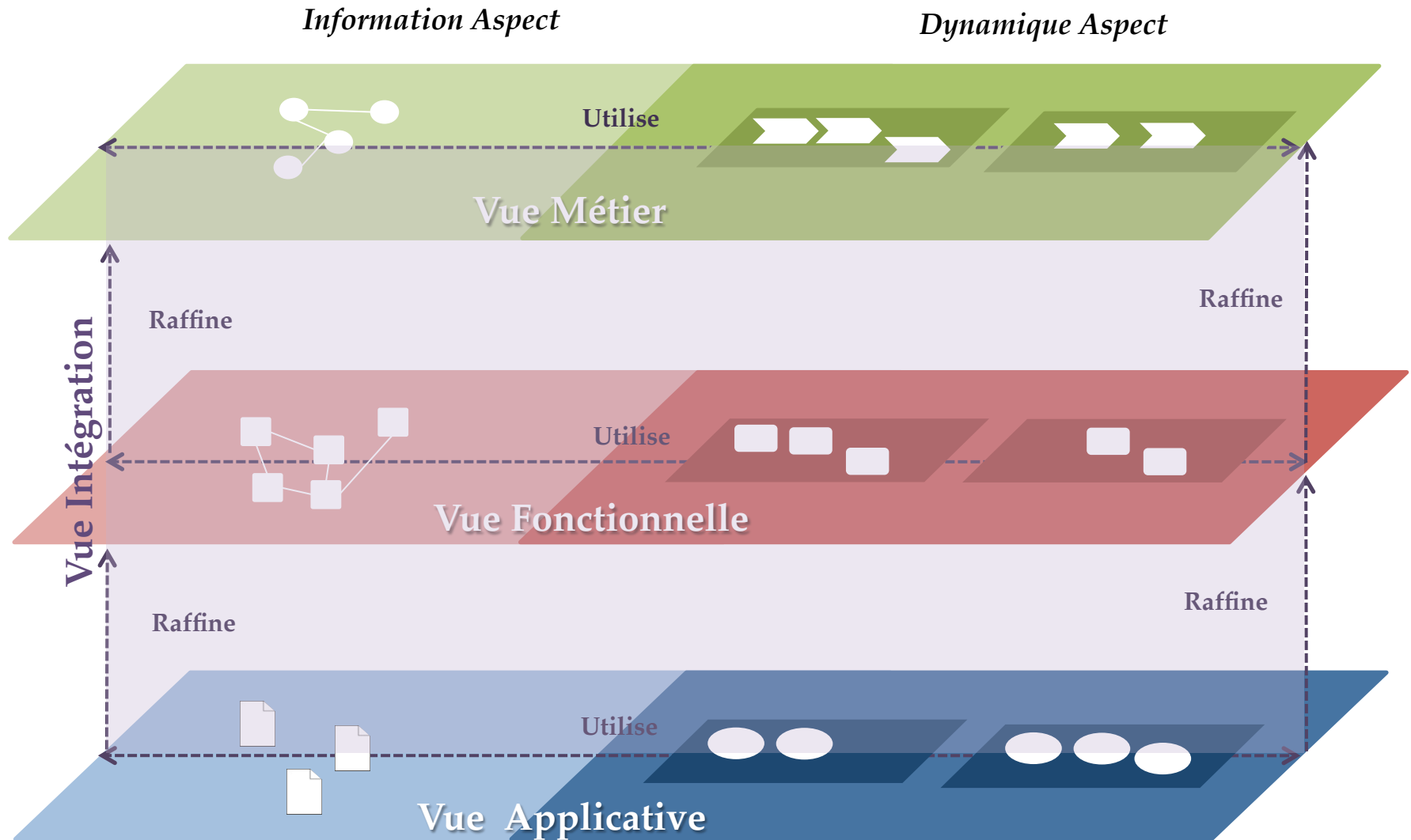
% allocate a combustion vehicle
constraint forall(i in Tours, k in CombustionVehicles)
(allocation[i]= id_CV[k] -> kmElec[i]=0);

%maximize km done by electric vehicles
solve maximize sum(i in Tours)(kmElec[i]);'
```

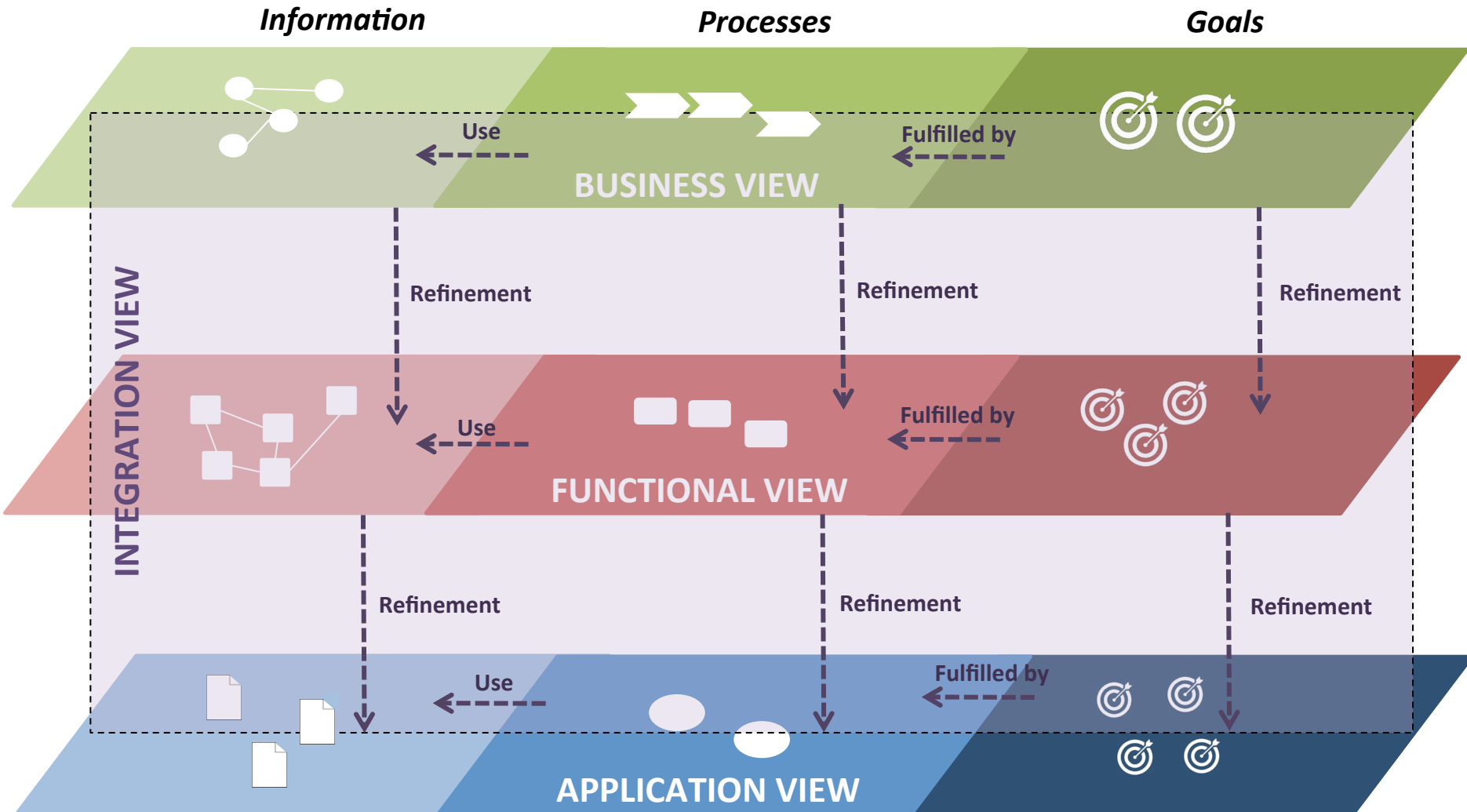
# Démarche



# Approach



# Démarche



# Démarche

