Contribution à la prise en compte des plates-formes logicielles d'exécution dans une ingénierie générative dirigée par les modèles

Frédéric Thomas

Directeur : Pr. François Terrier [1]
Encadrant : Dr. Jérôme Delatour [2]
Dr. Sébastien Gérard [1]

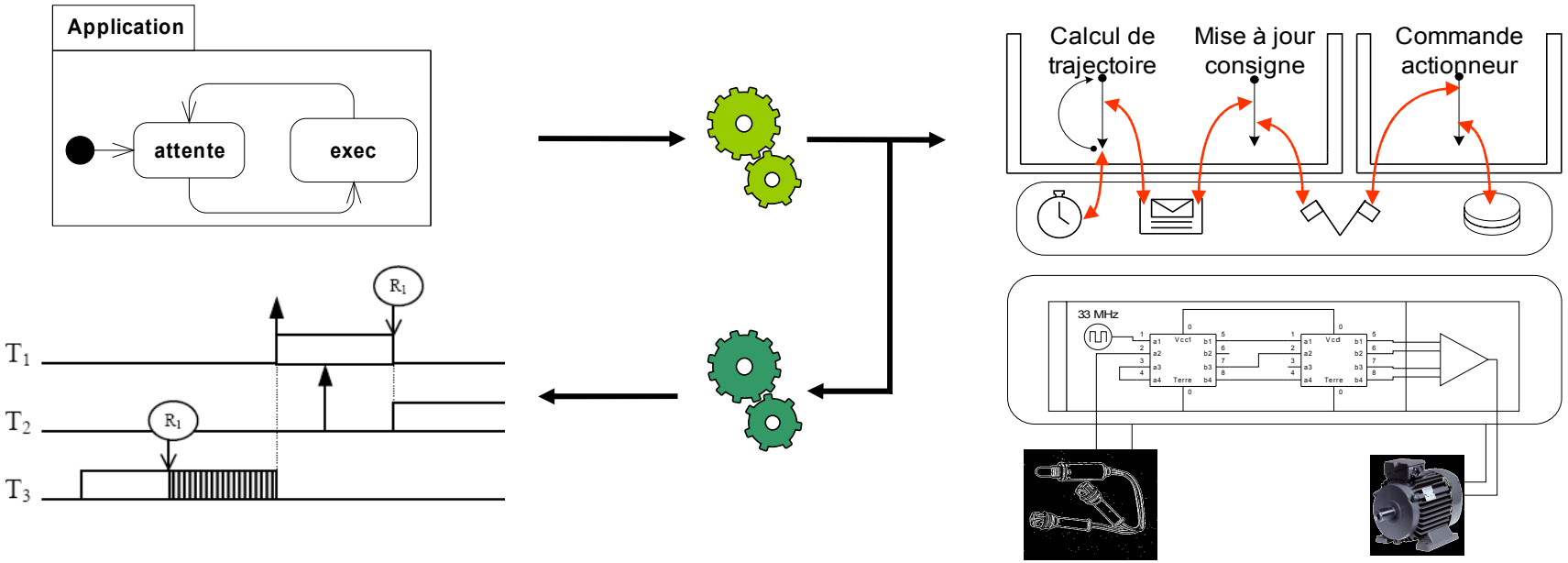
[1] CEA LIST / LISE
[2] ESEO / Trame

21 Novembre 2008

Un contexte dual

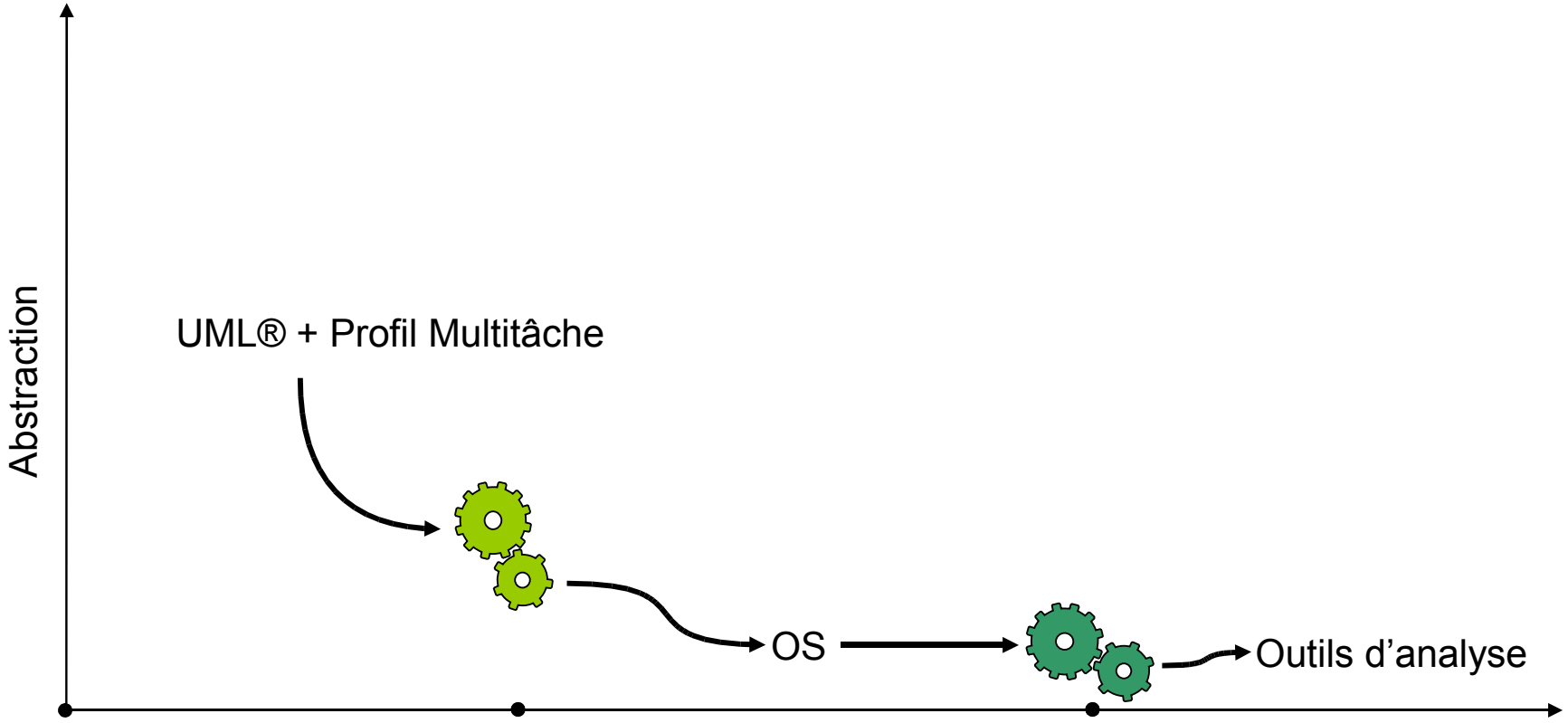
- Contexte applicatif : Applications concurrentes (multitâches) embarquées
  - Faciliter la conception des applications
  - Gérer l'hétérogénéité des plates-formes d'exécution

- Contexte technologique : Ingénierie dirigée par les modèles
  - Faciliter l'implantation et la maintenance
  - Faciliter l'intégration des outils et des techniques de développement



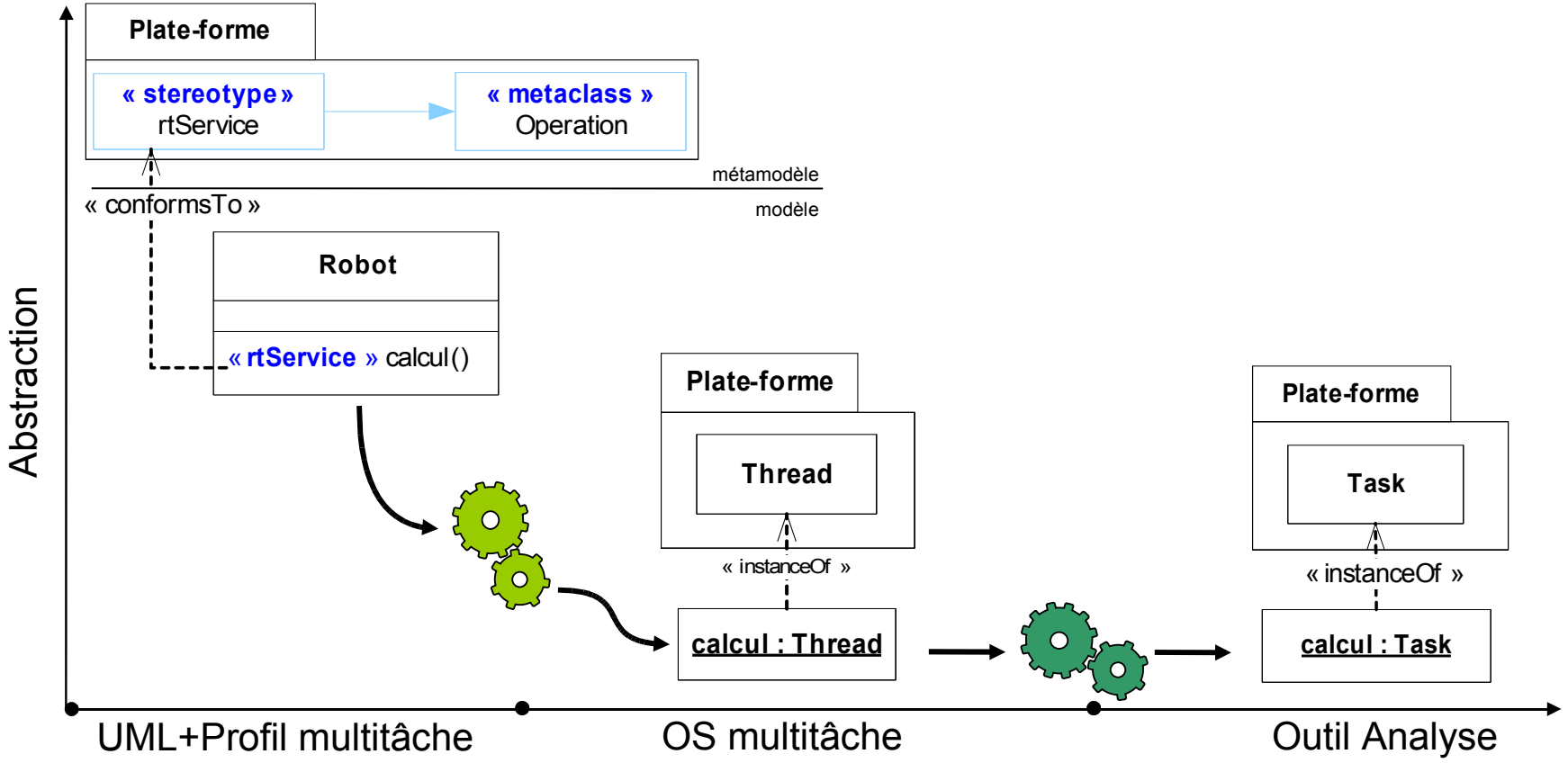
Une ingénierie dirigée par les plates-formes

- Prise en compte du caractère concurrent (multitâche) des systèmes
  - A différent niveau d'abstraction
  - Description implicite (déclarative) et/ou explicite (impérative)



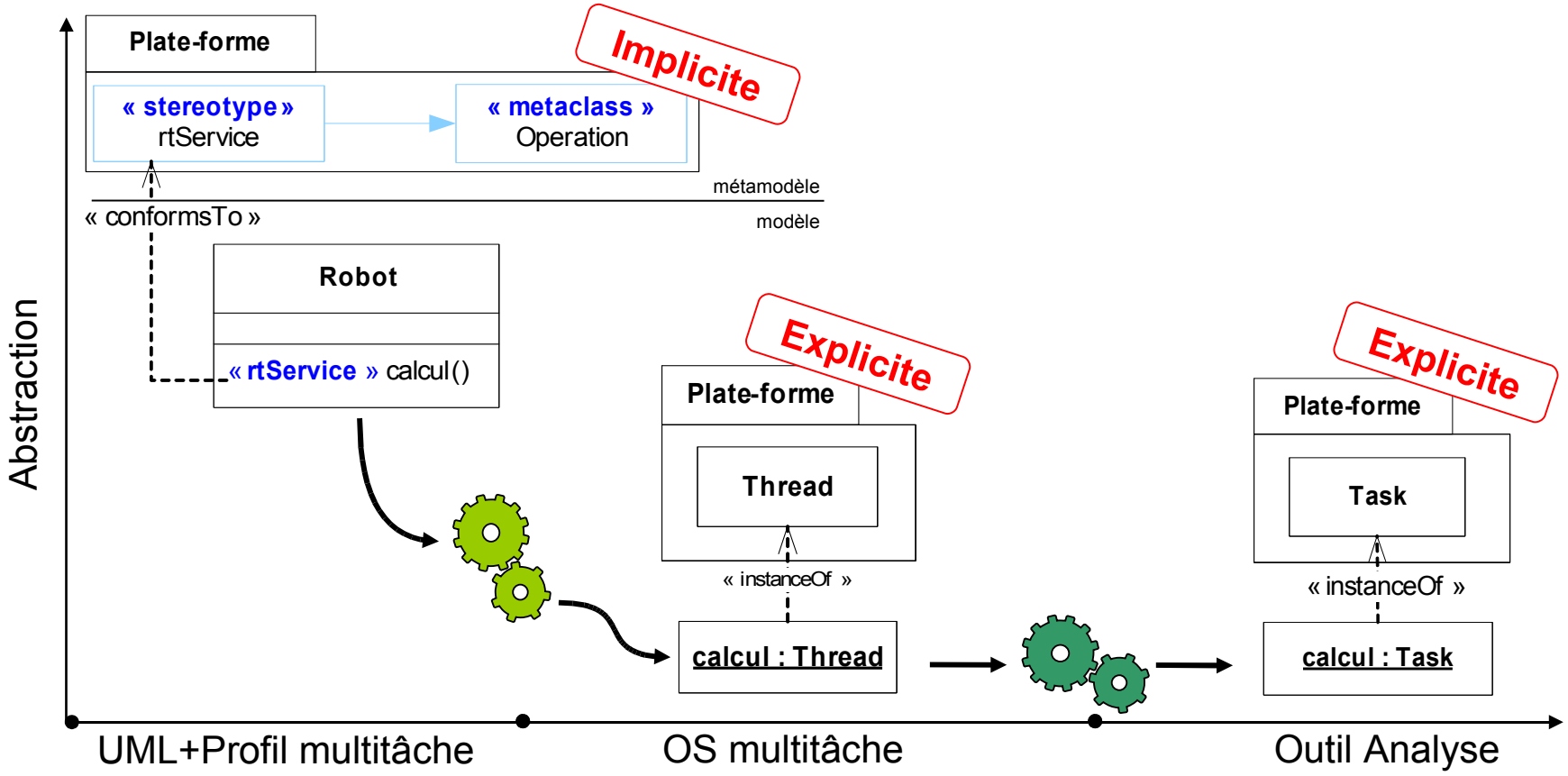
Une ingénierie dirigée par les plates-formes

- Prise en compte du caractère concurrent (multitâche) des systèmes
  - A différent niveau d'abstraction
  - Description implicite (déclarative) et/ou explicite (impérative)



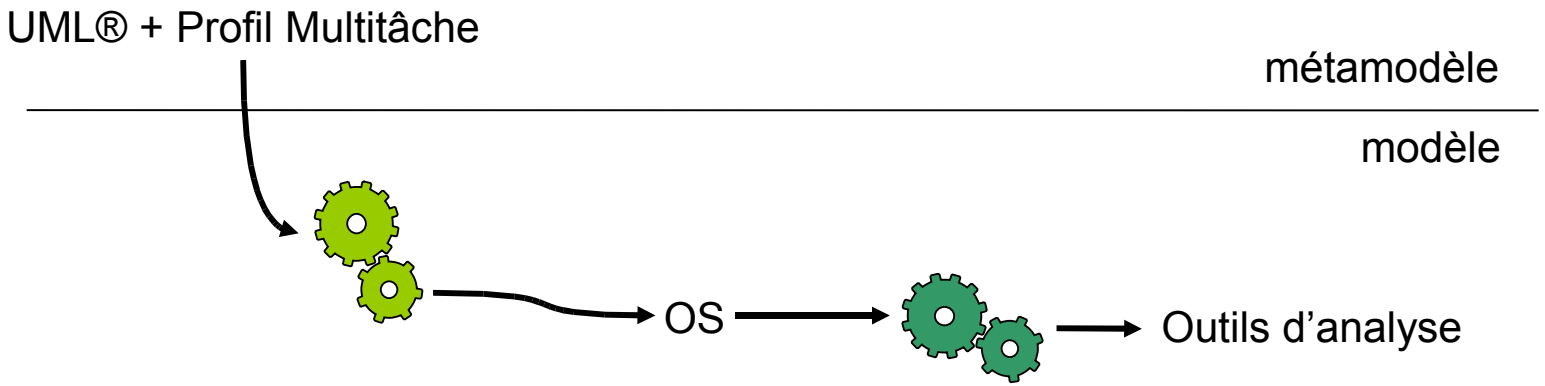
Une ingénierie dirigée par les plates-formes

- Prise en compte du caractère concurrent (multitâche) des systèmes
  - A différent niveau d'abstraction
  - Description implicite (déclarative) et/ou explicite (impérative)



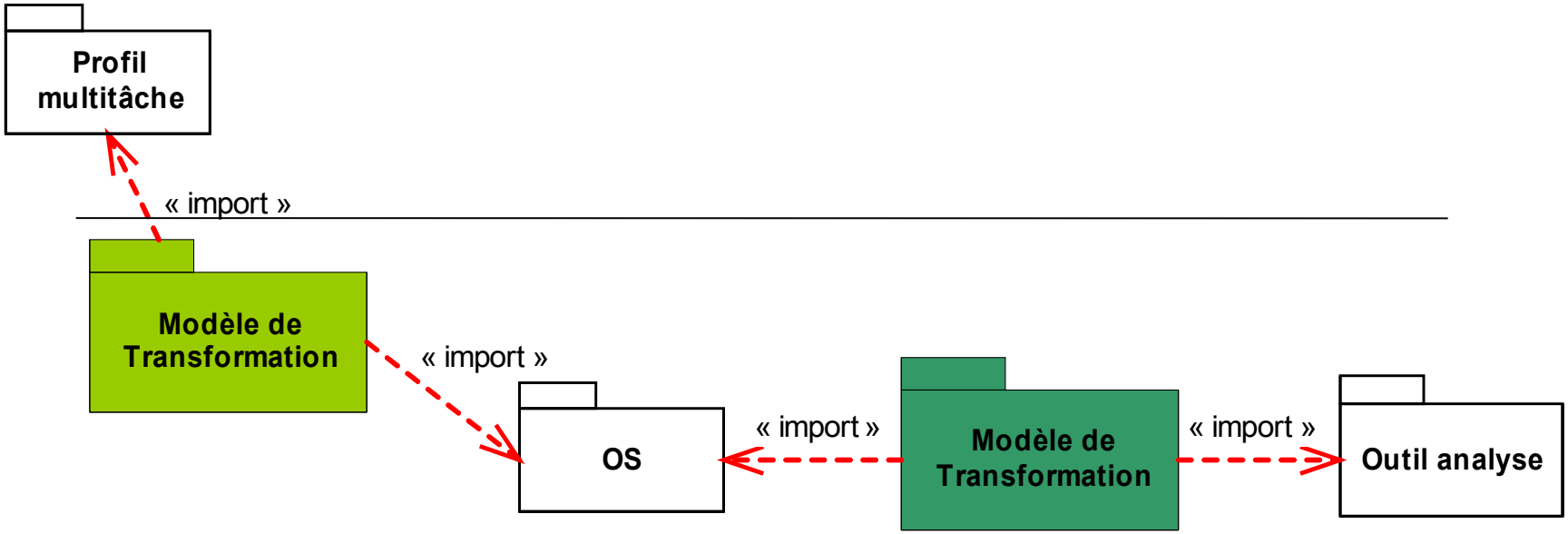
Des transformations de portage

- Portage « linéaire »
  - explicite -> explicite
  - implicite -> implicite
- Portage « transverse »
  - Transverse aux niveaux de modélisation (modèle, métamodèle)
  - implicite <-> explicite



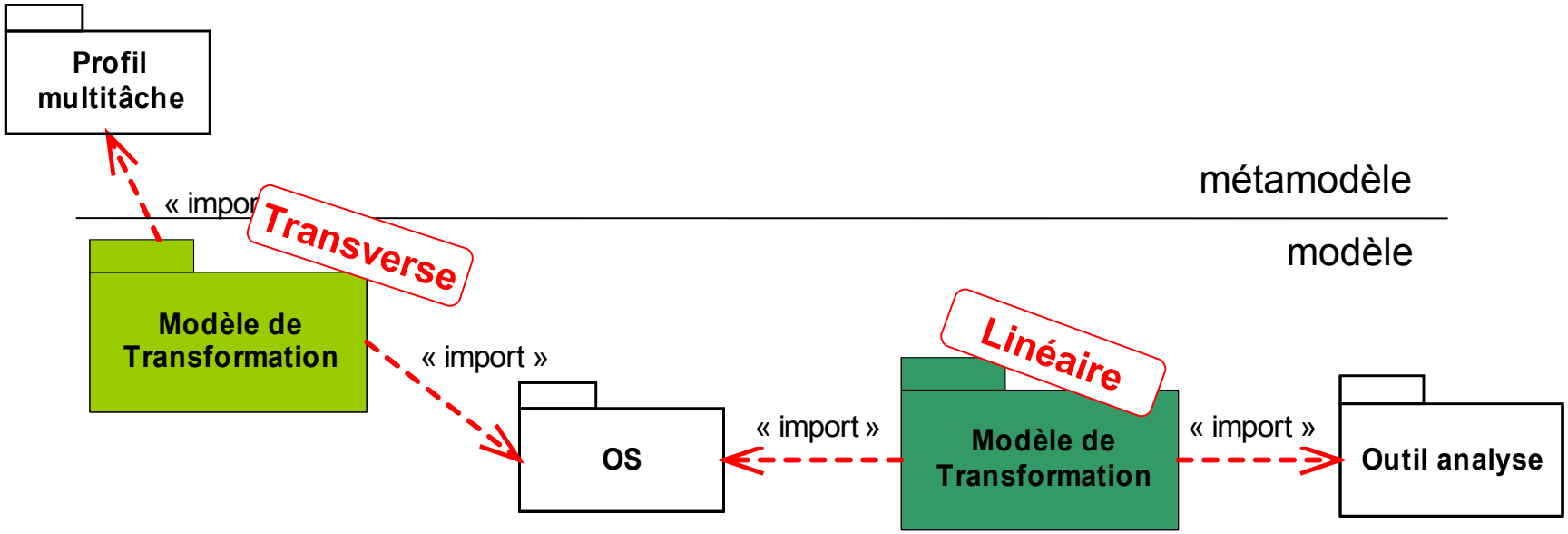
Des transformations de portage

- Portage « linéaire »
  - explicite -> explicite
  - implicite -> implicite
- Portage « transverse »
  - Transverse aux niveaux de modélisation (modèle, métamodèle)
  - implicite <-> explicite



Des transformations de portage

- Portage « linéaire »
  - explicite -> explicite
  - implicite -> implicite
- Portage « transverse »
  - Transverse aux niveaux de modélisation (modèle, métamodèle)
  - implicite <-> explicite



Problématique : spécificité des générateurs

- Import des plates-formes dans les descriptions des transformations
  - Amalgame des préoccupations dans les générateurs
    - Préoccupations liées aux formalises et technologies de l'IDM
    - Préoccupations liées au multitâches (mise en œuvre des mécanismes)

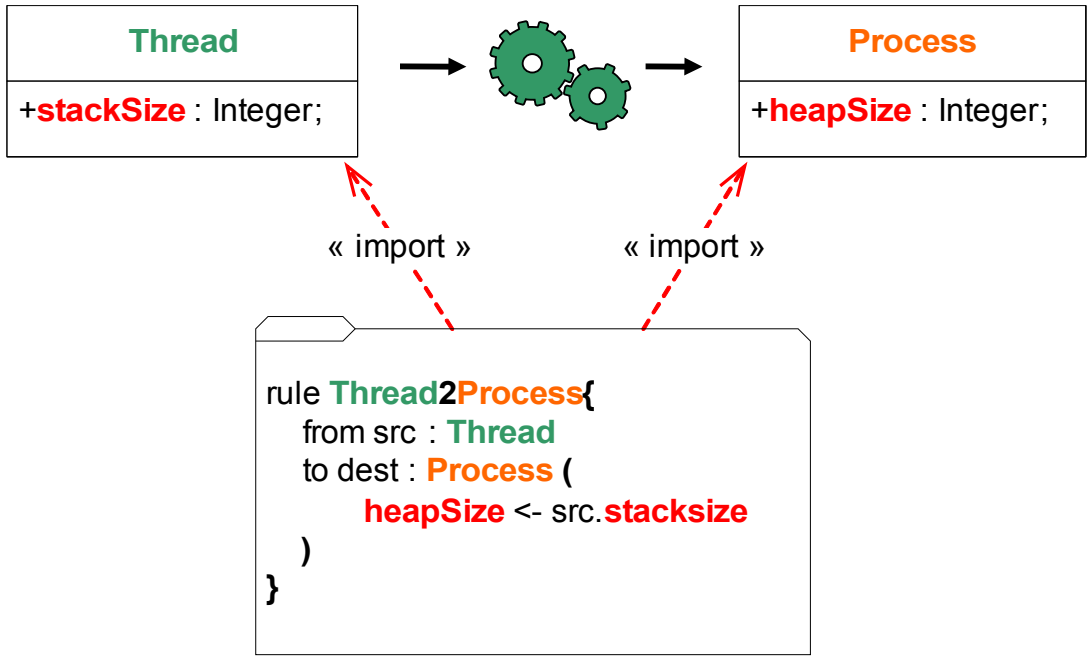
- Générateurs difficiles à concevoir, à maintenir et à faire évoluer
  - Nécessite une double compétence

- Générateurs peu flexibles
  - Les générateurs doivent s'adapter aux contraintes de développement

- Générateurs spécifiques à un contexte de développement donné
  - Les systèmes produits sont évolutifs

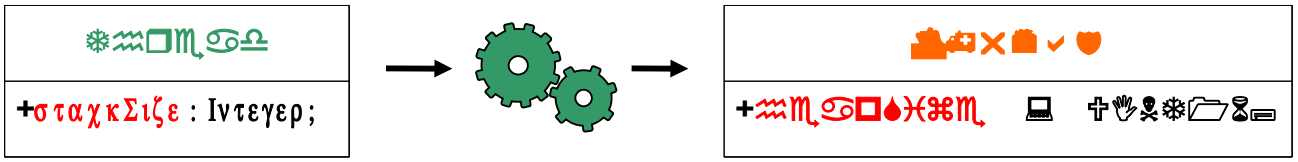
Besoins : Des générateurs évolués

- Conception des générateurs actuels



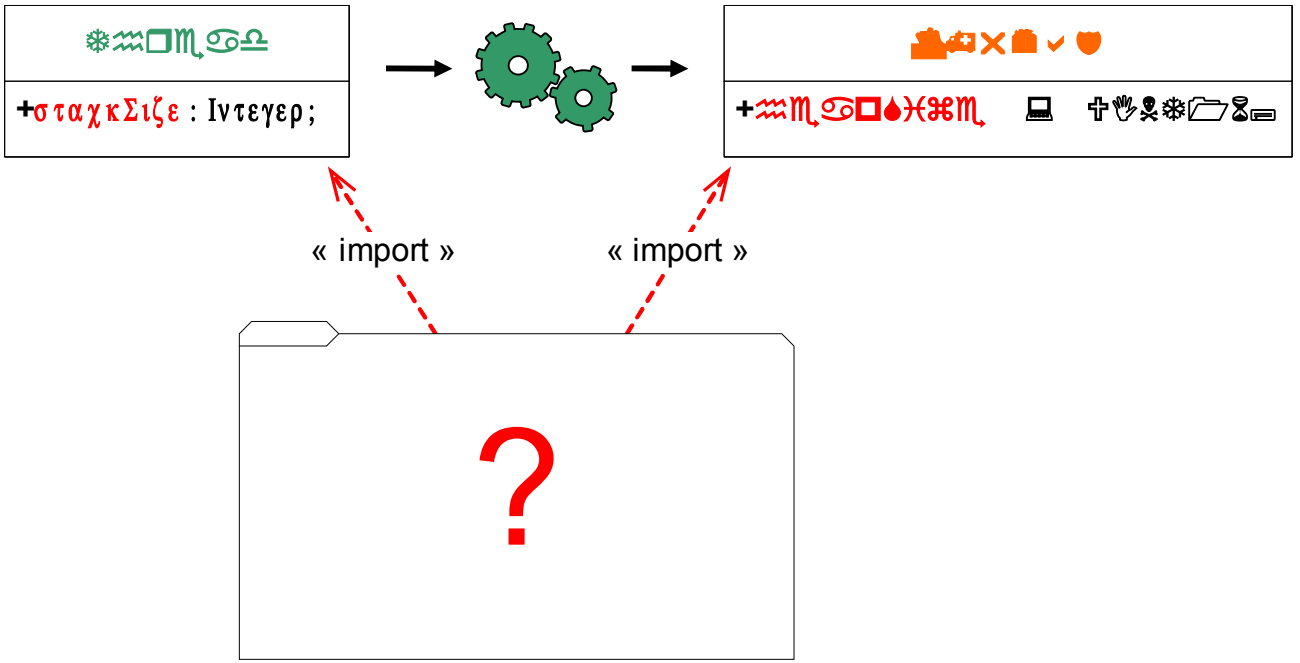
Besoins : Des générateurs évolués

- Conception des générateurs actuels
  - Dirigée par la connaissance de la source et de la cible



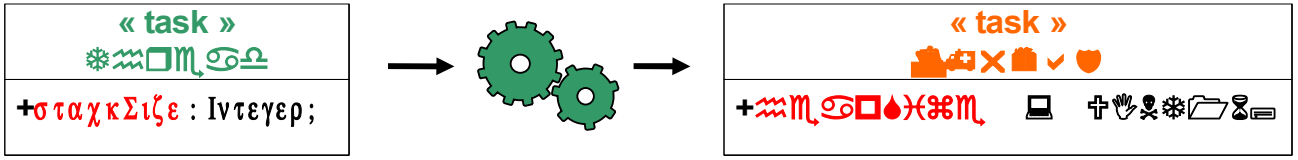
Besoins : Des générateurs évolués

- Conception des générateurs actuels
  - Dirigée par la connaissance de la source et de la cible



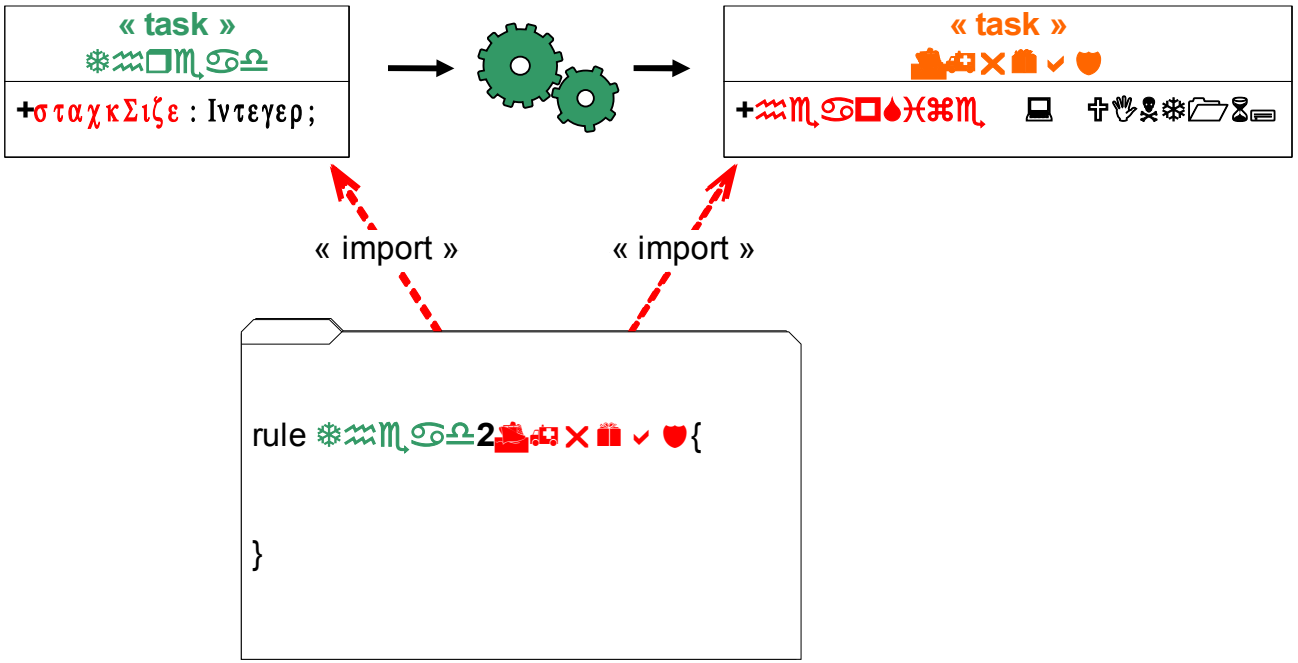
Besoins : Des générateurs évolués

- Conception des générateurs actuels
  - Dirigée par la connaissance de la source et de la cible
- Générateurs proposés
  - Base de connaissance commune pour décrire les plates-formes



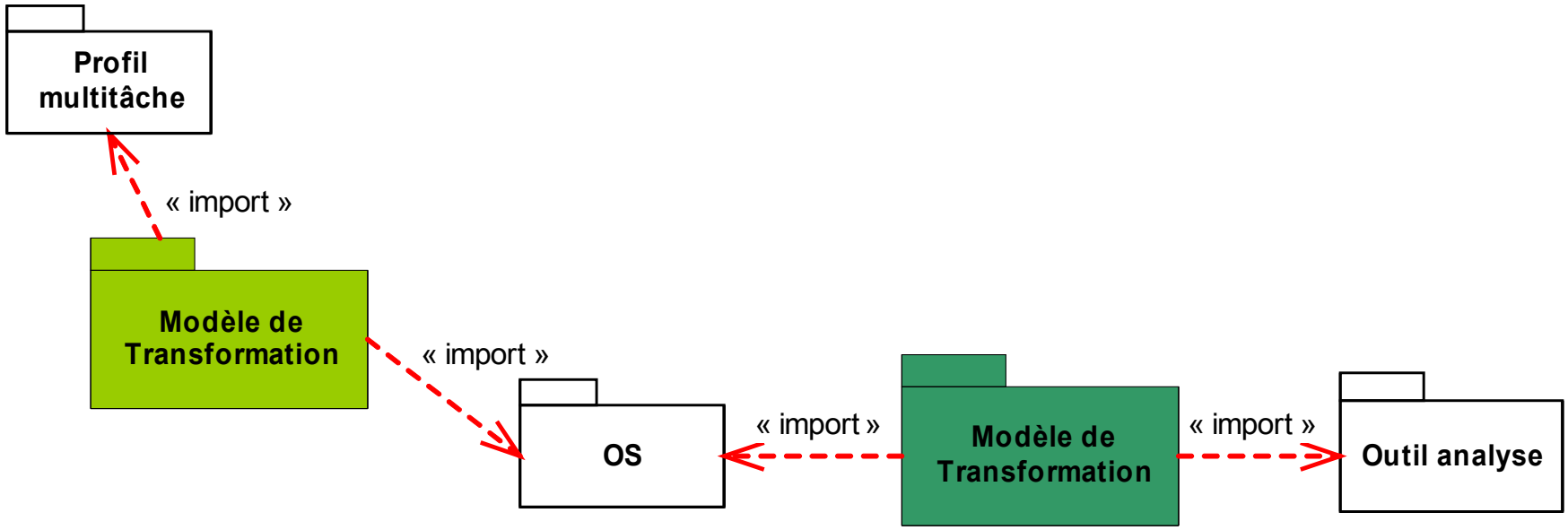
Besoins : Des générateurs évolués

- Conception des générateurs actuels
  - Dirigée par la connaissance de la source et de la cible
- Générateurs proposés
  - Base de connaissance commune pour décrire les plates-formes
  - Générateurs dirigés par les plates-formes



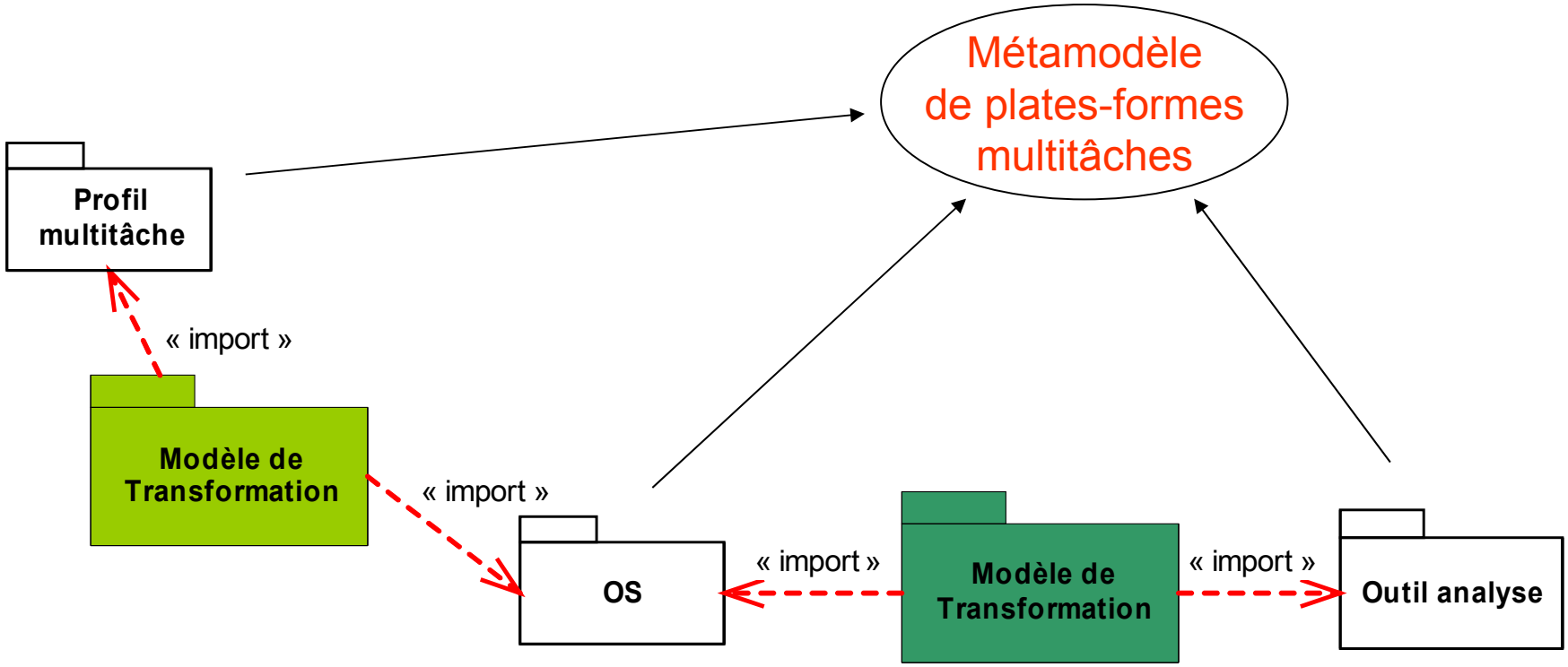
Approche : Expliciter les plates-formes

- Expliciter des modèles de plates-formes
  - Capitaliser les transformations
  - Réutiliser les transformations
  - Aider à la conception des transformations



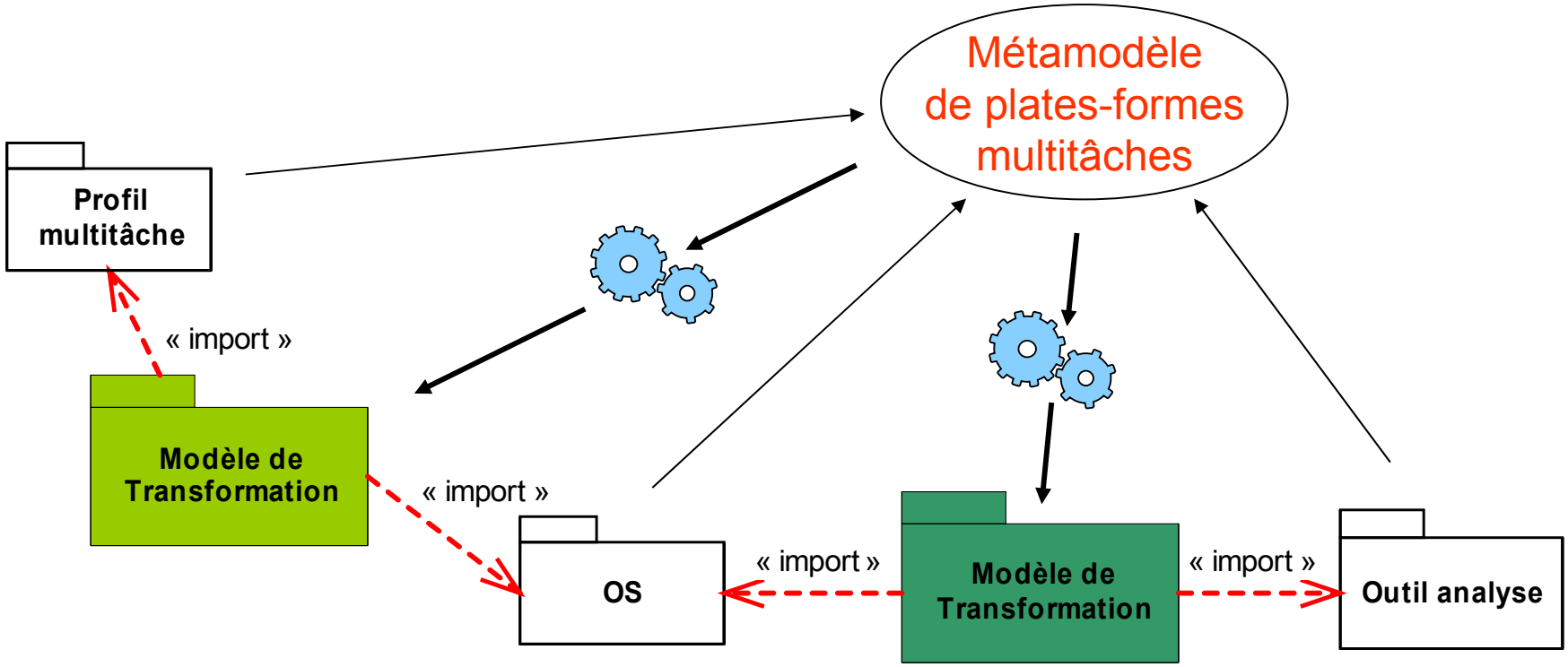
Approche : Expliciter les plates-formes

- Expliciter des modèles de plates-formes
  - Capitaliser les transformations
  - Réutiliser les transformations
  - Aider à la conception des transformations



Approche : Expliciter les plates-formes

- Expliciter des modèles de plates-formes
  - Capitaliser les transformations
  - Réutiliser les transformations
  - Aider à la conception des transformations



Problèmes de recherche

- Qu'est-ce qu'un modèle de plates-formes ?
- Qu'est-ce qu'un métamodèle de plates-formes ?
- Comment capitaliser les transformations de portage ?

Plan de la démarche

- **Qu'est-ce qu'un modèle de plates-formes ?**
  - État de l'art et définition
- **Qu'est-ce qu'un métamodèle de plates-formes ?**
  - Introduction du motif « Resource-Service »
  - Spécification et réalisation d'un outillage support
- **Comment capitaliser les transformations de portage ?**
  - Définition d'une infrastructure de transformation
- **Evaluation**
  - Définition du métamodèle Software Resource Modeling
  - Réalisation d'une infrastructure de transformation
- **Conclusion**

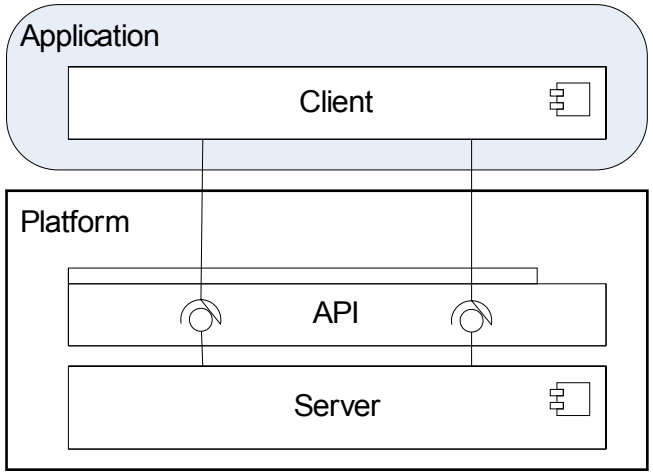
Plan de la démarche

- **Qu'est-ce qu'un modèle de plates-formes ?**
  - Etat de l'art et définition
- **Qu'est-ce qu'un métamodèle de plates-formes ?**
  - Introduction du motif « Resource-Service »
  - Spécification et réalisation d'un outillage support
- **Comment capitaliser les transformations de portage ?**
  - Définition d'une infrastructure de transformation
- **Evaluation**
  - Définition du métamodèle Software Resource Modeling
  - Réalisation d'une infrastructure de transformation
- **Conclusion**

État de l'art : Plate-forme d'exécution

- Définitions existantes
  - Un ensemble de systèmes qui fournissent des fonctionnalités au travers d'une interface que chaque application peut utiliser sans être concernée par les détails d'implantation [MDA, OMG]
  - Une plate-forme spécifie quand et comment sont réalisés les traitements de l'application [Marvie et al.]
  - Un serveur qui répond au besoin de l'application [Kühne et al.]
  - Une couche d'abstraction proposant une API [Vincentelli et al.]
  - Abstraite (fictive) ou concrète (Opérationnelle) [Almeida]

- Réification proposée
  - Des systèmes implicites (niveau méta)
  - Des systèmes explicites (niveau modèle)



Etat de l'art : Approches de modélisation

- SPT [OMG]

+ Profil UML

+ Notion de « Ressource » et de « Service »

- Proposition d'une nouvelle plate-forme

- PML [Szemethy] et Metropolis [Vicentelli et al.],

+ Langage spécifique

+ Artefacts pour modéliser des plates-formes existantes

- Concepts très abstraits (composant, port, …) pour envisager l'inférence des règles de transformation

- Besoins :
  - Structurer efficacement un métamodèle de plates-formes
  - Proposer des concepts métiers concrets (boîtes au lettres, tâches …)

Plan de la démarche

- **Qu'est-ce qu'un modèle de plates-formes ?**
  - Etat de l'art et définition
- **Qu'est-ce qu'un métamodèle de plates-formes ?**
  - Introduction du motif « Resource-Service »
  - Spécification et réalisation d'un outillage support
- **Comment capitaliser les transformations de portage ?**
  - Définition d'une infrastructure de transformation
- **Evaluation**
  - Définition du métamodèle Software Resource Modeling
  - Réalisation d'une infrastructure de transformation
- **Conclusion**

Positionnement du motif « Resource-Service »

- Objectifs
  - Définir un bonne pratique pour construire les métamodèles de plates-formes
  - Outiller cette bonne pratique

- Réutilisation des concepts de SPT
  - **Resource** : un mécanisme dont les capacités d'instanciation et d'exécution sont finies.
  - **ResourceInstance** : l'instance d'une ressource.
  - **ResourceService** : un service offert par une ressource à l'application.

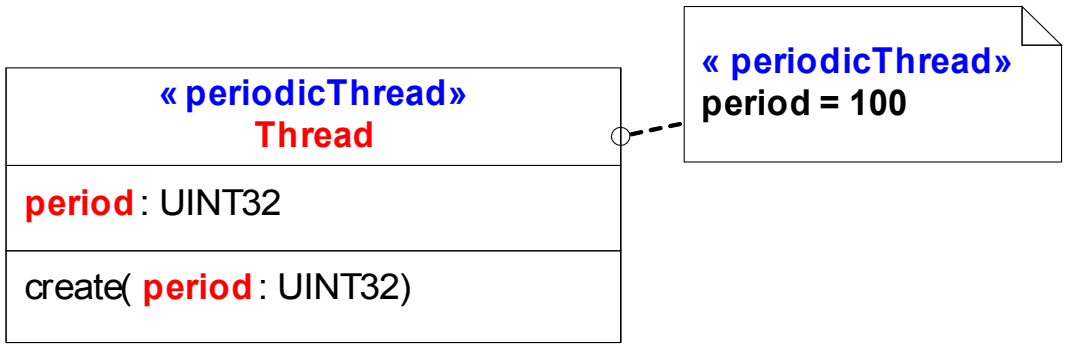
- Limitations
  - Description des propriétés et des paramètres
  - L'héritage de **ResourceInstance** ne permet pas de modéliser les plates-formes

Spécification du motif « Resource-Service »

- Besoins
  - Structurer les concepts autour de la ressource
  - Ne pas imposer une approche de modélisation
  - Trouver « une bonne pratique » pour les propriétés, les paramètres et les services
    - Minimiser les concepts
    - Flexibilité des modélisations

- Le motif résultant doit être une « bonne pratique » (un motif) pour concevoir un métamodèle de plate-forme

- Ce ne doit pas être une nouvelle API

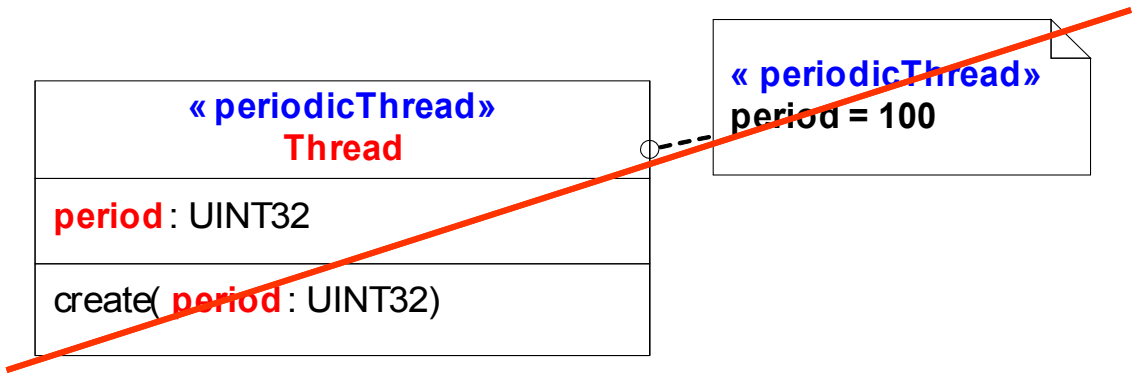


Spécification du motif « Resource-Service »

- Besoins
  - Structurer les concepts autour de la ressource
  - Ne pas imposer une approche de modélisation
  - Trouver « une bonne pratique » pour les propriétés, les paramètres et les services
    - Minimiser les concepts
    - Flexibilité des modélisations

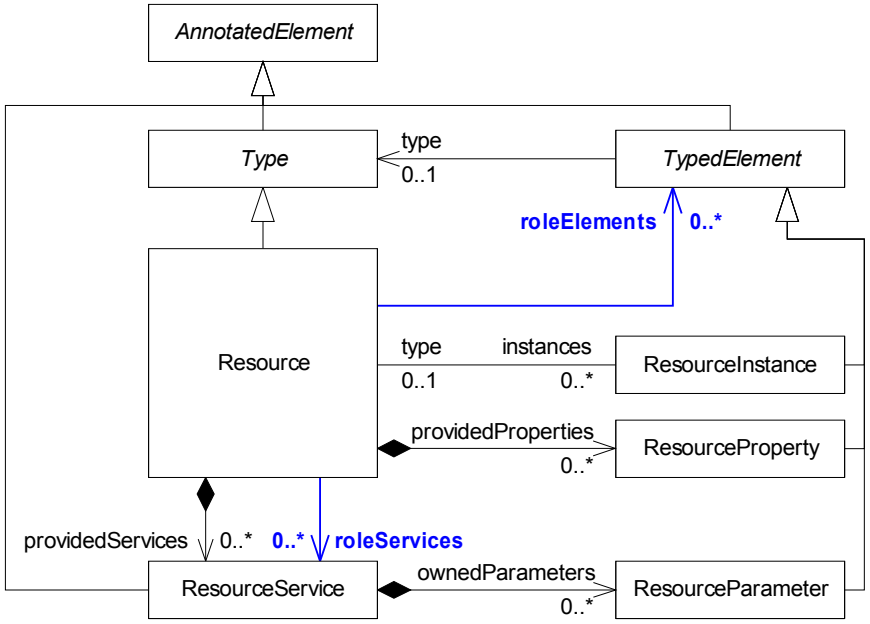
- Le motif résultant doit être une « bonne pratique » (un motif) pour concevoir un métamodèle de plate-forme

- Ce ne doit pas être une nouvelle API



Réalisation du motif « Resource-Service »

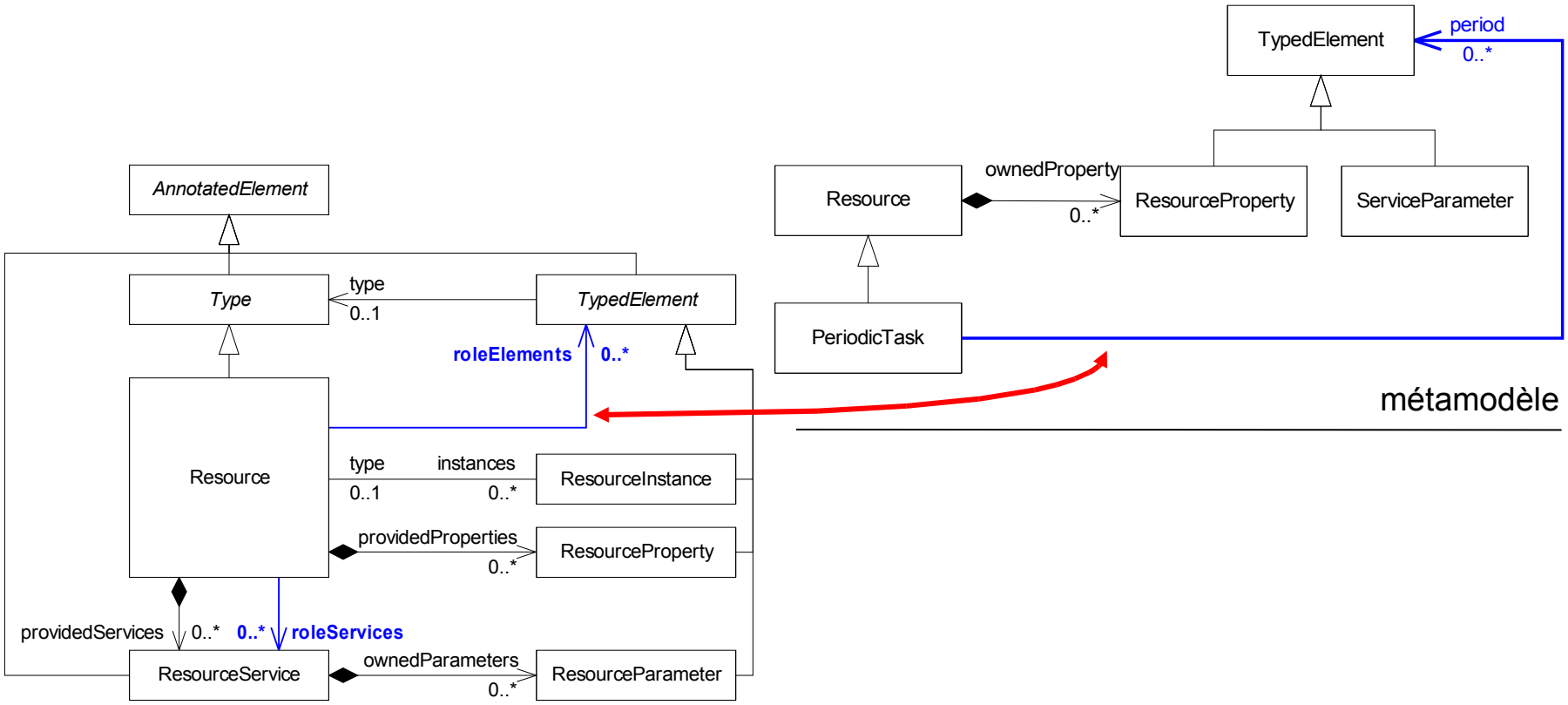
Le motif « Resource-Service »



Réalisation du motif « Resource-Service »

Le motif « Resource-Service »

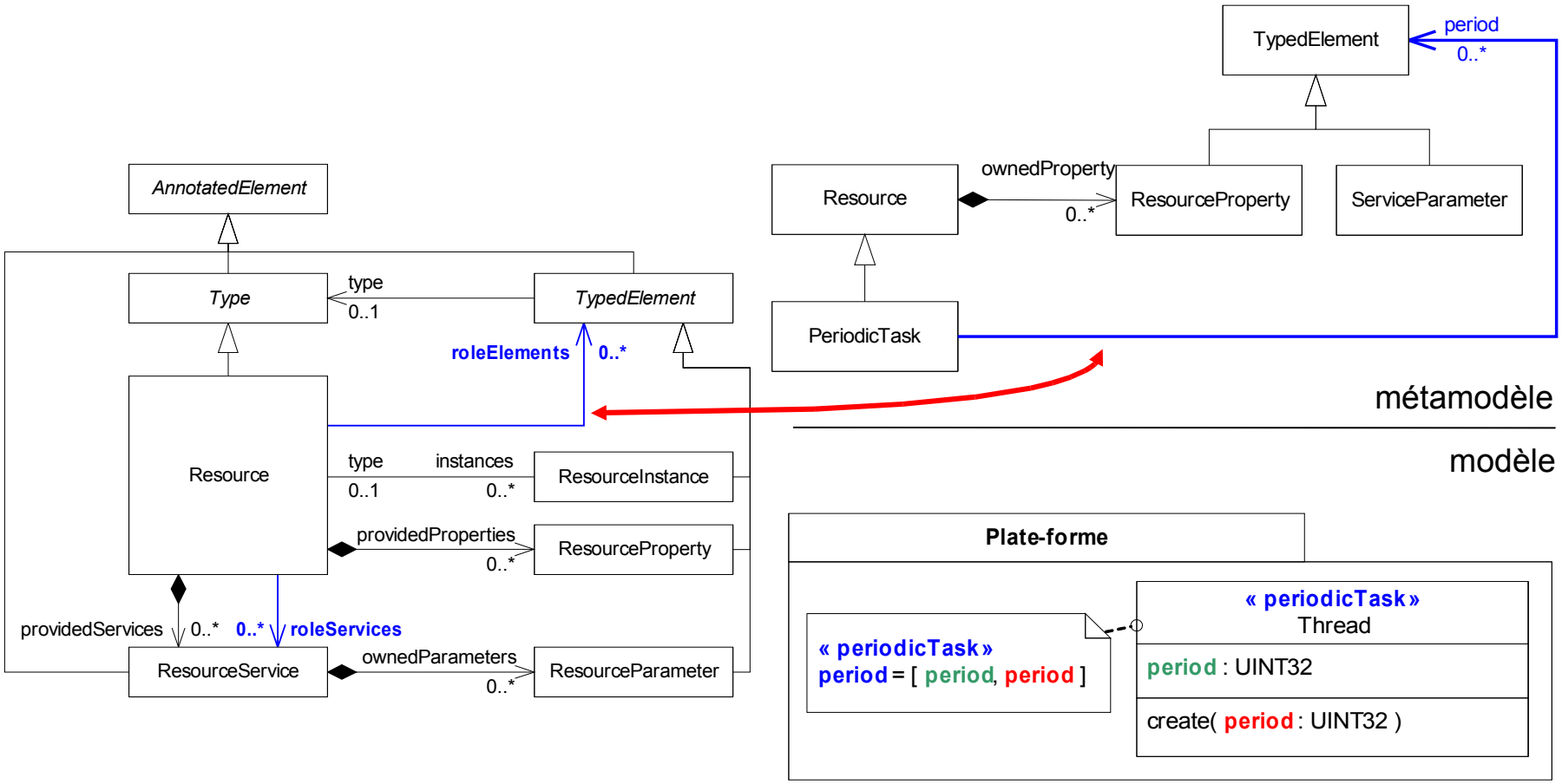
Application de cette bonne pratique



Réalisation du motif « Resource-Service »

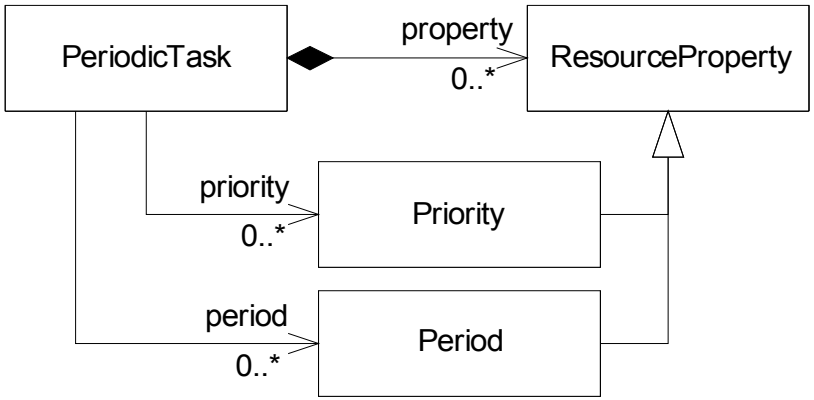
Le motif « Resource-Service »

Application de cette bonne pratique



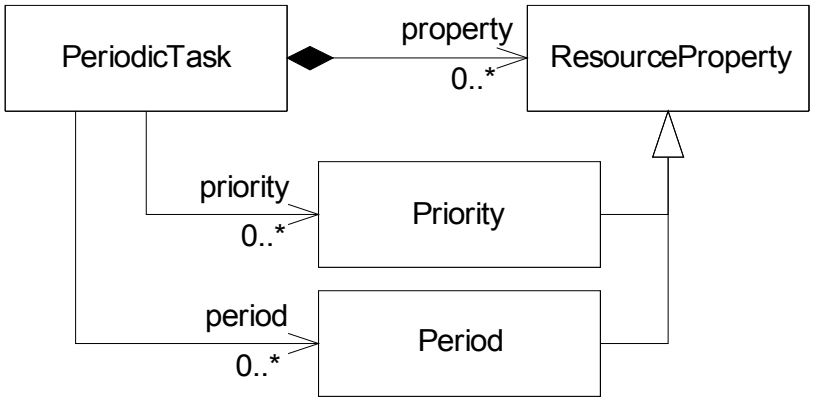
Justification du motif « Resource-Service »

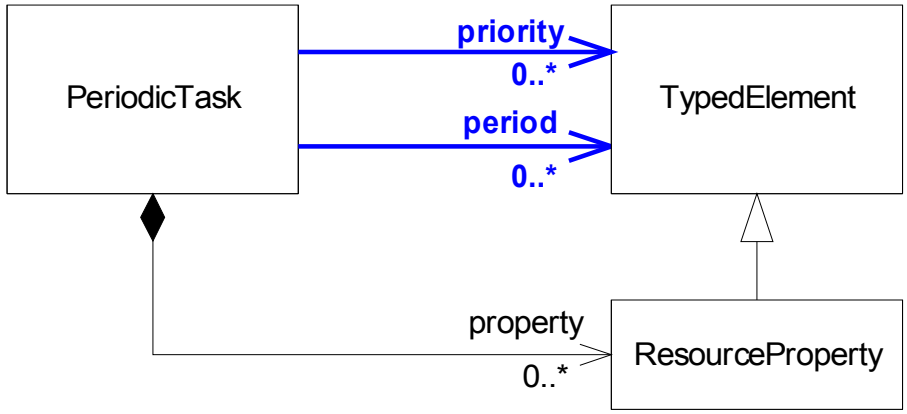
- Limite le nombre de métatypes



Justification du motif « Resource-Service »

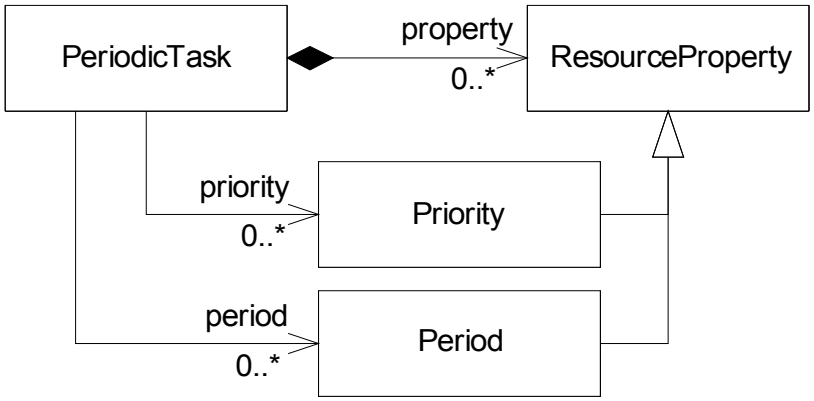
- Limite le nombre de métatypes

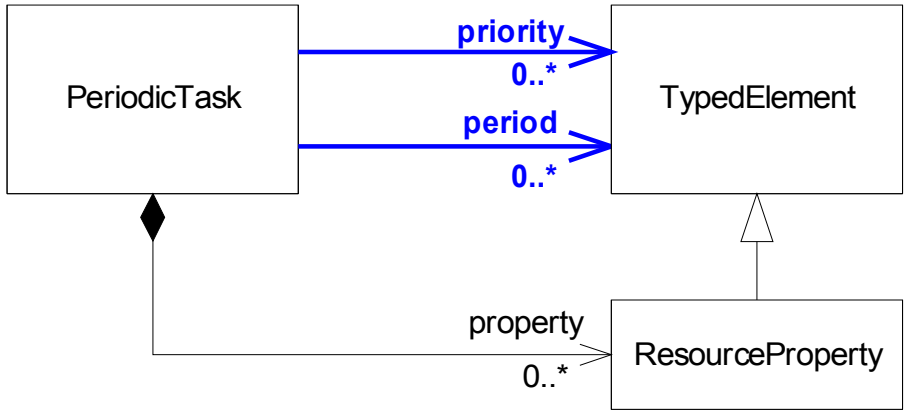




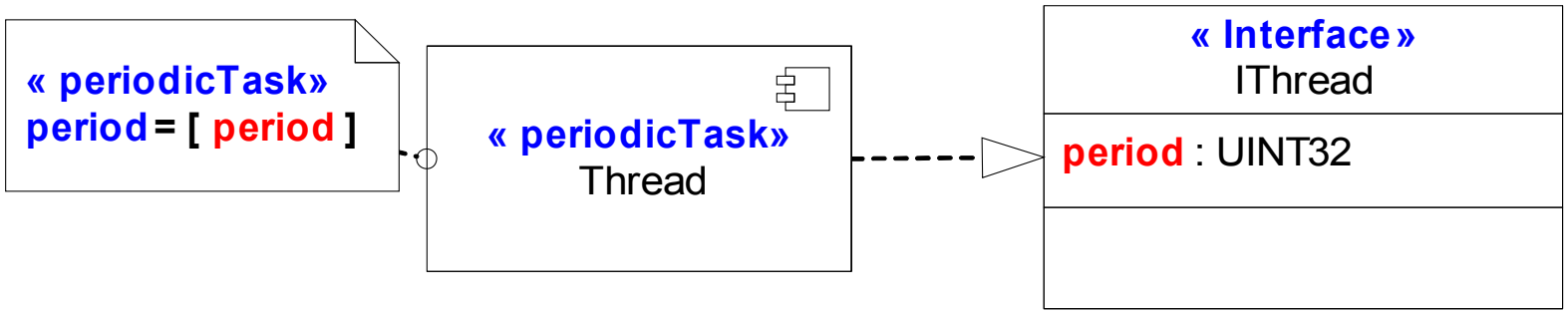
Justification du motif « Resource-Service »

- Limite le nombre de métatypes



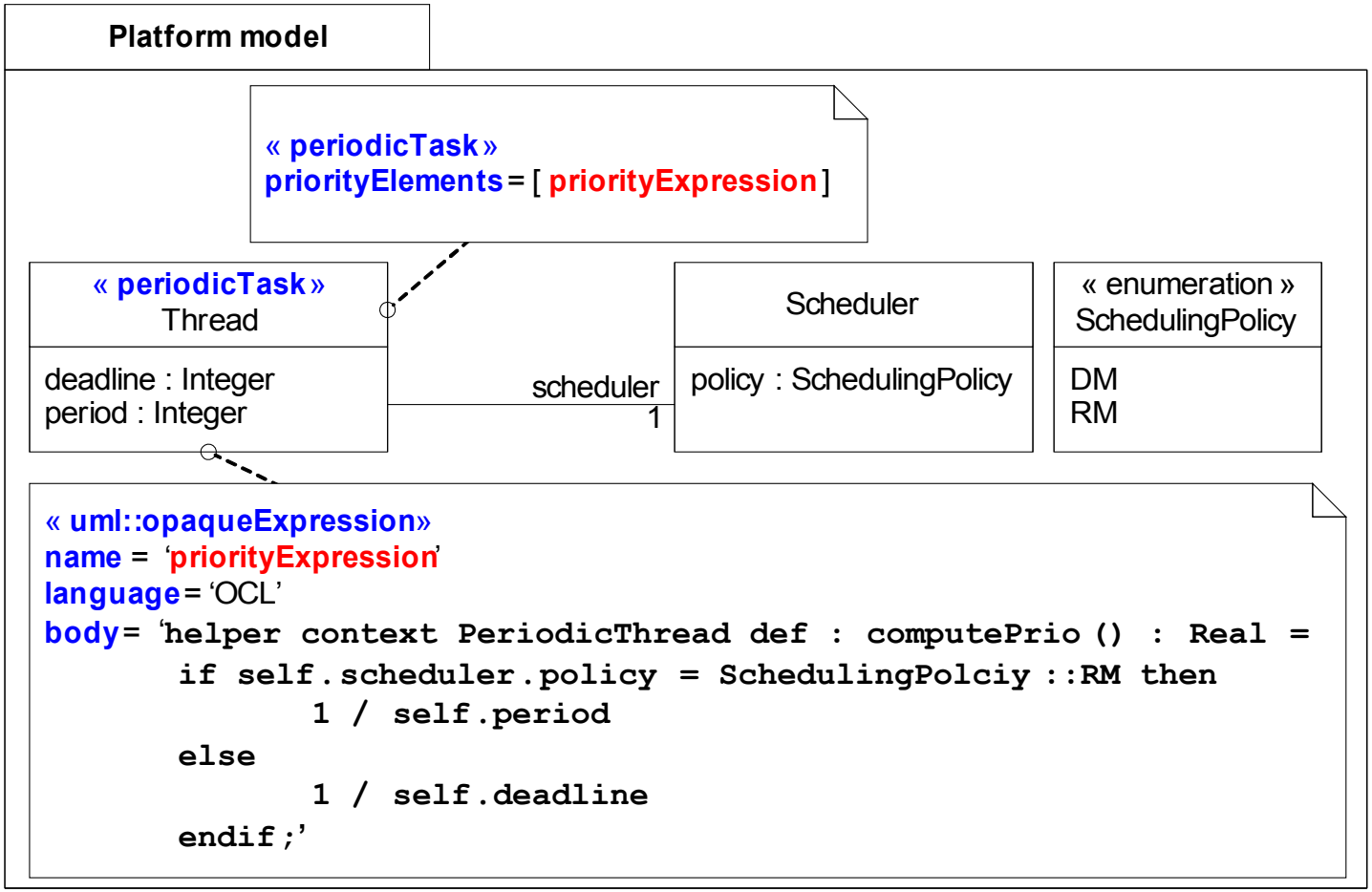


- Indépendance d'une méthodologie de modélisation



Justification du motif Resource-Service

- Description de caractéristiques implicites
- Pour les services utilisation de diagramme d'activité



Outillage du motif

- Formalisation d'un métamodèle de plate-forme :
  - Un métamodèle de plate-forme est un métamodèle appliquant le motif « Resource-Service »

- Spécification d'un ensemble de services outillant le motif
  - Exemples :
    - Toutes les références pour une propriété
    - Toutes les références pour un service
    - Opérateurs (équivalence)

- Spécification sous la forme d'ensembles mathématiques
  - Implantation d'une librairie OCL manipulée dans des transformations ATL

- Besoin
  - Utiliser ce motif dans un processus dirigé par les plates-formes

Plan de la démarche

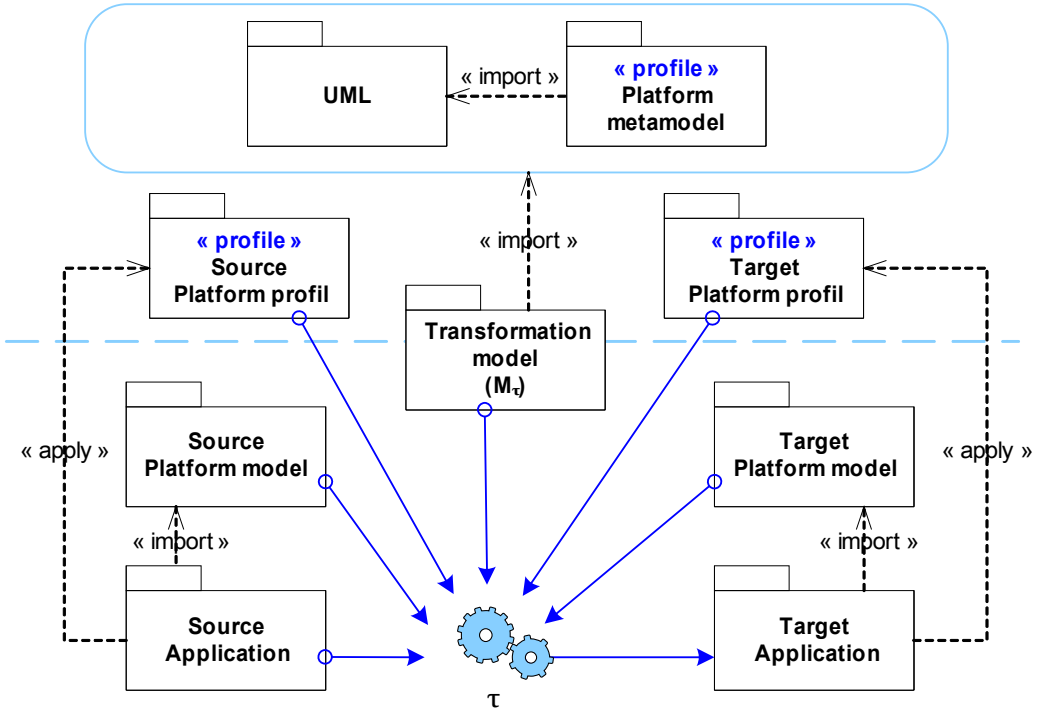
- **Qu'est-ce qu'un modèle de plates-formes ?**
  - Etat de l'art et définition
- **Qu'est-ce qu'un métamodèle de plates-formes ?**
  - Introduction du motif « Resource-Service »
  - Spécification et réalisation d'un outillage support
- **Comment capitaliser les transformations de portage ?**
  - Définition d'une infrastructure de transformation
- **Evaluation**
  - Définition du métamodèle Software Resource Modeling
  - Réalisation d'une infrastructure de transformation
- **Conclusion**

Approche de transformation

- Objectifs
  - Utiliser les métamodèles de plates-formes comme des pivots sémantiques
  - Définir une architecture de transformation automatisant les portages

- Métamodèles de plate-forme sont des bases de connaissances pour inférer les portages

- Le motif « Resource-Service » généralise le moteur d'inférences

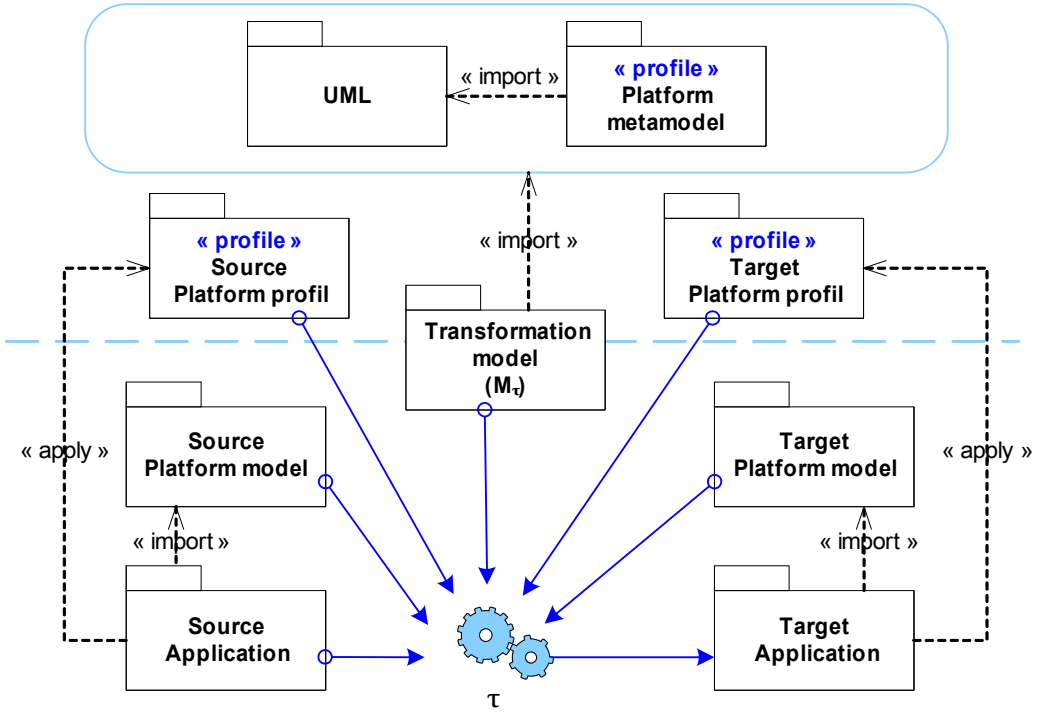


Approche de transformation

- Objectifs
  - Utiliser les métamodèles de plates-formes comme des pivots sémantiques
  - Définir une architecture de transformation automatisant les portages

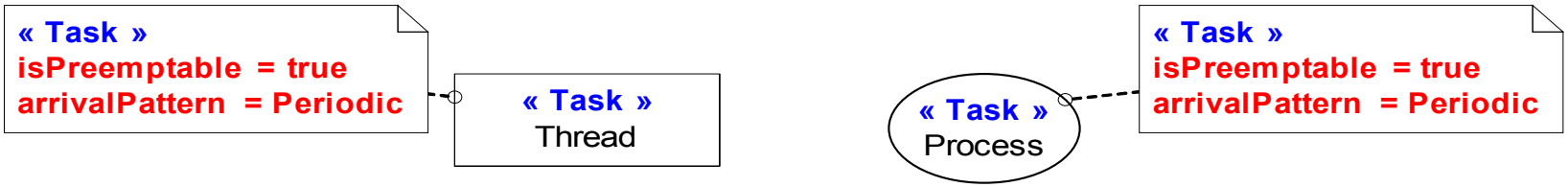
- Métamodèles de plate-forme sont des bases de connaissances pour inférer les portages

- Le motif « Resource-Service » généralise le moteur d'inférences



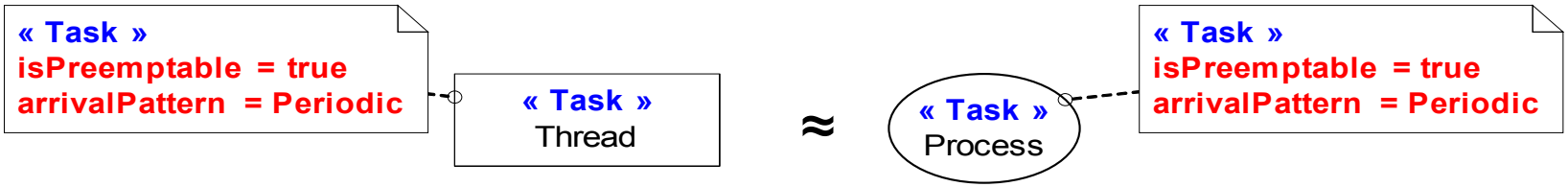
Inférence par équivalence

- Inférence basée sur un opérateur d'équivalence
  - Equivalence (≈) observée entre deux éléments
  - Eléments décrits par un même métamodèle de plate-forme
  - L'équivalence est évaluée sur les méta-propriétés
    - primitives (entières, booléennes, …)
    - énumérées



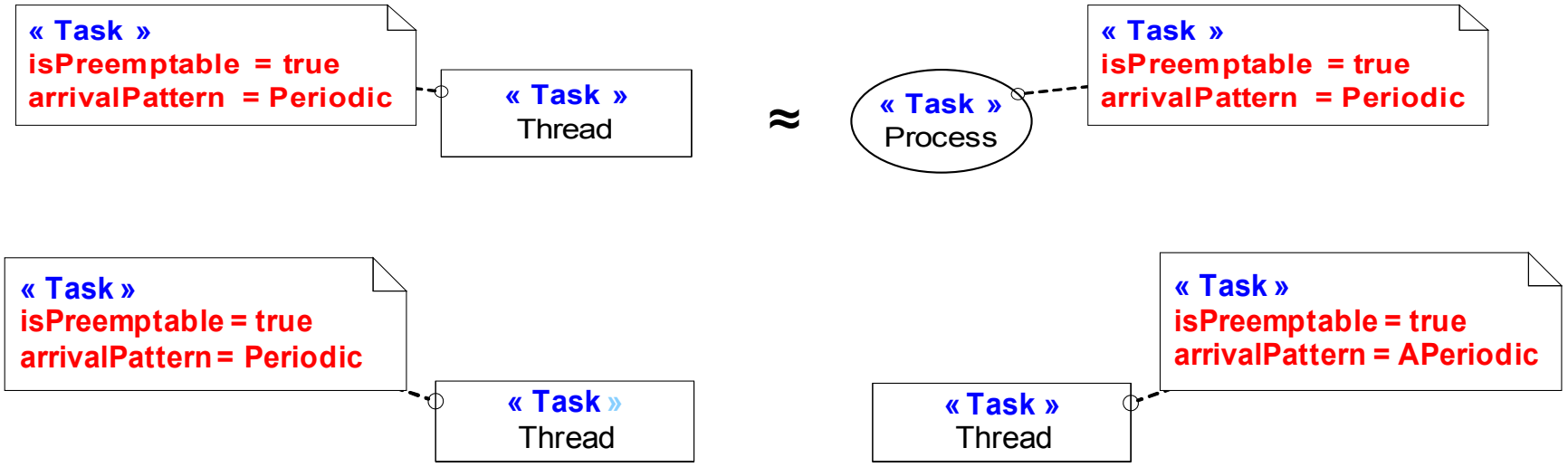
Inférence par équivalence

- Inférence basée sur un opérateur d'équivalence
  - Equivalence (≈) observée entre deux éléments
  - Eléments décrits par un même métamodèle de plate-forme
  - L'équivalence est évaluée sur les méta-propriétés
    - primitives (entières, booléennes, …)
    - énumérées



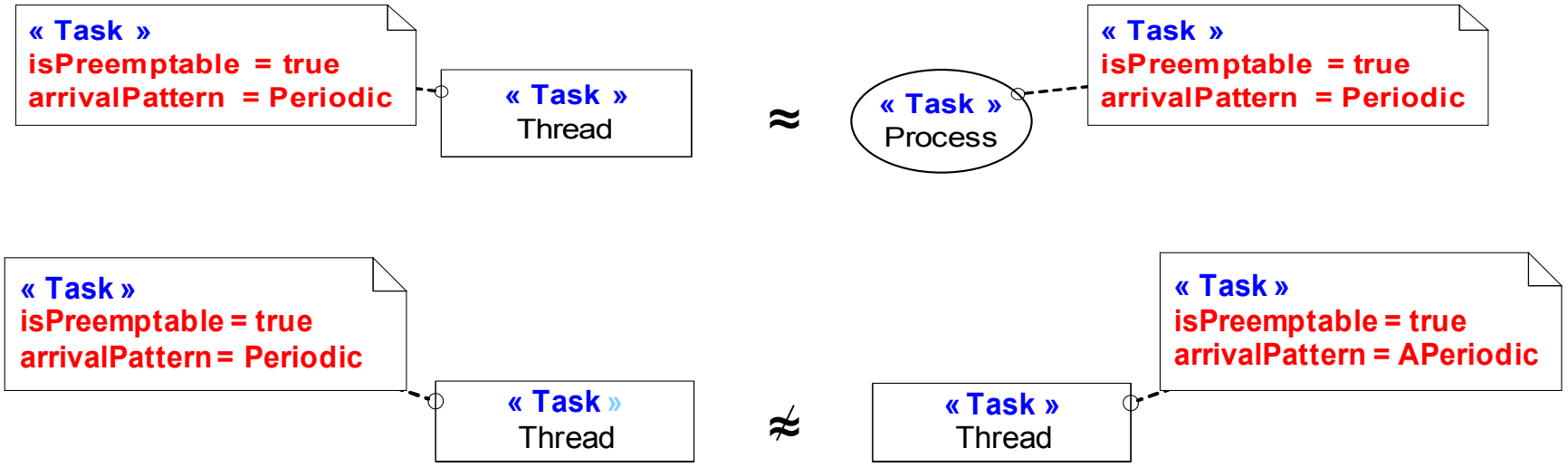
Inférence par équivalence

- Inférence basée sur un opérateur d'équivalence
  - Equivalence (≈) observée entre deux éléments
  - Eléments décrits par un même métamodèle de plate-forme
  - L'équivalence est évaluée sur les méta-propriétés
    - primitives (entières, booléennes, …)
    - énumérées



Inférence par équivalence

- Inférence basée sur un opérateur d'équivalence
  - Equivalence (≈) observée entre deux éléments
  - Eléments décrits par un même métamodèle de plate-forme
  - L'équivalence est évaluée sur les méta-propriétés
    - primitives (entières, booléennes, …)
    - énumérées



Algorithme de portage des ressources

- Pour chaque **instance** {

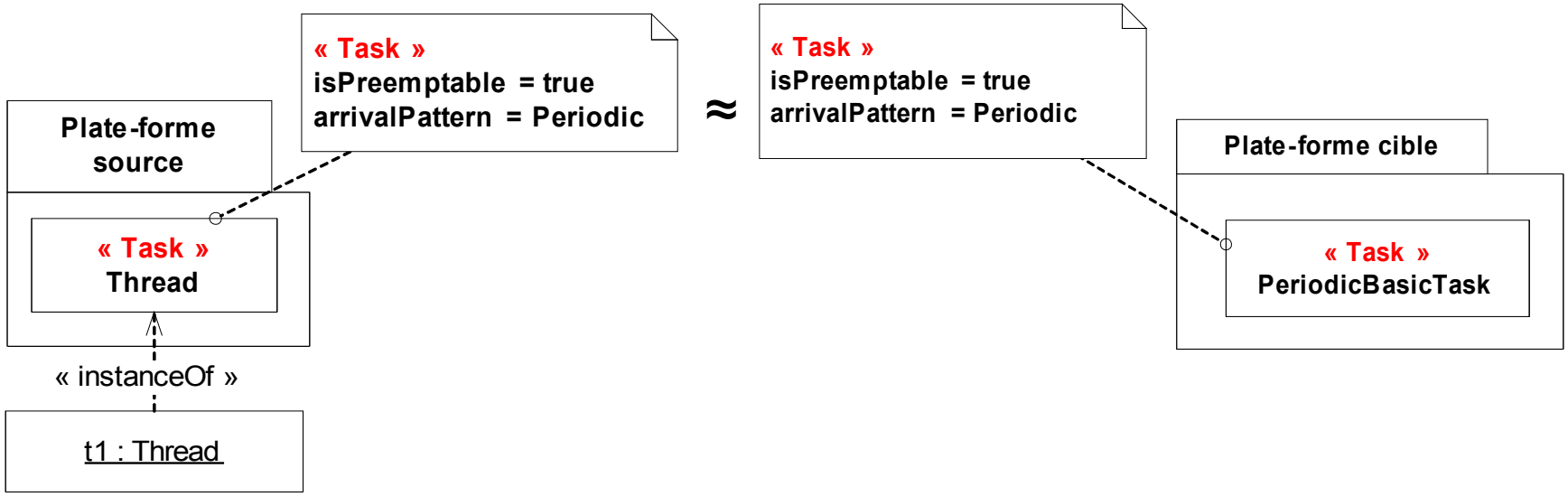
pour chaque **ressource** typant l'instance {

s'il existe une ressource **équivalente** dans la plate-forme cible **alors**

générer une instance de cette ressource cible

}

}



Algorithme de portage des ressources

- Pour chaque **instance** {

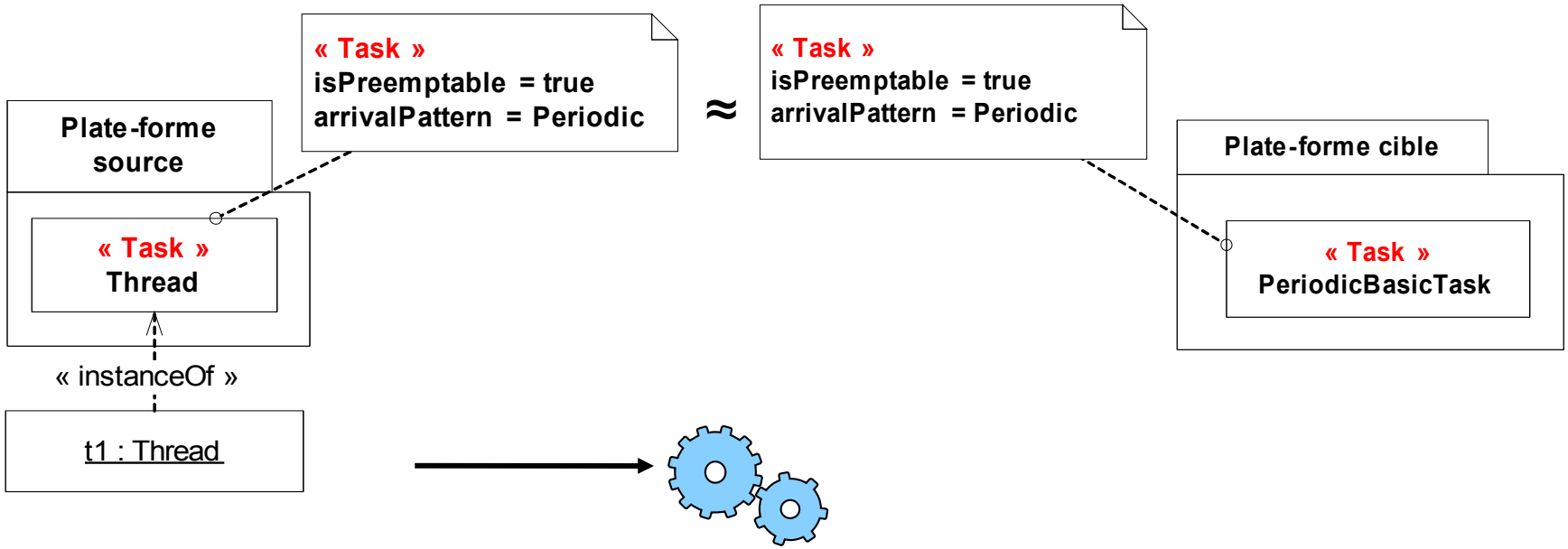
pour chaque **ressource** typant l'instance {

s'il existe une ressource **équivalente** dans la plate-forme cible **alors**

générer une instance de cette ressource cible

}

}



Algorithme de portage des ressources

- Pour chaque **instance** {

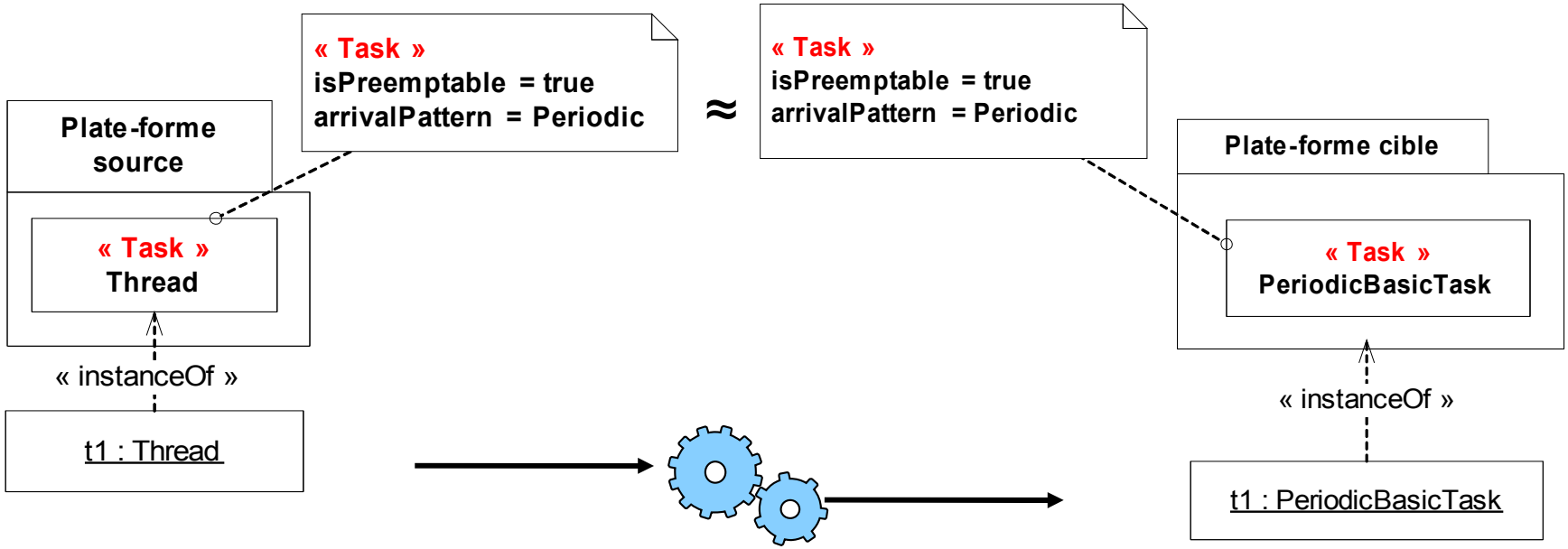
pour chaque **ressource** typant l'instance {

s'il existe une ressource **équivalente** dans la plate-forme cible **alors**

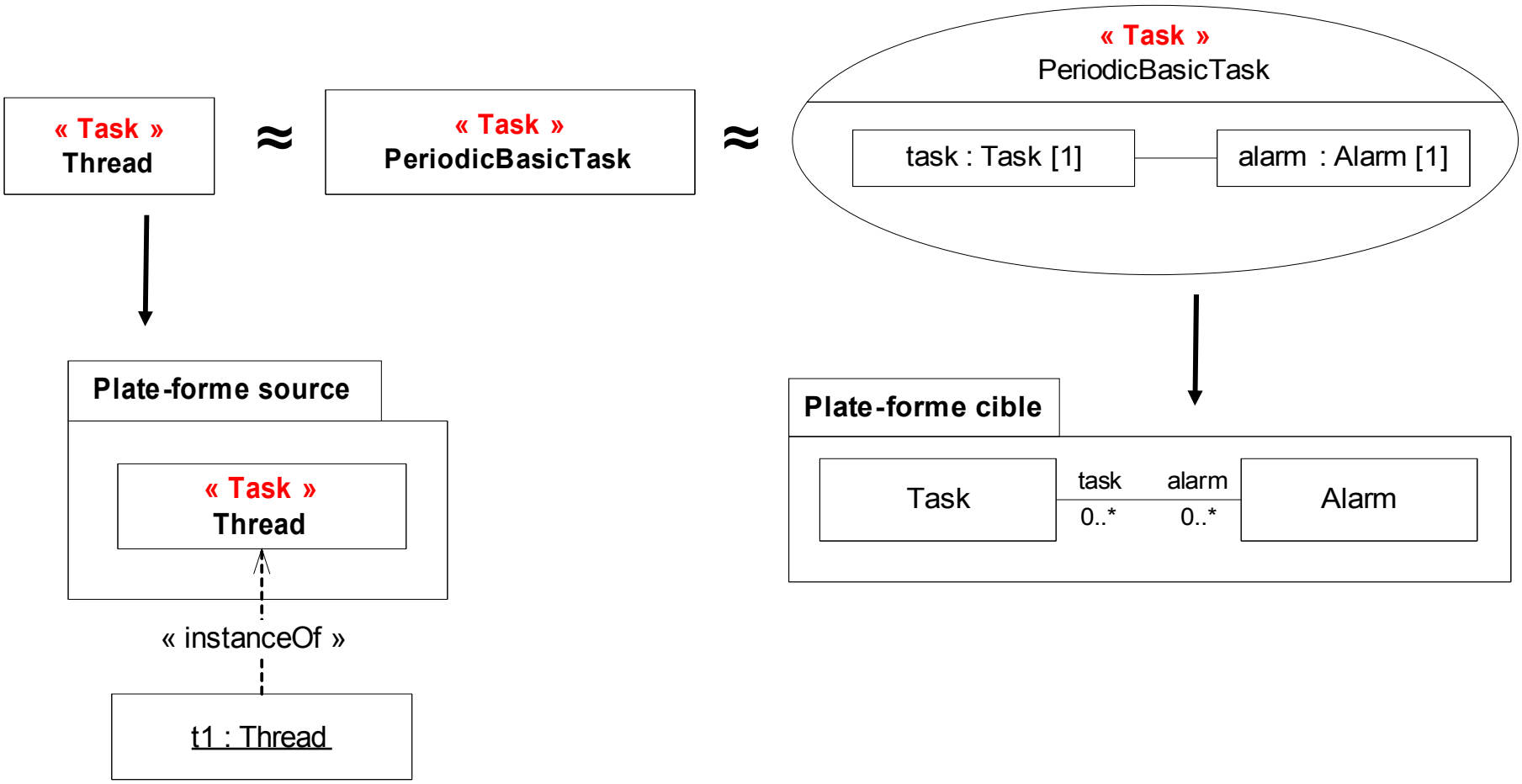
générer une instance de cette ressource cible

}

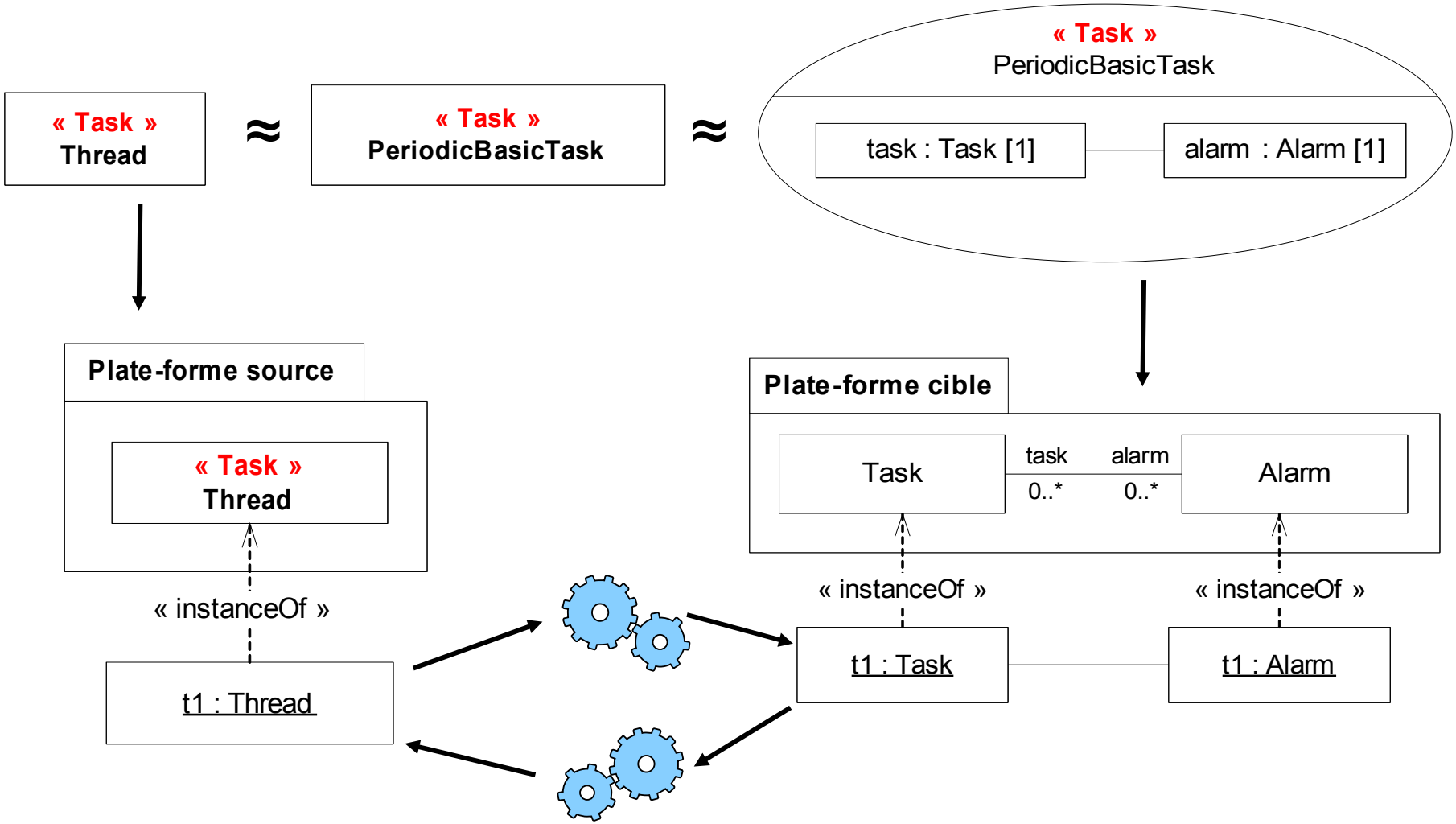
}



Portage des ressources



Portage des ressources



Portage des propriétés

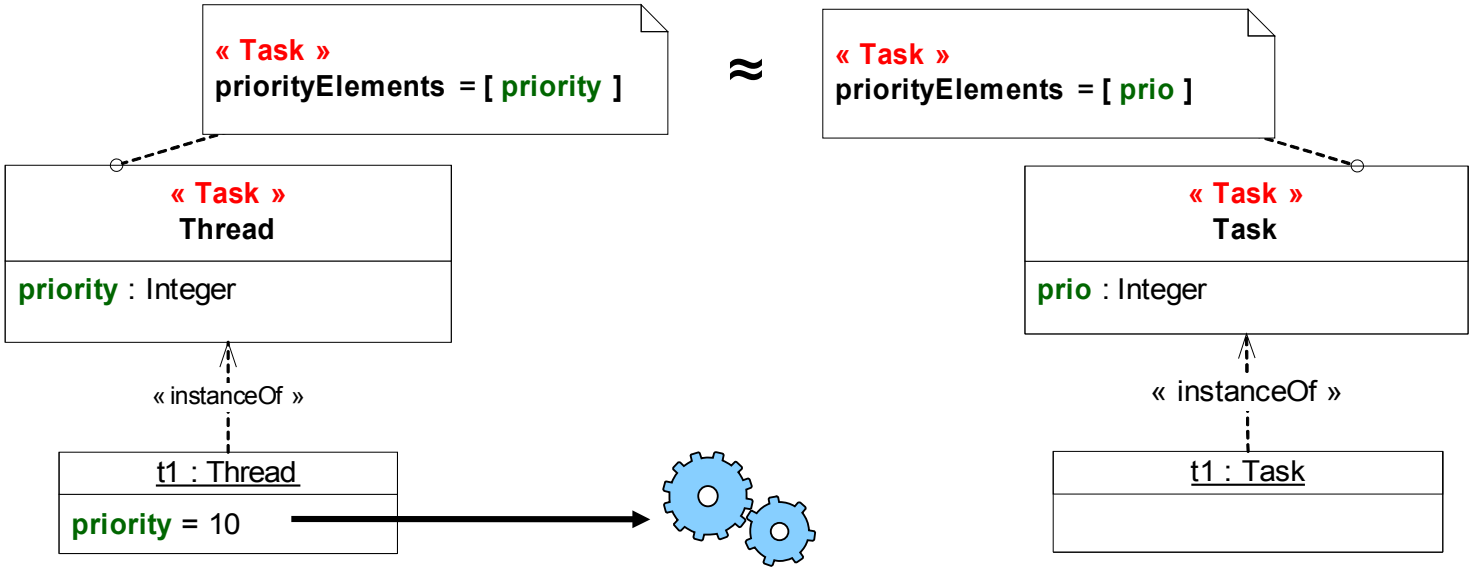
- Raisonnement identique pour les propriétés ou les services
  - Equivalence sur les références
  - **Pour chaque emplacement valué** {

si son **type est référencé** alors

créer un emplacement dans la cible

typer cette emplacement **par la propriété référencée de la même façon** dans la ressource cible

}



Portage des propriétés

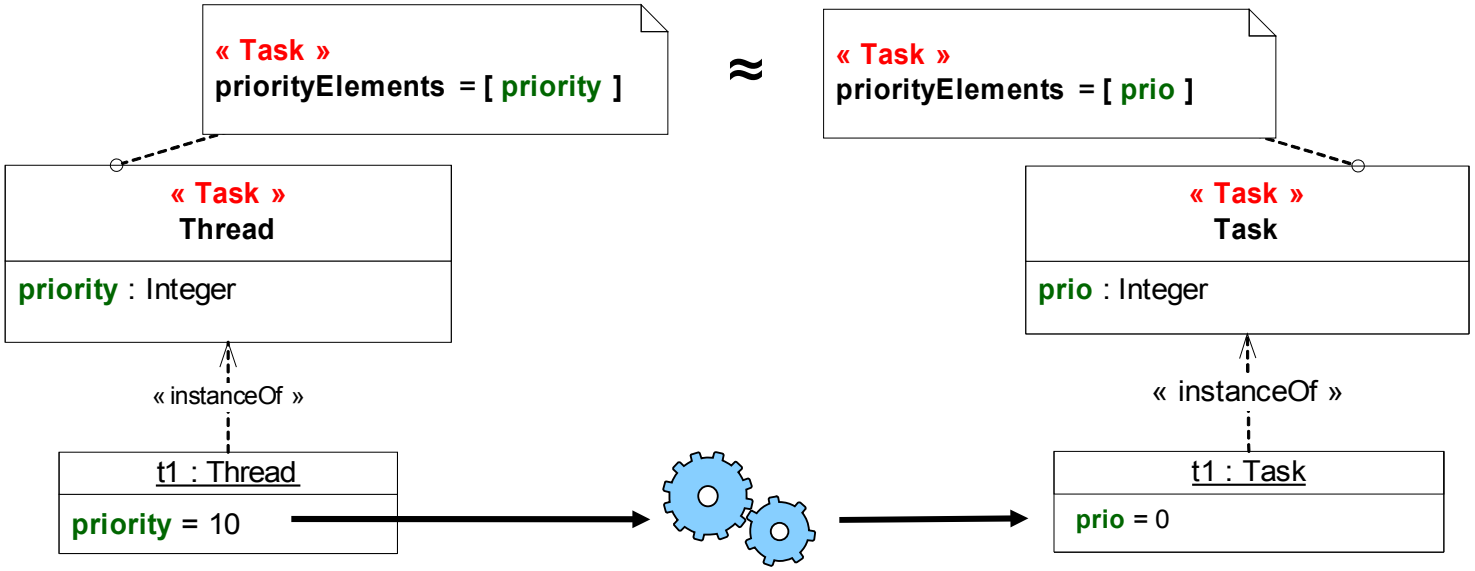
- Raisonnement identique pour les propriétés ou les services
  - Equivalence sur les références
  - **Pour chaque emplacement valué** {

si son **type est référencé** alors

créer un emplacement dans la cible

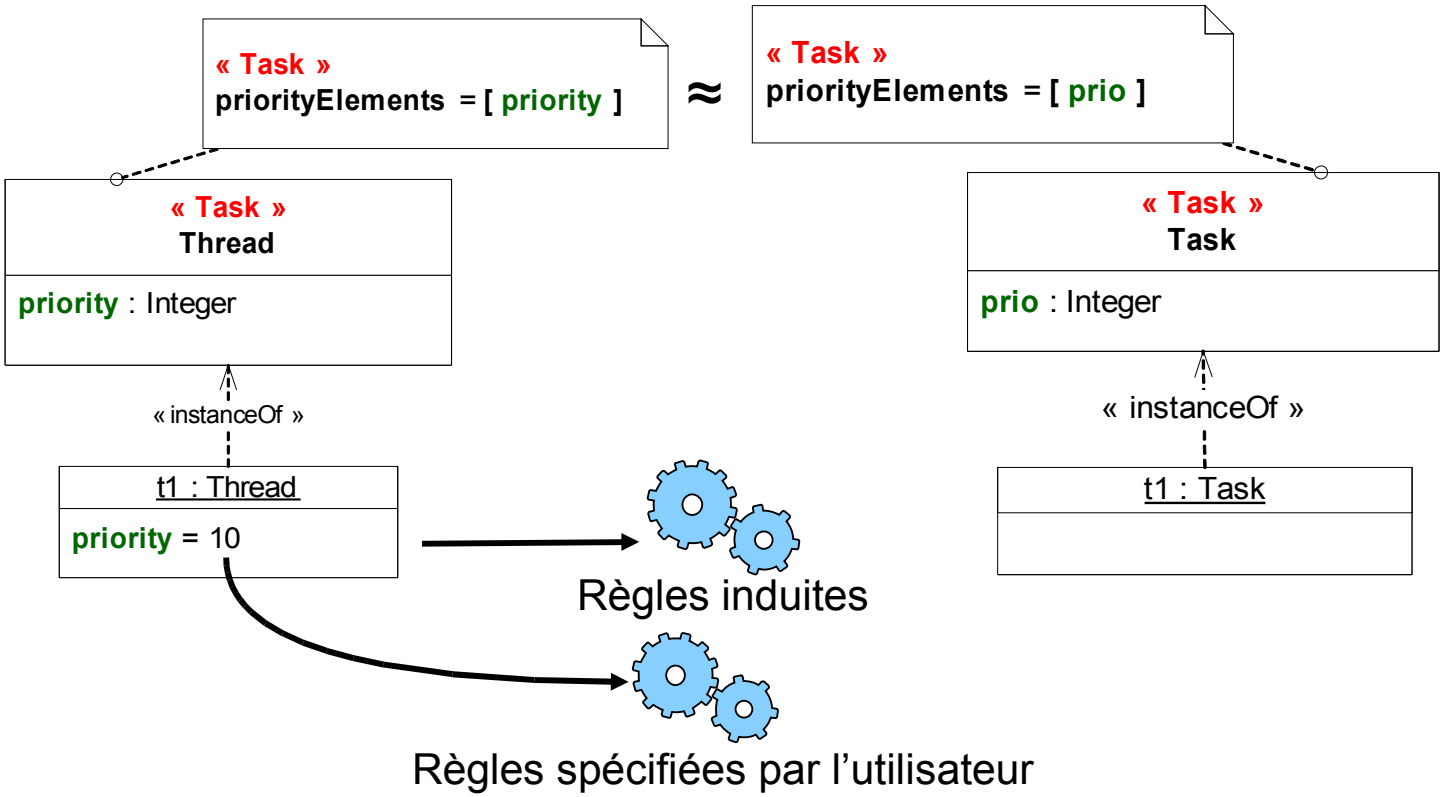
typer cette emplacement **par la propriété référencée de la même façon** dans la ressource cible

}



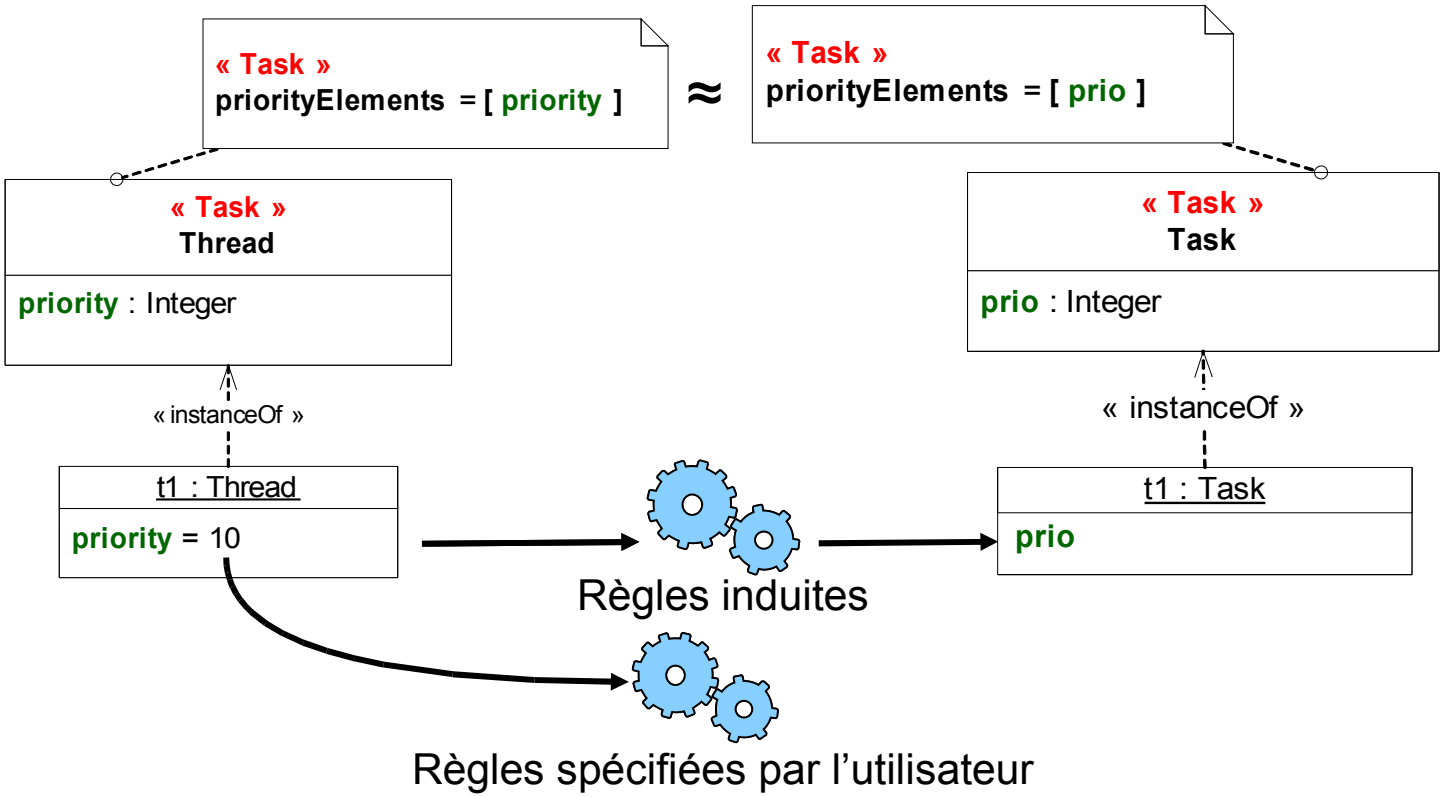
Conception de l'opération de portage

- Transformation des types de données (valeurs primitives, énumérées et données)
  - Nécessite l'intervention de l'utilisateur
  - L'utilisateur décrit ses propres règles de transformation



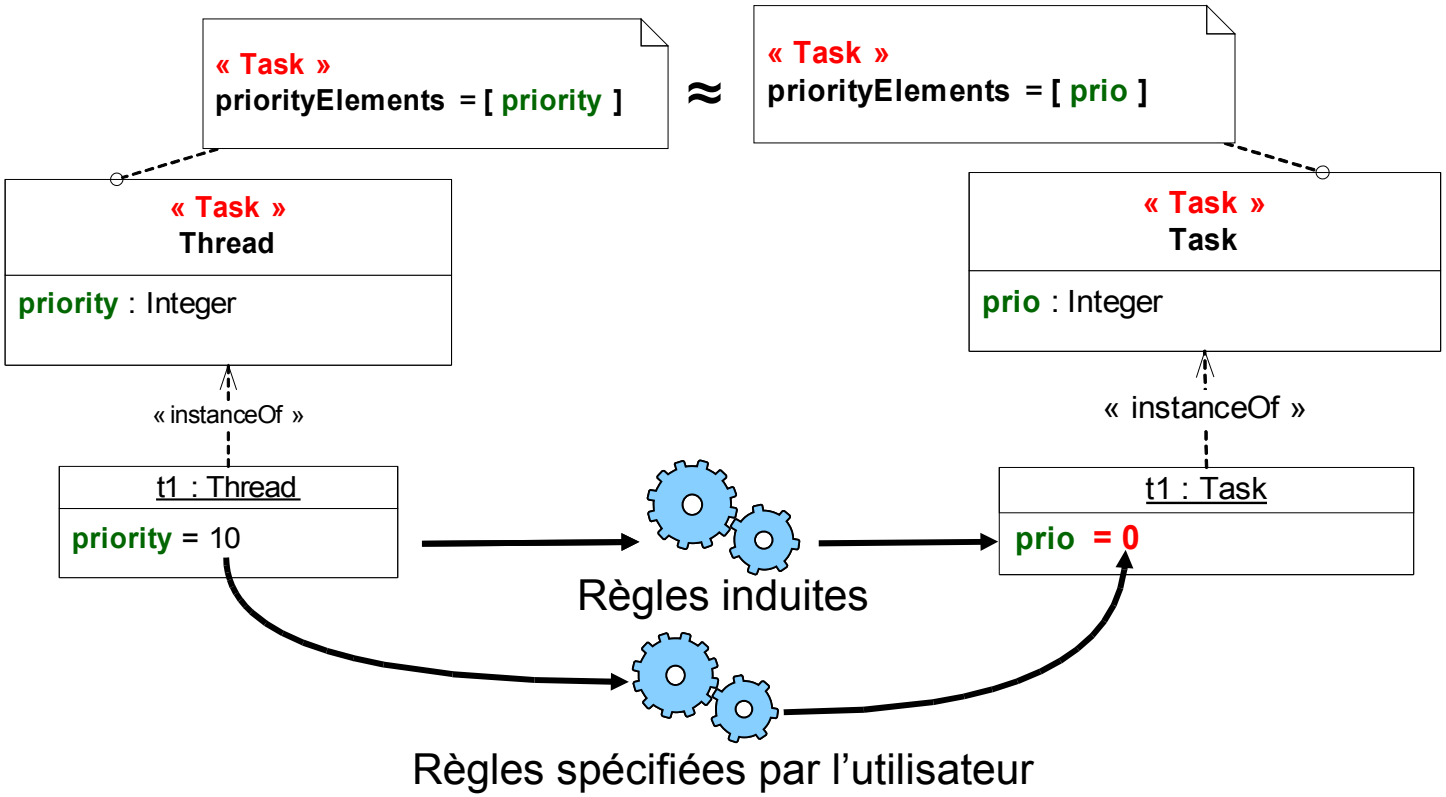
Conception de l'opération de portage

- Transformation des types de données (valeurs primitives, énumérées et données)
  - Nécessite l'intervention de l'utilisateur
  - L'utilisateur décrit ses propres règles de transformation



Conception de l'opération de portage

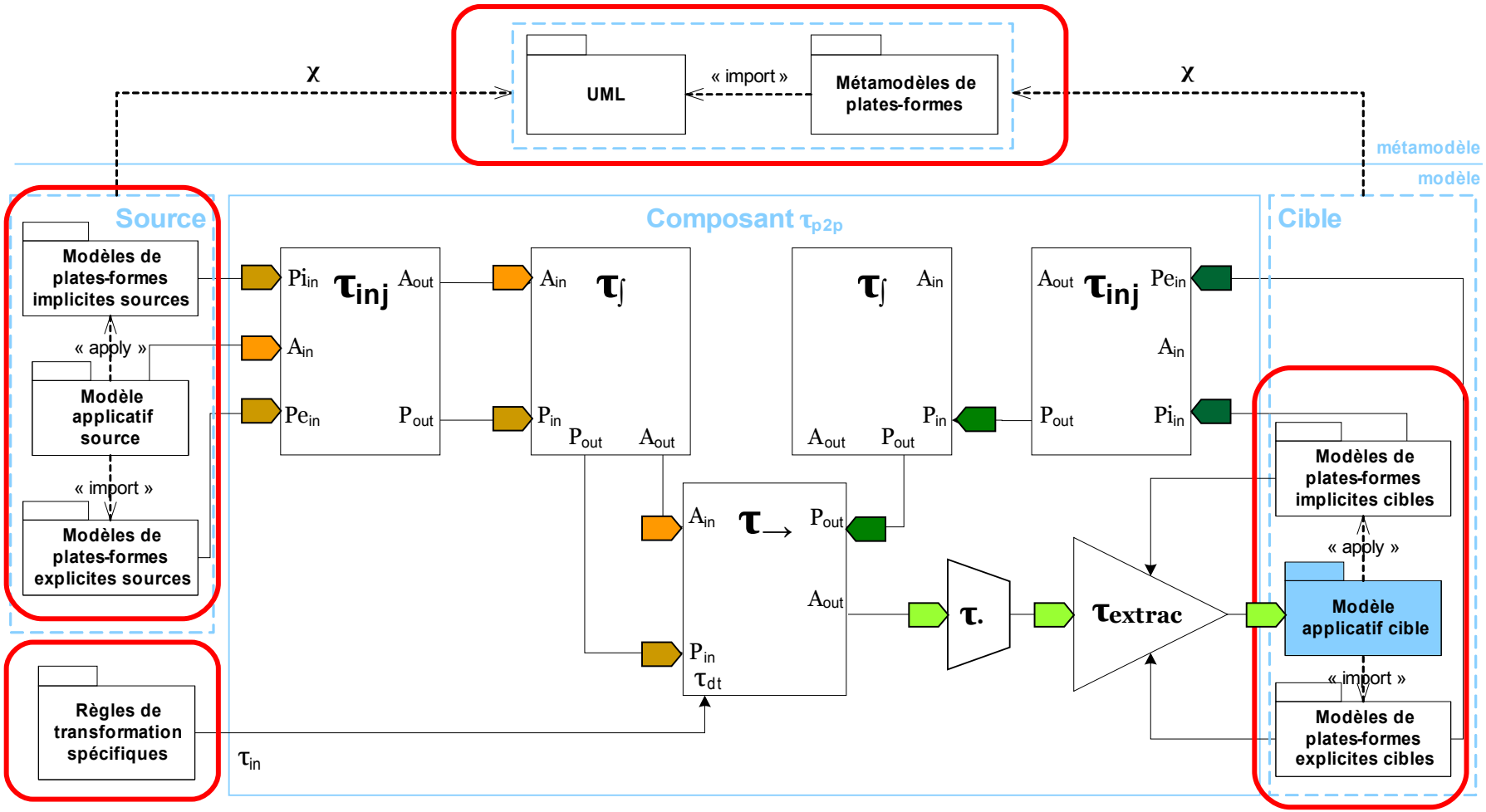
- Transformation des types de données (valeurs primitives, énumérées et données)
  - Nécessite l'intervention de l'utilisateur
  - L'utilisateur décrit ses propres règles de transformation



Conception de l'infrastructure

- Etapes de transformations
  1. Injection
     - Explicite les plates-formes implicites (promotion inverse)
     - Modifie le modèle applicatif source
  2. Intégration
     - Factorise les modèles de plates-formes sources et cibles
     - Produit des plates-formes abstraites
     - Rend l'application spécifique aux plates-formes abstraites sources
  3. Portage
     - Porte l'application spécifique aux plates-formes sources
     - Produit l'application spécifique aux plates-formes cibles
  4. Dérivation
     - Inverse l'opération d'intégration
     - Produit une application spécifique aux plates-formes cibles
  5. Extraction
     - Inverse l'opération d'injection
     - Implicite les plates-formes cibles décrites au niveau méta

Conception de l'infrastructure

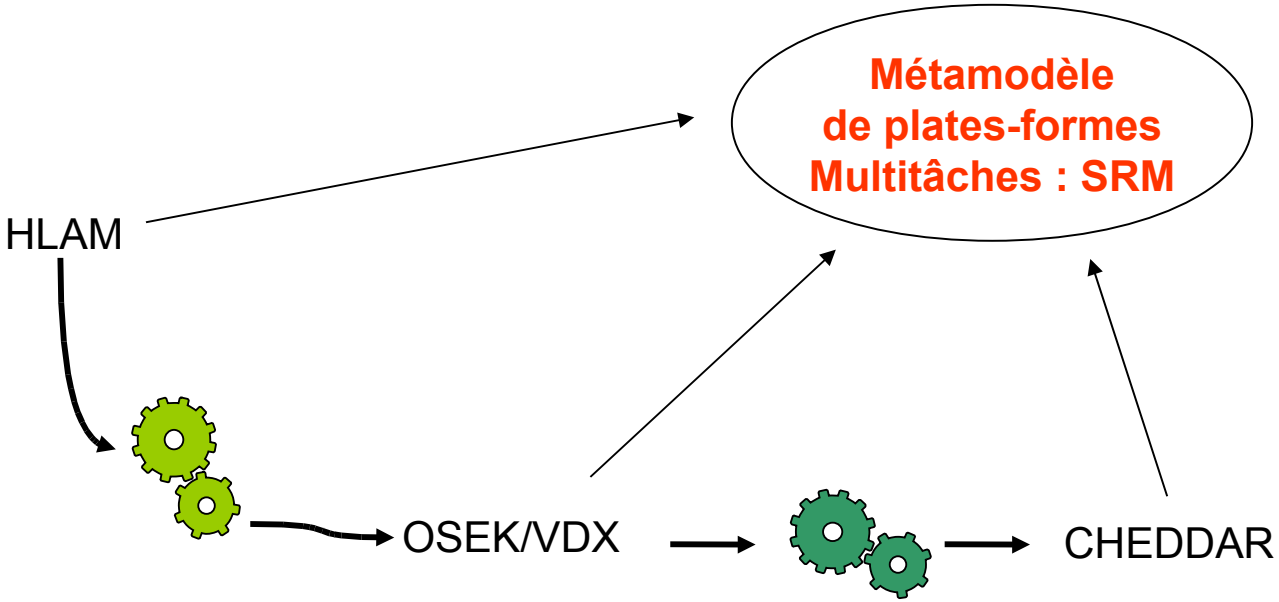


Plan de la démarche

- **Qu'est-ce qu'un modèle de plates-formes ?**
  - Etat de l'art et définition
- **Qu'est-ce qu'un métamodèle de plates-formes ?**
  - Introduction du motif « Resource-Service »
  - Spécification et réalisation d'un outillage support
- **Comment capitaliser les transformations de portage ?**
  - Définition d'une infrastructure de transformation
- **Evaluation**
  - Définition du métamodèle Software Resource Modeling
  - Réalisation d'une infrastructure de transformation
- **Conclusion**

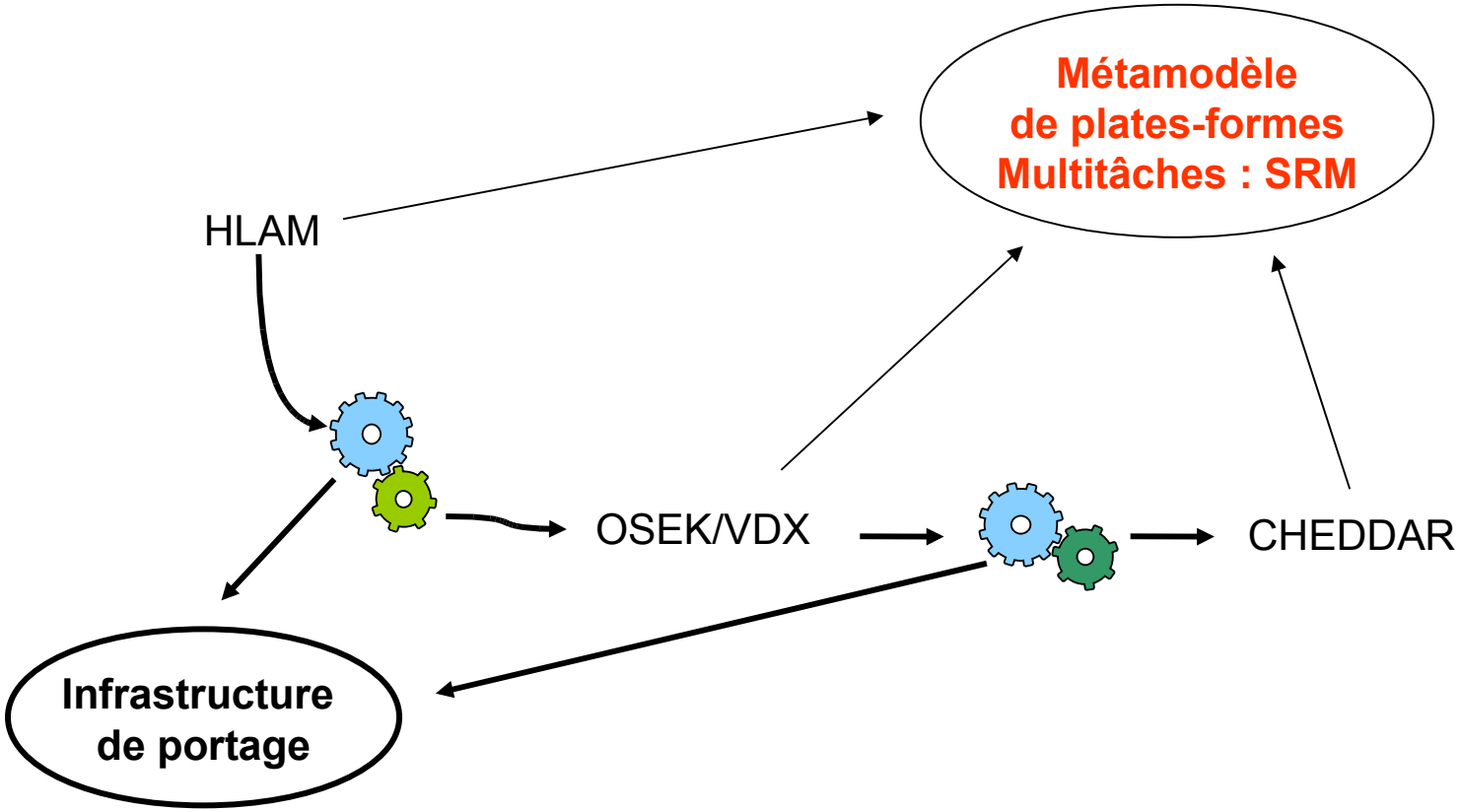
Contexte expérimental

- Objectifs
  - Générer le fichier de configuration d'OSEK/VDX (fichier OIL)
  - Générer le fichier de configuration de l'outil CHEDDAR



Contexte expérimental

- Objectifs
  - Générer le fichier de configuration d'OSEK/VDX (fichier OIL)
  - Générer le fichier de configuration de l'outil CHEDDAR



Plates-formes expérimentales

- Modéliser différentes plates-formes avec SRM
  - HLAM
    - Un métamodèle (plate-forme implicite)
    - rtUnit : partition dans lequel évolue plusieurs contextes d'exécution
    - rtService : contexte d'exécution concurrent (tâche) périodique
  - OSEK/VDX
    - Un modèle (plate-forme explicite)
    - Un système d'exploitation multitâche (Ordonnanceur, Tâche, …)
  - CHEDDAR
    - Un modèle (plate-forme explicite)
    - Un outil d'analyse d'ordonnancement
      - Considéré comme une plate-forme exécutant des descriptions multitâches
      - Tâche, ordonnanceur

Software Resource Modeling (SRM)

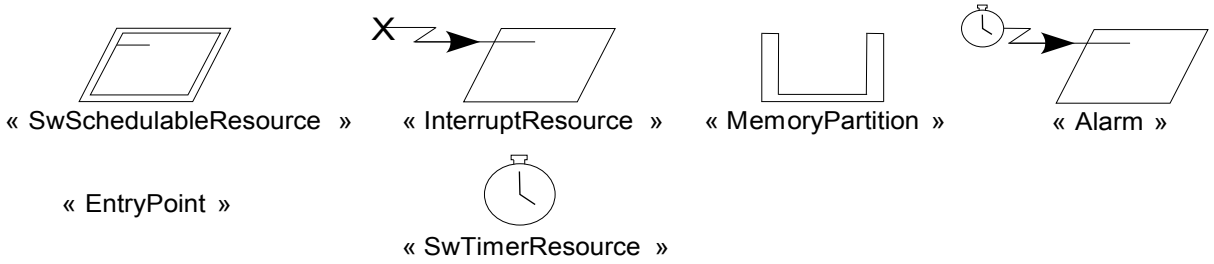
- Objectifs
  - Décrire un profil UML dédié à la modélisation des API de plates-formes multitâches
  - Appliquer le motif « Resource-Service » à ce profil

- Démarche de construction de SRM
  - API industrielles
    - VxWorks, RTAI,
  - API standards
    - POSIX, OSEK/VDX
  - API académiques
    - SCEPTRE 2, Lacatre

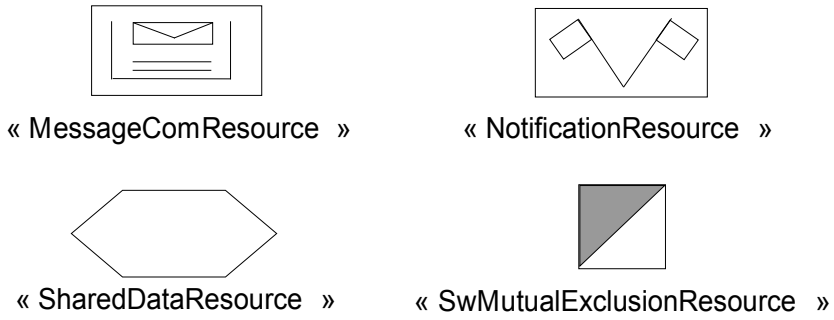
  - API commune

Le profil UML SRM

SRM::SW_Concurrency

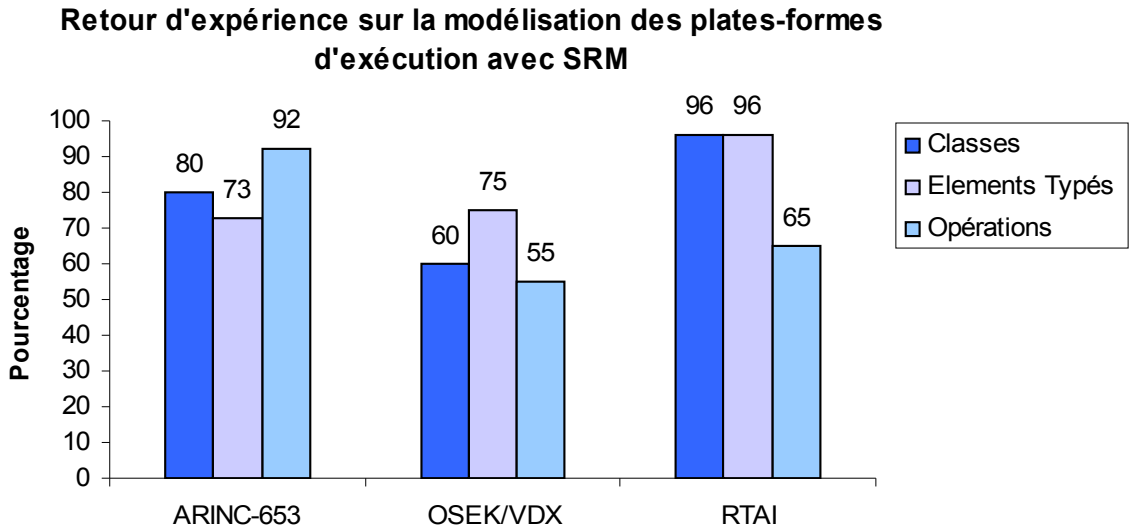


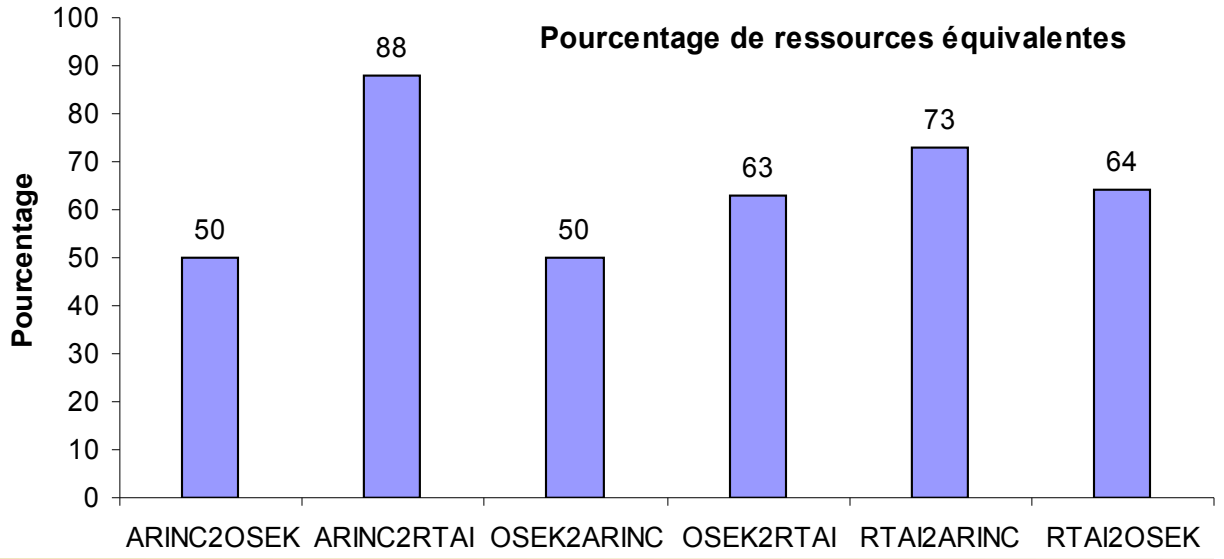
SRM::SW_Interaction



SRM::SW_Brokering

Evaluation de SRM

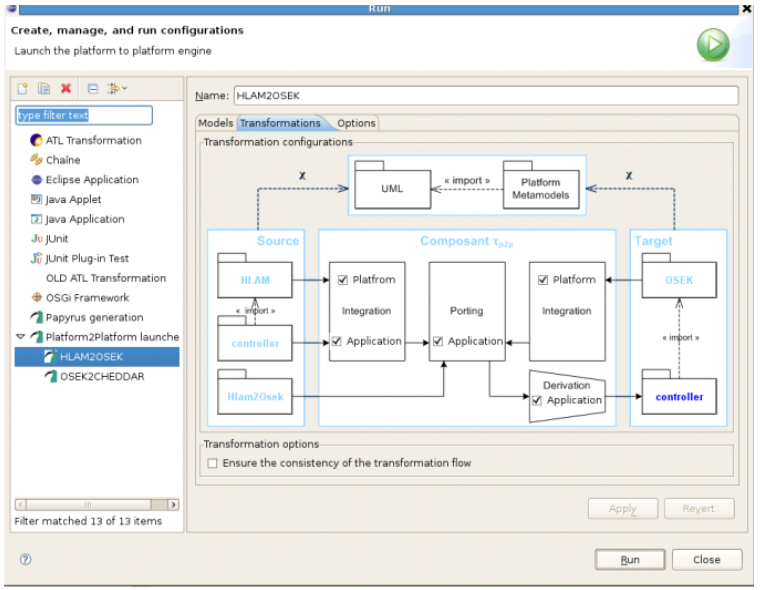




Réalisation de l'infrastructure de portage

- Modèles
  - Papyrus (www.papyrusuml.org)

- Infrastructure de transformation prototype
  - Codée en ATL (www.eclipse.org/m2m/atl)
  - Pas de génération de transformation
    - L'exécution de l'infrastructure est paramétrée par les modèles de plates-formes
  - Intégration à la plate-forme Eclipse



Evaluation de l'infrastructure

- Résultats

	Ressource			Propriété		
	Liens inférés	Liens totales	%	Liens inférés	Liens totales	%
HLAM2OSEK	5	7	71	12	15	80
OSEK2Cheddar	7	9	77	12	17	71
Total	12	16	**75**	24	32	**75**

- Capitalisation de l'infrastructure de transformation
  - Dépendante des hypothèses de modélisation de l'application (Classe singleton)
  - Dépendante des hypothèses de connexion : classe – instance
  - Dépendante des services outillant le motif « Resource-Service »
  - Indépendante de SRM
    - Utilisation conjointe de plusieurs métamodèles de plates-formes

Plan de la démarche

- Qu'est-ce qu'un modèle de plates-formes ?
  - Etat de l'art
- Qu'est-ce qu'un métamodèle de plates-formes ?
  - Définition du motif « Resource-Service »
  - Définition d'un outillage
- Comment capitaliser les transformations de portage ?
  - Définition d'une infrastructure de transformation
- Evaluation
  - Définition du métamodèle Software Resource Modeling
  - Réalisation d'une infrastructure de transformation
- **Conclusion**

Conclusion

- Contexte
  - Conception et implantation des applications concurrentes (multitâches)
  - Utilisation de l'ingénierie dirigée par les modèles pour faciliter l'intégration des outils et techniques de développement
- Problèmes
  - Les transformations portent les applications de plates-formes en plates-formes
  - Les plates-formes sont implicitement décrites dans les générateurs
  - Les générateurs sont des amalgames spécifiques de préoccupations
- Approche
  - Expliciter les plates-formes en entrée des générateurs
  - Définir des générateurs dirigés par les plates-formes

Conclusion

- Problèmes de recherche
  - Qu'est-ce qu'un modèle de plates-formes ?
    - Un modèle de son API (limitation au modélisation structurelle)
  - Qu'est-ce qu'un métamodèle de plates-formes ?
    - Un métamodèle appliquant le motif « Resource-Service »
    - Une illustration par le profil UML Software Resource Service (SRM)
  - Ces contributions sont intégrées au standard international OMG MARTE
  - Comment capitaliser les transformations de portage ?
    - Une infrastructure de transformation générique au motif « Resource-Service »
    - Une réalisation de cette infrastructure dans la plate-forme Eclipse
- Discussions
  - SRM a montré que le motif « Resource-Service » est un motif opérationnel
  - L'infrastructure de transformation a permis de capitaliser un noyau de portage

Perspectives

- Perspectives à court terme
  - Enrichir le profil UML SRM
    - tolérance aux fautes, initialisation des plates-formes
  - Enrichir l'architecture de transformation
    - Services et comportements

- Perspectives à long terme
  - Étudier la modélisation des comportements
  - Définir une méthodologie outillée basée sur des modèles de plates-formes explicites

Questions ?