



Un méta-modèle pour la représentation de patrons de conception à base d'aspects

O. Hachani

Ouafa.Hachani@imag.fr

LSR-IMAG, Grenoble

Equipe **SIGMA**

<http://www-lsr.imag.fr/sigma.html>



Motivations

★ Les nouvelles réalisations par aspects de patrons de conception par objets permettent de *tracer*, d'*adapter* et de *réutiliser* plus facilement chacune des instances des patrons imités

☆ Solutions semi-formelles et difficiles à appréhender : il appartient au programmeur de les référencer, les instancier et les imiter, ce qui est souvent source d'erreurs : *imitations invalides*

- Besoin d'un formalisme adéquat pour une meilleure application, réutilisation et traçabilité de patrons à base d'aspects depuis la phase de conception jusqu'à la phase d'implémentation



- Un cadre conceptuel de modélisation : Un méta-modèle pour la représentation structurelle des patrons prenant en compte les spécificités de la technologie

aspect

- I. Implémentation de patrons de conception par objets à l'aide d'AspectJ : exemple de réalisation du Visiteur***
- II. Un méta-modèle de patrons avec aspects***
- III. Conclusion et perspectives***



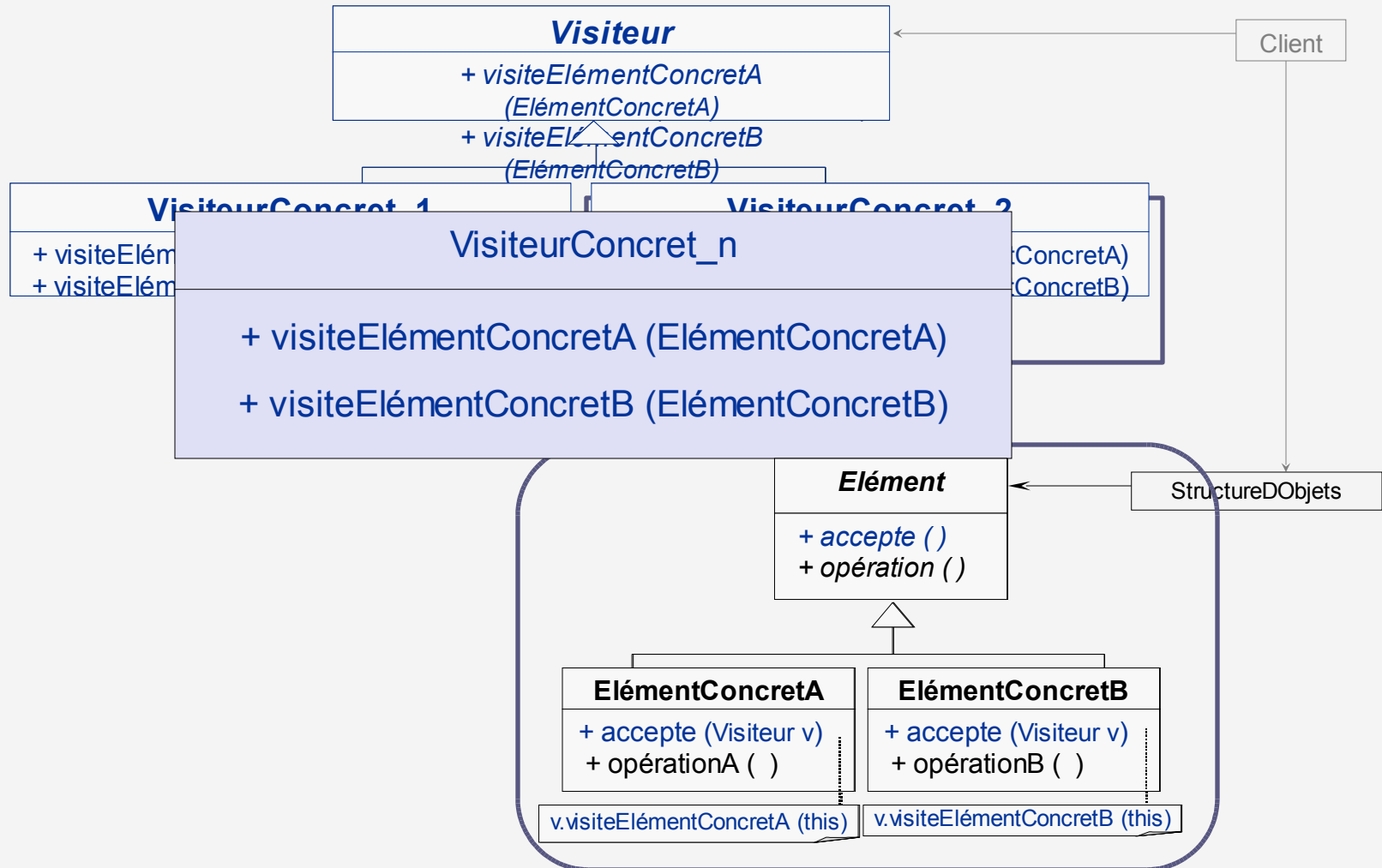
I

*Réalisation et description de patrons
de conception à l'aide d'AspectJ*

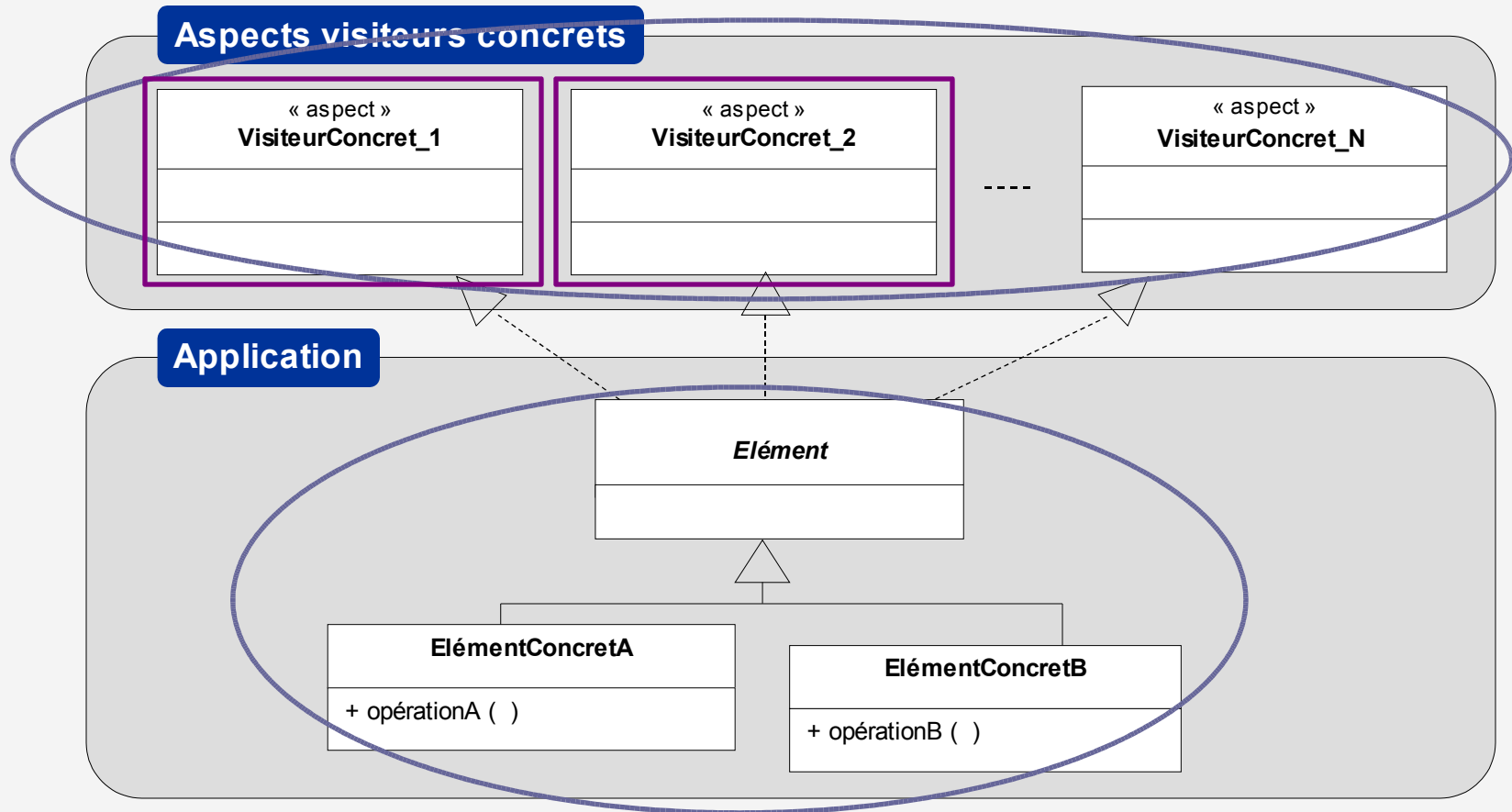
PPA et patrons de conception

- Plusieurs travaux récemment proposés traitent de l'implémentation de patrons de conception par objets à l'aide des langages de programmation par aspects (*AspectJ*, *HyperJ*, *AspectS*, etc.)
- Notre approche :
 - Caractérisation de quatre problèmes liés à l'utilisation de patrons de conception dans leur structure objet
 - De nouvelles implémentations de patrons du GoF à l'aide d'AspectJ inspirées essentiellement de l'intention des patrons
 - De nouvelles descriptions de patrons de conception dans une approche Aspect

Rappel de Visiteur



Patron Visiteur par aspects



Squelette du code d'un aspect VisiteurConcret

Chaque classe d'élément

```
privileged aspect VisiteurConcret_n
{
    possède une opération
    comportementAjouté( )

    // Introduction du comportement à ajouter dans la
    // classe abstraite Elément
    public abstract Object Elément.comportementAjouté( ) ;

    // Introduction du comportement à ajouter dans la
    // classe ElémentConcretA
    public Object ElémentConcretA.comportementAjouté( )
    { ... }

    // ... et ainsi de suite pour tous les Eléments concrets
}
Une implantation définie et
```

exécutée dans le contexte de la
classe d'élément elle-même

Vers de nouvelles descriptions des patrons

- Ne pas se contenter de nouvelles implantations possibles, travailler aussi sur leurs descriptions
- Rubriques *Intention* et *Indications d'utilisation* des patrons conservées
- Rubriques *Structure* et *Exemple de code* : réalisation par aspects du patron
- D'autres rubriques peuvent être modifiés : *Constituants*, *Collaboration*, *Implémentation*, *Conséquences*

II

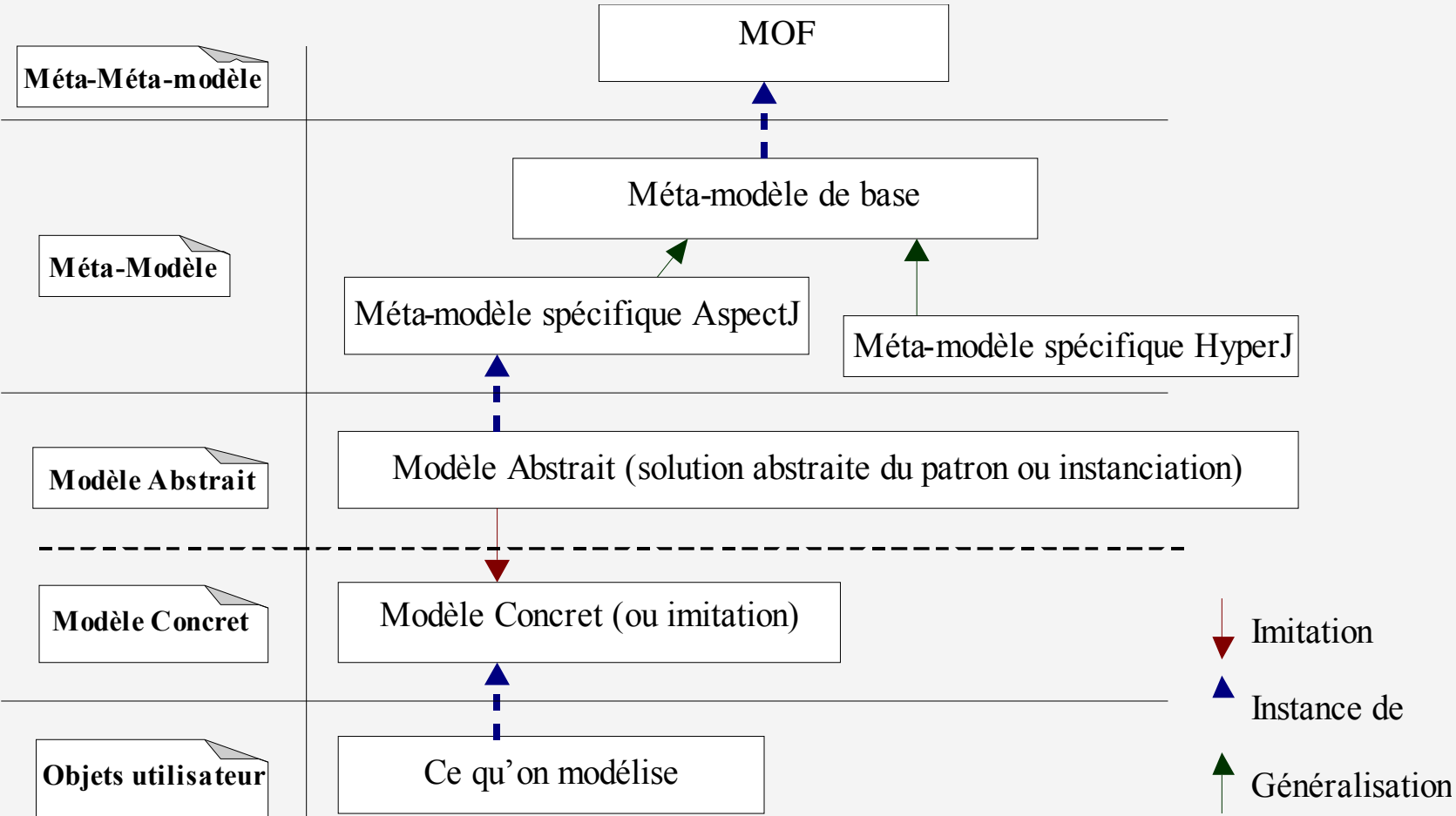
*Un méta-modèle de patrons
avec AspectJ*

Méta-modélisation et patrons de conception

- Abstraction et représentation des nouvelles structures par aspects des patrons à un niveau d'abstraction plus élevé par un méta-modèle intégrant tous les éléments de modélisation participant dans un patron
- Les mécanismes spécifiques aux langages de programmation par aspects sont plus au moins divergents
 - Méta-modèle de base générique
 - Spécialisation en autant de méta-modèles spécifiques que de langages de programmation par aspects considérés
- Méta-modèle spécialisé pour AspectJ

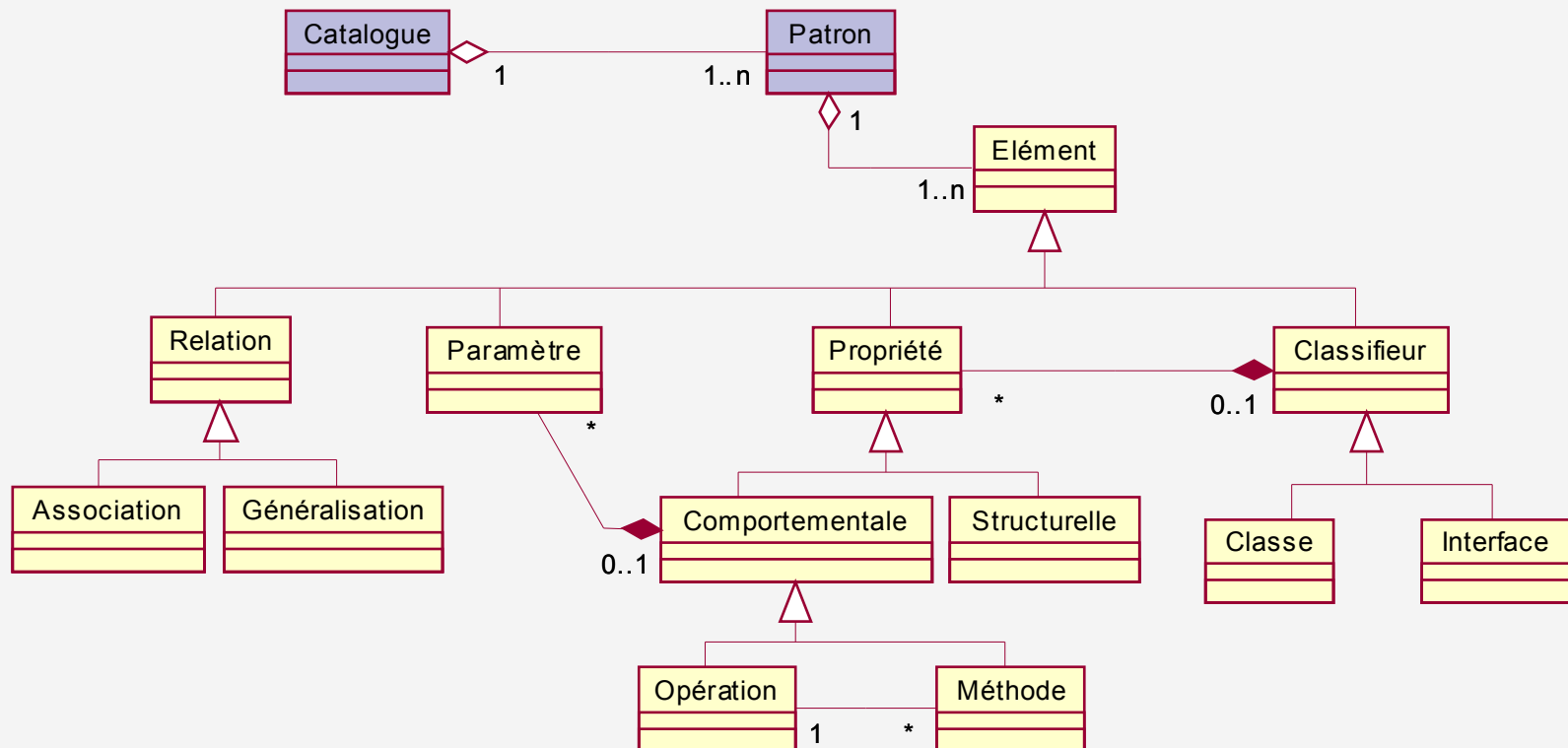
Organisation des méta-modèles

- Architecture à 4 niveau



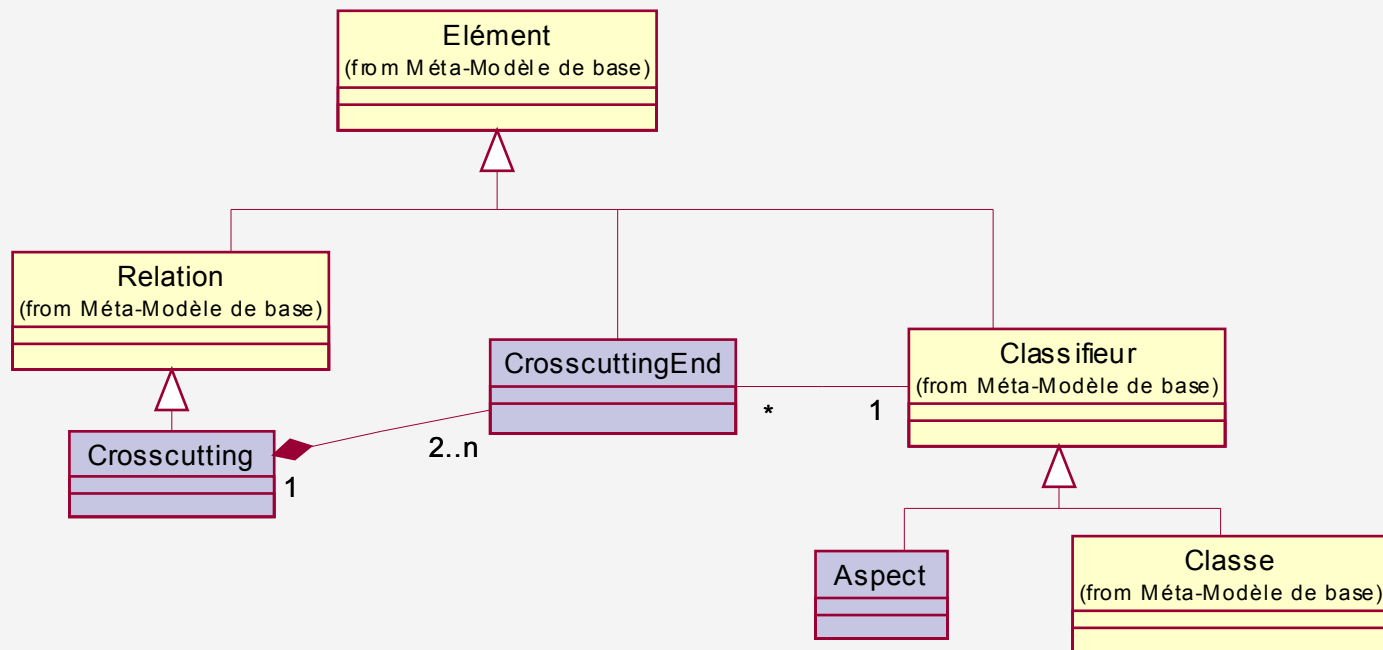
Méta-modèle de base

- Sous-ensemble simplifié du méta-modèle de base (extrait simplifié du méta modèle d'UML)



Méta-modèle spécialisé pour AspectJ

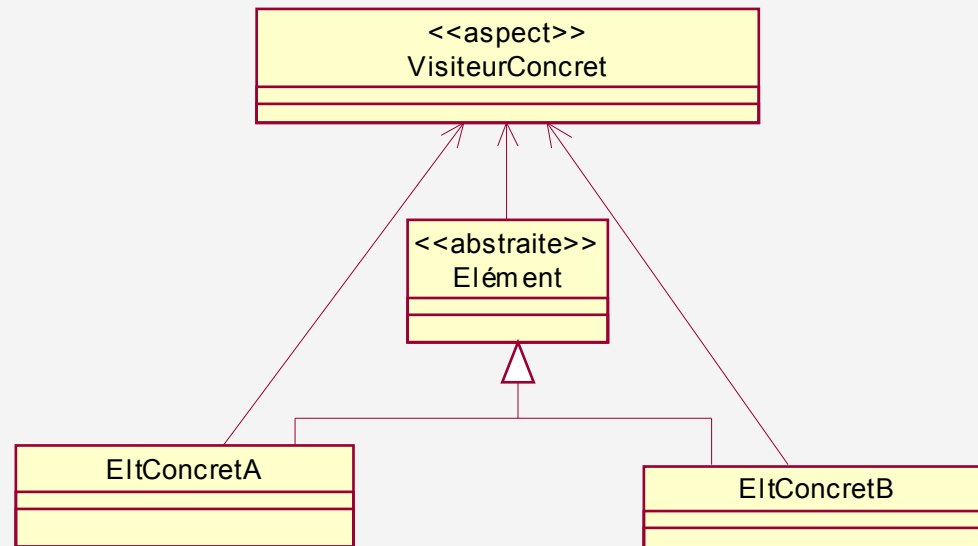
- Spécialisation du méta-modèle de base en rajoutant tous les éléments de modélisation spécifiques à AspectJ (aspects, introductions, pointcut, advice, etc.)



Exemple d'instanciation du patron Visiteur

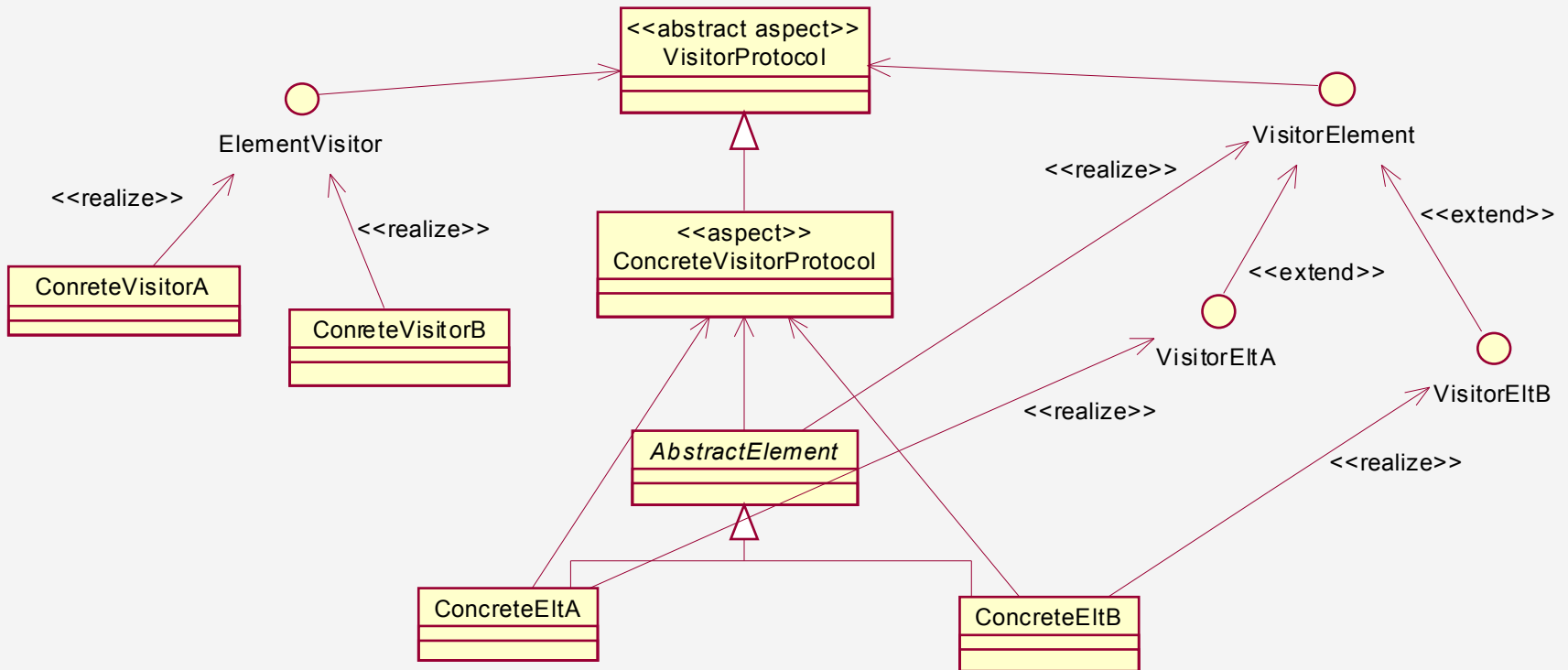
(1)



- Modèle abstrait du patron *Visiteur* en AspectJ : une première variante correspondante à notre approche



Exemple d'instanciation du patron Visiteur (2)

- Modèle abstrait du patron *Visiteur* en AspectJ : une deuxième variante relative aux travaux de Hannemann et Kiczales





III

Conclusion et Perspectives

- Une approche par méta-modélisation pour la représentation structurelle de patrons de conception par aspects
 - Proposition d'un méta-modèle de base générique
 - Un méta-modèle spécialisé pour AspectJ
- Présentation du schéma général d'utilisation et deux exemples d'instanciation du patron *Visiteur*
- **Réalisation de l'Objectif** : abstraction de tous les éléments nécessaires à la représentation structurelle de patrons à base d'AspectJ, dans le but d'en améliorer leur réutilisation lors de la conception

- Généralisation à différents langages de programmation par aspects : travailler d'avantage sur l'abstraction des éléments de modélisation spécifiques
- Généralisation de l'approche à d'autres patrons de conception spécifiques aux aspects
- Plusieurs applications possibles pour ce travail concernant divers besoins de modélisation : instrumentation de l'imitation, génération de code, détection, etc.

Méta-modèle spécifique à AspectJ

