La réutilisation dans le domaine de la coopération de systèmes d'information : une approche mixte patterns et frameworks

Vincent Couturier
Équipe MODEME, UMR CNRS 5055
Centre de recherche de l'IAE - Université Jean Moulin - Lyon 3
15, quai Claude Bernard – 69007 LYON
Tél. 04 78 78 71 58 - Fax. 04 78 78 77 50
vincent.couturier@univ-lyon3.fr

Workshop OCM-SI. Nancy. 3 juin 2003.

MODEME - UMR CNRS 5055

Objectif

- **Capitaliser, modéliser et réutiliser les connaissances mises en œuvre lors du développement de systèmes d'information coopératifs.**

- **A partir de ces connaissances, fournir une implantation réutilisable lors du développement d'applications coopérantes, afin de faciliter la conception de telles applications.**

Constat

- La coopération prend en compte de nombreuses dimensions : distribution, interactions, interfaces, protocoles, modèles de données et de connaissances, sémantique,…

- Le développement de systèmes coopératifs nécessite le recours à des technologies complexes : middleware, systèmes multi-agents, protocoles, ontologies,…

- La capitalisation des connaissances liées au développement de systèmes d'information coopératifs ne peut se réduire à la modélisation de celles-ci sous forme de schémas UML.

Des concepts permettant de capitaliser et réutiliser les connaissances : les patterns et frameworks

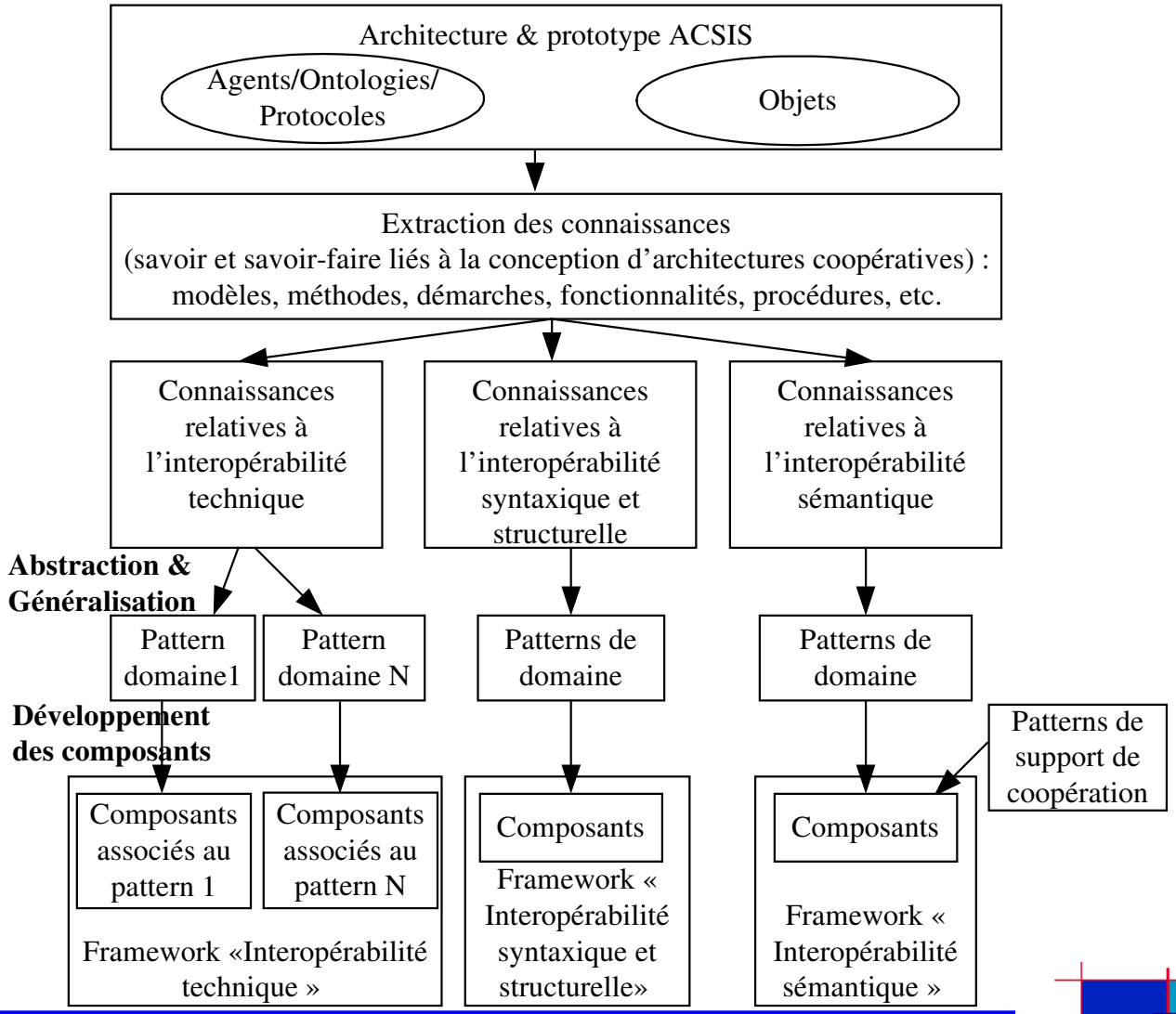
- **Pattern :**
  - **« abstractions de logiciels utilisées par des concepteurs et des programmeurs avancés dans leurs programmes » [COPL94]. L' idée de base des patterns est de capitaliser un savoir et un savoir-faire et de proposer une solution efficace et réutilisable à un problème donné.**

- **Framework :**
  - **Les frameworks sont, au même titre que les patterns, conçus par des experts pour répondre à un problème particulier et être utilisés et réutilisés par des "moins experts" [JONH92]. Ils permettent de faciliter le développement d' application, par utilisation systématique d' un cadre applicatif réutilisable.**

Quelle démarche ? Quel type de pattern ?

- **Trois types de démarche de conception des patterns et des frameworks possibles (généricité vs opérationnalité) :**
  - **Conception des patterns de coopération à partir de patterns de conception génériques existants ([SAID02])**
  - **Conception des patterns de coopération à partir des architectures coopératives existantes.**
  - **Conception des patterns de coopération à partir d'une architecture donnée ([FERN00], [GANG01]).**
- **Quel type de pattern :**
  - **Nos patterns de coopération sont dérivés des patterns de domaine ([FOWL97]) :**
    - **Restriction à un domaine particulier**
    - **Patterns génératifs**
    - **Patterns de support**

Démarche de conception des patterns et des frameworks de coopération



Les patterns de coopération 1/2

θ Les patterns de domaine de coopération

- **dérivés des patterns de domaine de Fowler.**
- **patterns génériques et génératifs : spécifient des composants logiciels réutilisables et le savoir-faire associé en utilisant la structure du pattern comme documentation de ces composants.**
- **permettent de modéliser les problèmes génériques et les principes fondamentaux du domaine de la coopération, mais aussi de générer les composants réutilisables associés (frameworks).**

Les patterns de coopération 2/2

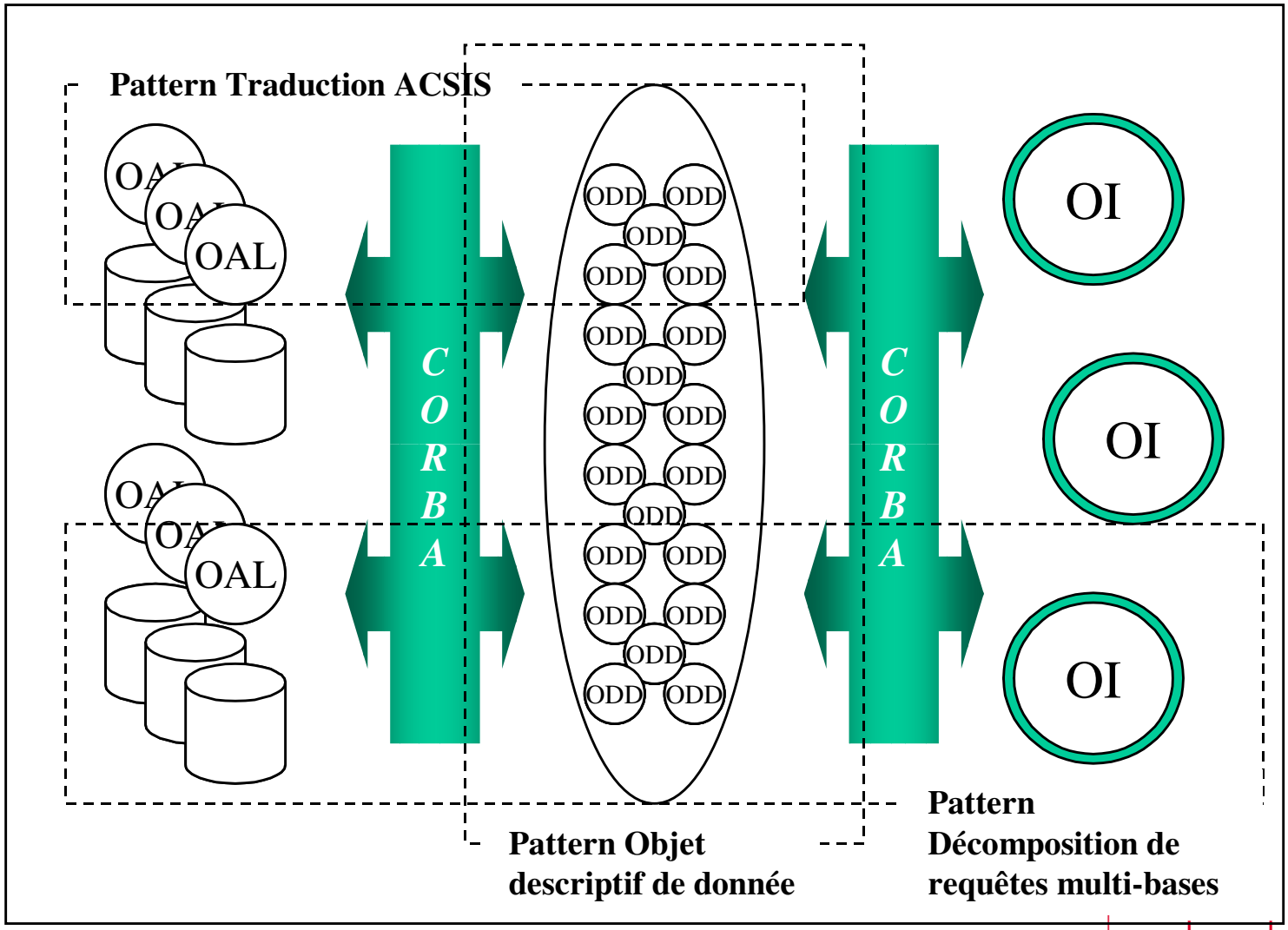
θ Les patterns de support de coopération

- **dérivés des patterns de support de Fowler.**
- **décrivent comment utiliser les patterns de domaine de coopération et comment les assembler (notamment leur partie solution).**
- **génèrent des composants techniques (transformation de modèles) qui vont implémenter les connaissances modélisées par les patterns de domaine de coopération dans l'environnement technique choisi par le programmeur.**
- **patterns originaux ou patterns dérivés des patterns de support de Fowler.**
- **Utilisent des patterns génériques de conception et d'architecture mais aussi des patterns de conception et d'implémentation liés aux plates-formes de développement et langages de programmation (patterns de conception CORBA, idiomes Java,…).**

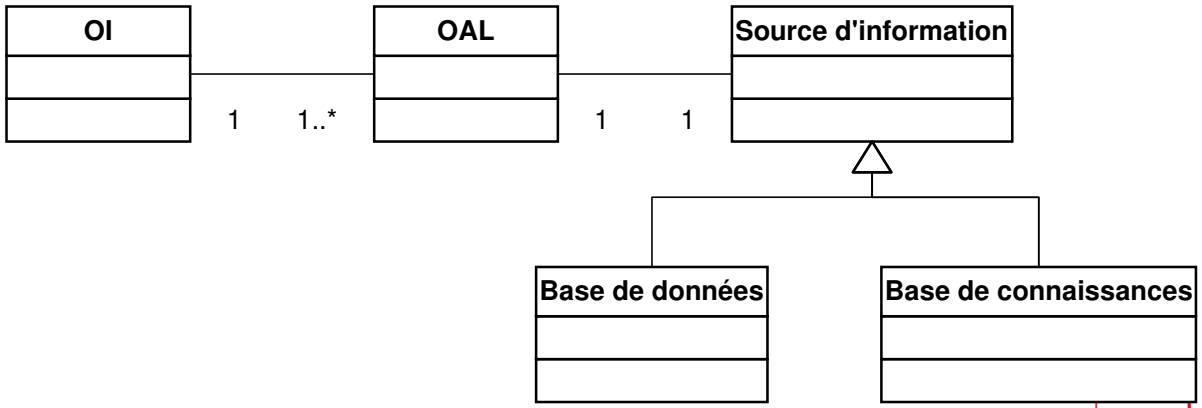
Les frameworks de coopération

- Frameworks verticaux (domaine).
- Chaque framework permet de résoudre un problème de coopération bien défini :
  - framework dédié à la résolution de l'hétérogénéité technique des sources d'information,
  - framework dédié à la résolution des hétérogénéités syntaxiques et structurelles
  - framework dédié à la résolution des conflits sémantiques.
- Frameworks « boîte noire ».
- Chaque framework est associé aux différents patterns qui ont permis de le construire et qui le documentent.

Exemple de patterns : un langage de patterns dédié à la résolution des conflits syntaxiques 1/7



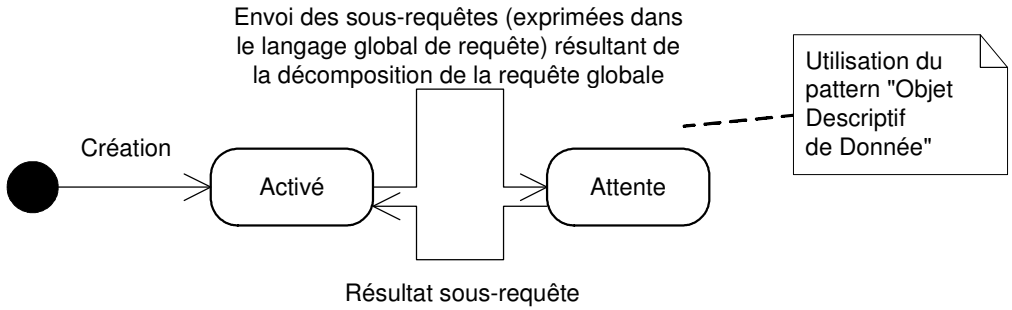
Exemple de patterns : un langage de patterns dédié à la résolution des conflits syntaxiques 2/7

Interface
Nom : Décomposition de requêtes multi-bases
Classification : Résolution de l'hétérogénéité syntaxique
Objectif : L'objectif de ce pattern est de décomposer les requêtes multi-bases en requêtes locales et de les transmettre aux sources d'information pour traitement. Lors de cette décomposition, les conflits syntaxiques sont résolus.
Applicabilité : Pattern génératif utilisable lors des phases de conception et d'implantation (Rubrique « solution »). Ce pattern nécessite l'application des patterns « Traduction ACSIS » et « Objet Descriptif de Donnée ». Il est applicable lors de la coopération de sources d'information de type bases de données ou bases de connaissances.
Force :Fournit un mécanisme de décomposition des requêtes soumises au système coopératif en requêtes locales.
Solution
Modèle : 

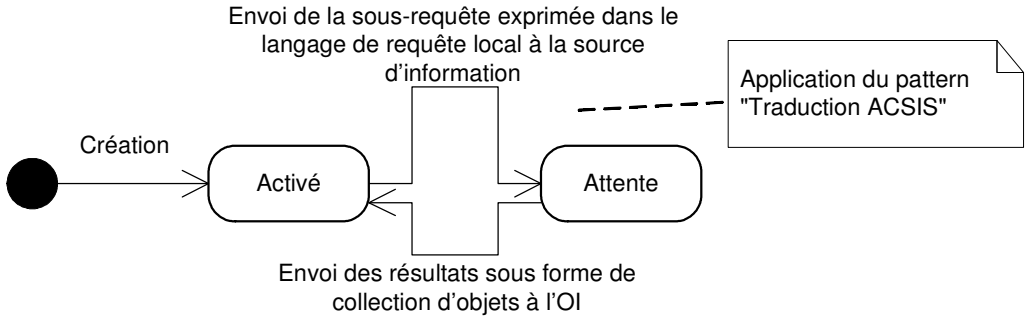
Exemple de patterns : un langage de patterns dédié à la résolution des conflits syntaxiques 3/7

Participants :

L'Objet Informationnel (OI) : La principale tâche de l'OI consiste à décomposer la requête globale soumise par un acteur externe et, après décomposition, d'envoyer les sous-requêtes (exprimées dans le langage de requête global) aux OAL impliqués. Lorsque l'OI reçoit une requête globale de la part d'un acteur externe, il réagit par la décomposition de cette requête globale en sous-requêtes grâce à l'utilisation du pattern « Objet Descriptif de Donnée » et les expédie aux OAL concernés.

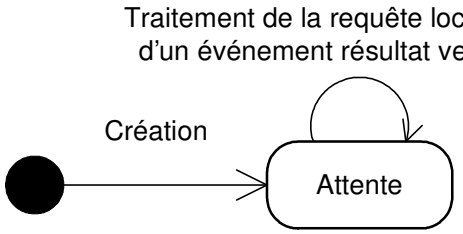


OAL : Chaque OAL se charge de la traduction de la sous-requête dont il a la charge en requête locale et de regrouper les résultats sous forme de collection d'objets.



Exemple de patterns : un langage de patterns dédié à la résolution des conflits syntaxiques 4/7

Source d'information : Classe capable de recevoir des requêtes, de les traiter et de retourner les résultats. Une source d'information peut être de deux types : base de données ou base de connaissances. Elle retourne les résultats de la requête locale transmise par l'OAL.



Collaboration

Un Objet Informationnel reçoit d' un acteur externe une requête globale. L'OI décompose la requête globale en sous-requêtes (exprimées dans le langage de requête global) et les envoie aux OAL concernés. Les OAL réceptionnent ces requêtes et, après traduction dans les langages de requête cibles, les transmettent aux sources d'information et attendent les résultats. Chaque OAL récupère les résultats, les formate sous forme de collection d'objets et les transmet à l'objet informationnel qui restructure l'ensemble des résultats.

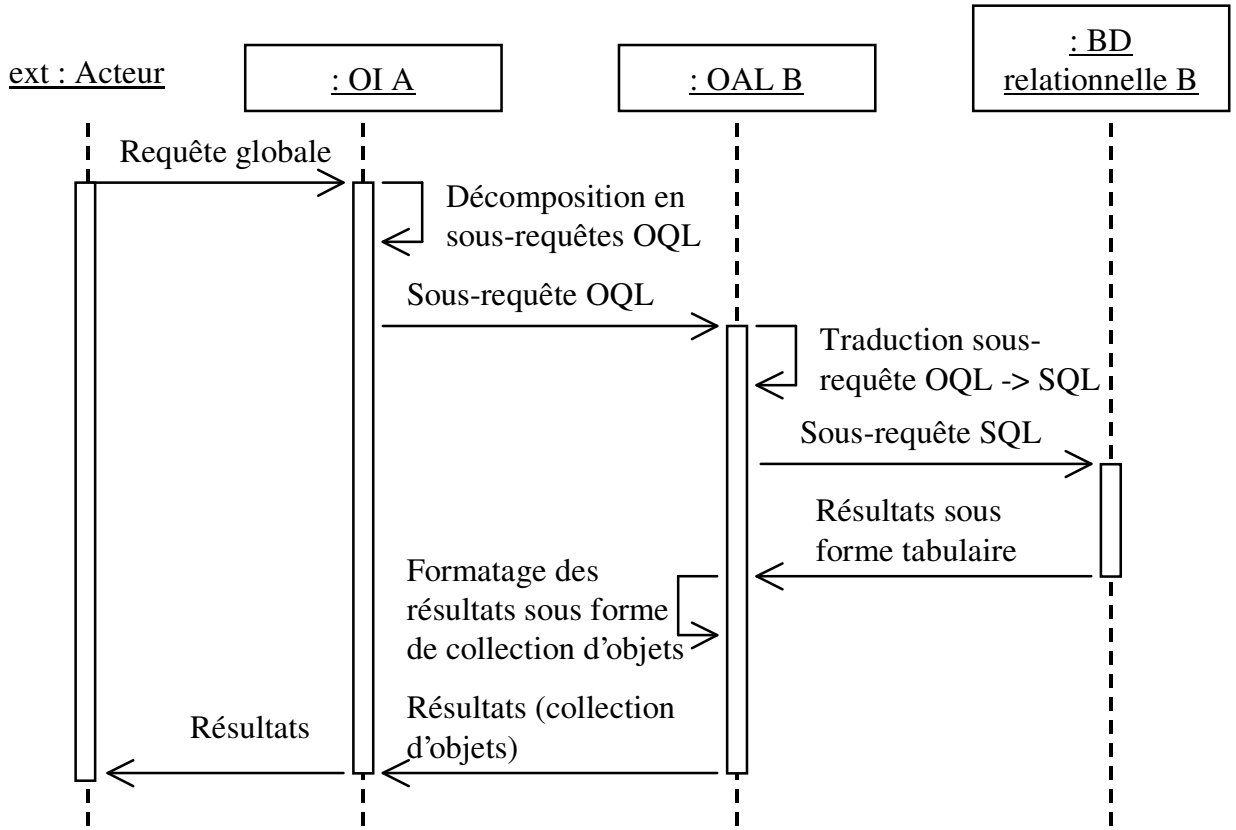
Implémentation

```
interface OAL {
  void SendQuery (in string Query, out boolean Status)
                raises (UserException);
  void GetAnswer (out DBresult Result, out short
Status)        raises (UserException);
  void DeleteObj() raises (UserException);  };
```

```
interface OI {
  void GlobalQuery (in string Query, out DBresult
Result, out short QueryStat, out short Resubmit)
                raises (UserException);
};
```

Exemple de patterns : un langage de patterns dédié à la résolution des conflits syntaxiques 5/7

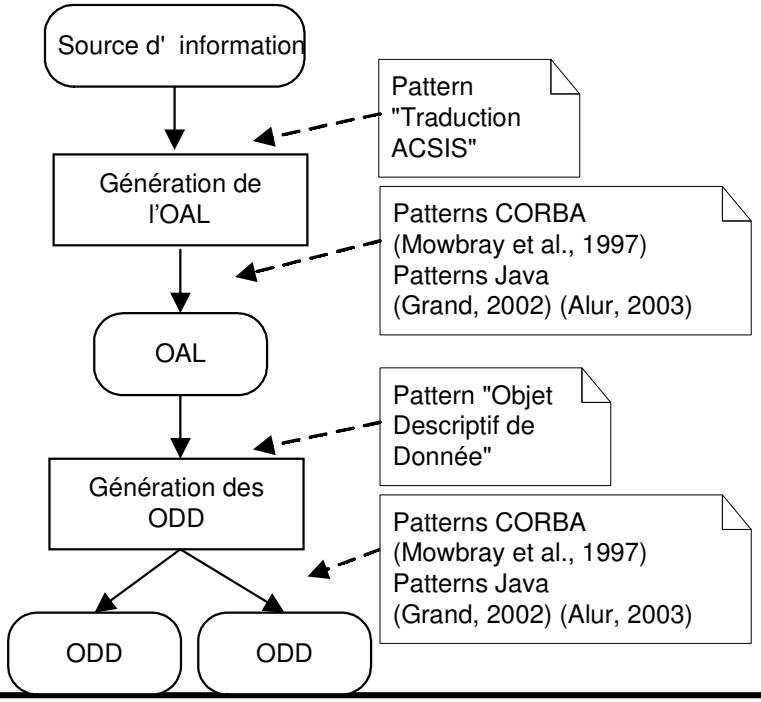
Exemples d'utilisation



Relations

Utilise : {Traduction ACSIS, Objet Descriptif de Donnée, Factory (Gamma, 1995)}

Exemple de patterns : un langage de patterns dédié à la résolution des conflits syntaxiques 6/7

Interface
Nom : Génération des ODD
Classification : Support ^ Résolution de l'hétérogénéité syntaxique
Objectif : Générer les Objets Descriptifs de Données à partir des sources d'information participant au système coopératif
Solution
Démarche : 

Exemple de patterns : un langage de patterns dédié à la résolution des conflits syntaxiques 7/7

Relations
Utilise : {Traduction ACSIS, Objet Descriptif de Donnee, Factory}

Bilan & perspectives

- **Proposition d'une définition des concepts de pattern et de framework de coopération :**
  - **Concepts basés sur l'exploitation d' un domaine d' application particulier.**
  - **Conceptualisation de l'expertise liée à ce domaine sous forme de patterns et automatisation de cette expertise sous forme de composants intégrés au sein de frameworks.**
- **Proposition d'une démarche de construction de ces patterns et frameworks.**
- **A terme, des bibliothèques de patterns, intégrées dans des ateliers de génie logiciels, apporteront un réel cadre méthodologique à la conception de systèmes d'information coopératifs et distribués.**

Références

- **Coplien J.O., Software Design Patterns: Common Questions and Answers, *Object Expo Conference Proceddings*, SIGS, 1994.**
- **Johnson R.E., Documenting Frameworks Using Patterns, *in Proc. of OOPSLA*, Vancouver, Canada, 1992.**
- **Fowler M., *Analysis Patterns*, Addison Wesley, 1997.**
- **Saidane M., Giraudin J.P., Rieu D., Des patrons pour l'ingénierie de la coopération des systèmes d'information, *Congrès Inforsid*, France, 2002.**
- **Fernandez G., Zhao L., A Pattern Language for a Federated Architecture, *In Proceedings of KoalaPloP 2000*, Melbourne, Australia, May 2000.**
- **Ganguly P., Rabhi F.A., Ray P.K., The Semantic Interoperability Pattern, *In Proc. The Second Asian-Pacific Pattern Languages of Programming Conference (KoalaPLoP' 2001),* May 2001.**