

Systemes d'intégration des données :

Une approche à composants

17 mars 2004

Auteurs : Mourad ALIA
Christine COLLET
Alexandre LEFEBVRE

*France Télécom R&D DTL/ASR Meylan
En collaboration avec LSR-IMAG Grenoble*

Le présent document contient des informations qui sont la propriété de France Télécom. L'acceptation de ce document par son destinataire implique, de la part de ce dernier, la reconnaissance du caractère confidentiel de son contenu et l'engagement de n'en faire aucune reproduction, aucune transmission à des tiers, aucune divulgation et aucune utilisation commerciale sans l'accord préalable écrit de France Télécom R&D

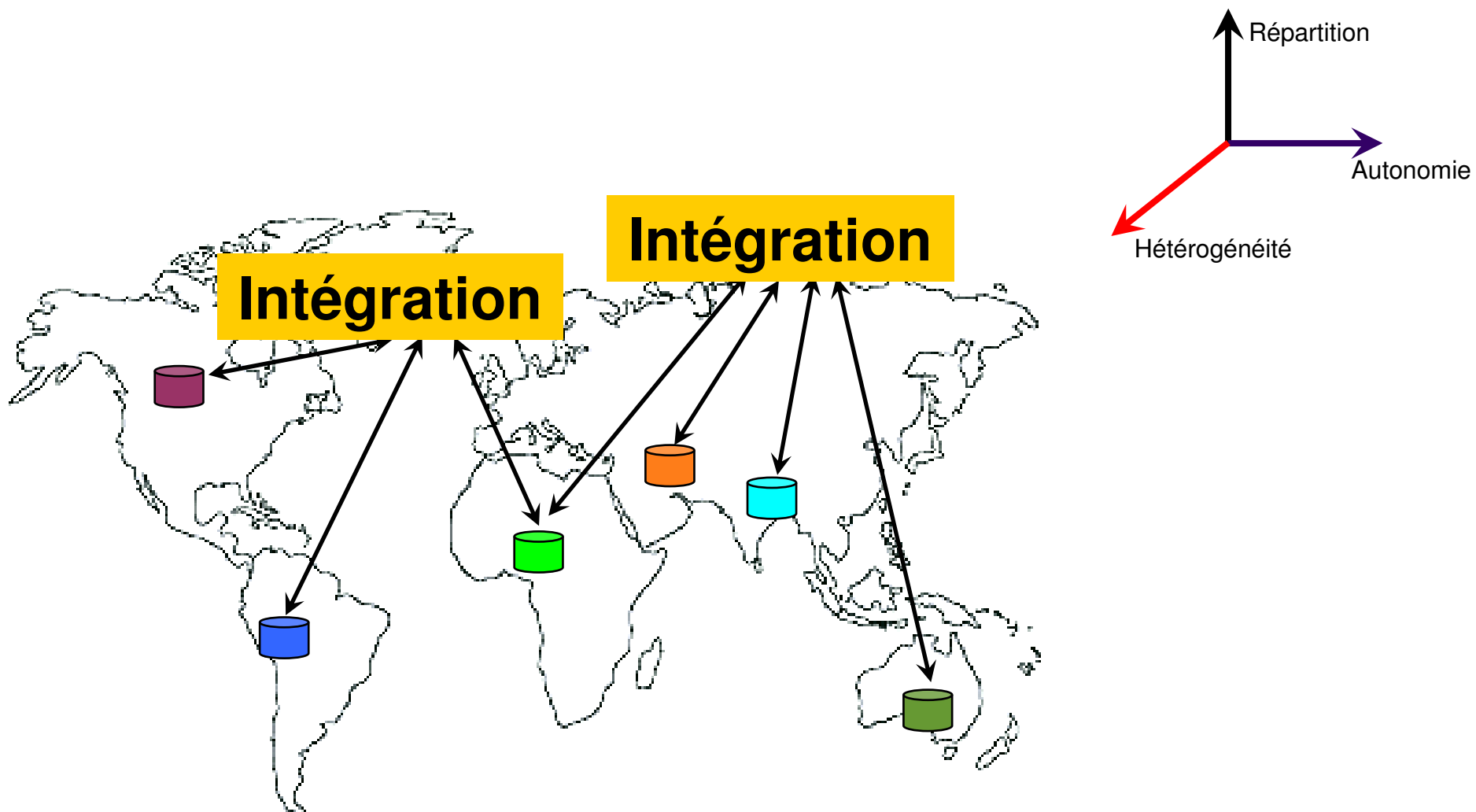


Plan

- **Introduction**
 - Architectures Existantes
 - Problèmes
- **Notre approche**
 - Un Système Adaptable
 - Modèle Architectural à Composants
- **Les canevas**
- **Conclusion et Perspectives**



Systemes d'intégration des données





Architectures des systèmes d'intégration de données

→ Trois grandes familles

- ⑦ Bases de données fédérées / multi-bases
- ⑦ Architecture orientée services
- ⑦ Architecture de médiation

→ Analyse

Bases de données fédérées et multi-bases



→ Bases de données fédérées

- ⑦ Schéma intégré sur plusieurs bases de données
- ⑦ Intégration « forte »

→ Multi-bases

- ⑦ Une certaine autonomie au niveau de chaque BD
- ⑦ Plusieurs points d'accès aux sources
 - Pas une seule vue intégrée



Architecture orientée services

→ **Intégration de l'information par un ensemble de services**

→ **Exemples**

⑦ **Architecture I3 - Intelligent Information Intégration [ARPA 96]**

- **Des familles de services : coordination et management, intégration sémantique et transformation, wrapping ...**

⑦ **SIM-TBASSCO (Semantic Interoperability Measures: Template-Based Assurance of Semantic Interoperability in Software Composition)**

- **[University of Southern California's Information Sciences Institute (ISI)]**

→ **Avantages**

⑦ **Modularité, ajout/suppression de services, templates de services ...**

→ **Inconvénients**

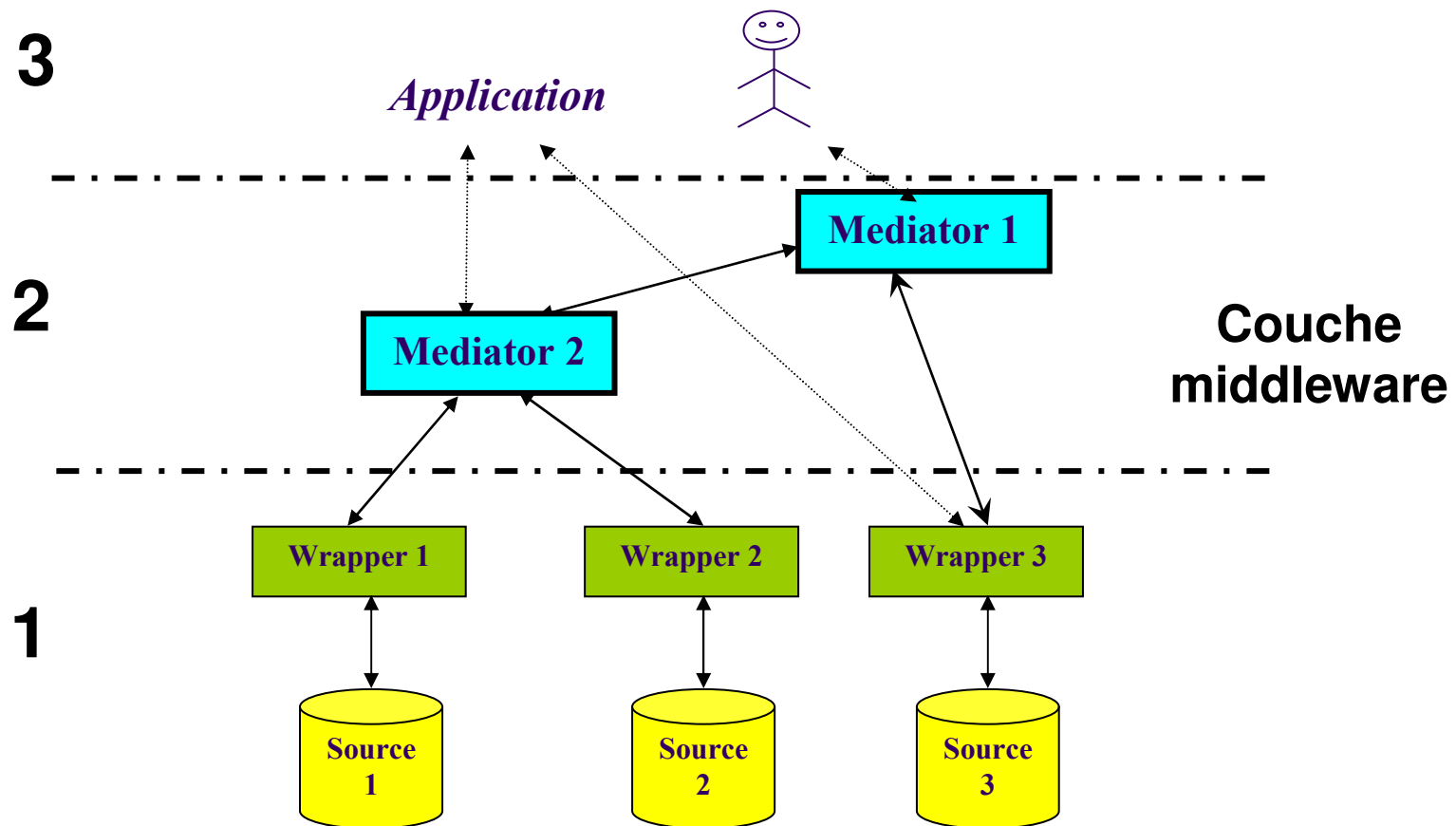
⑦ **Contrôle difficile de tout le système**

⑦ **Pauvre interopérabilité entre services (langages ou protocoles imposés)**

⑦ **Traitement des requêtes peu pris en compte**

Architecture de médiation

→ Modularité à trois niveaux





Problèmes des systèmes d'intégration des données

→ Approche « boîte noire »

- ⑦ Les systèmes existants se présentent comme des bibliothèques de codes !!
- ⑦ Manque d'interopérabilité
 - Interactions limitées (approches langage ou protocolaires)
- ⑦ Difficulté d'intégration d'autres services
 - Sécurité, ...

→ Construction ad hoc des systèmes existants

- ⑦ Déploiement et évolution non pris en compte

→ Impossibilité d'avoir un seul système générique

- ⑦ Prise en compte des spécificités des domaines d'application
- ⑦ Le système impose un langage de requêtes

 **Besoin d'architecture ouverte et adaptable pour un middleware d'intégration des données**



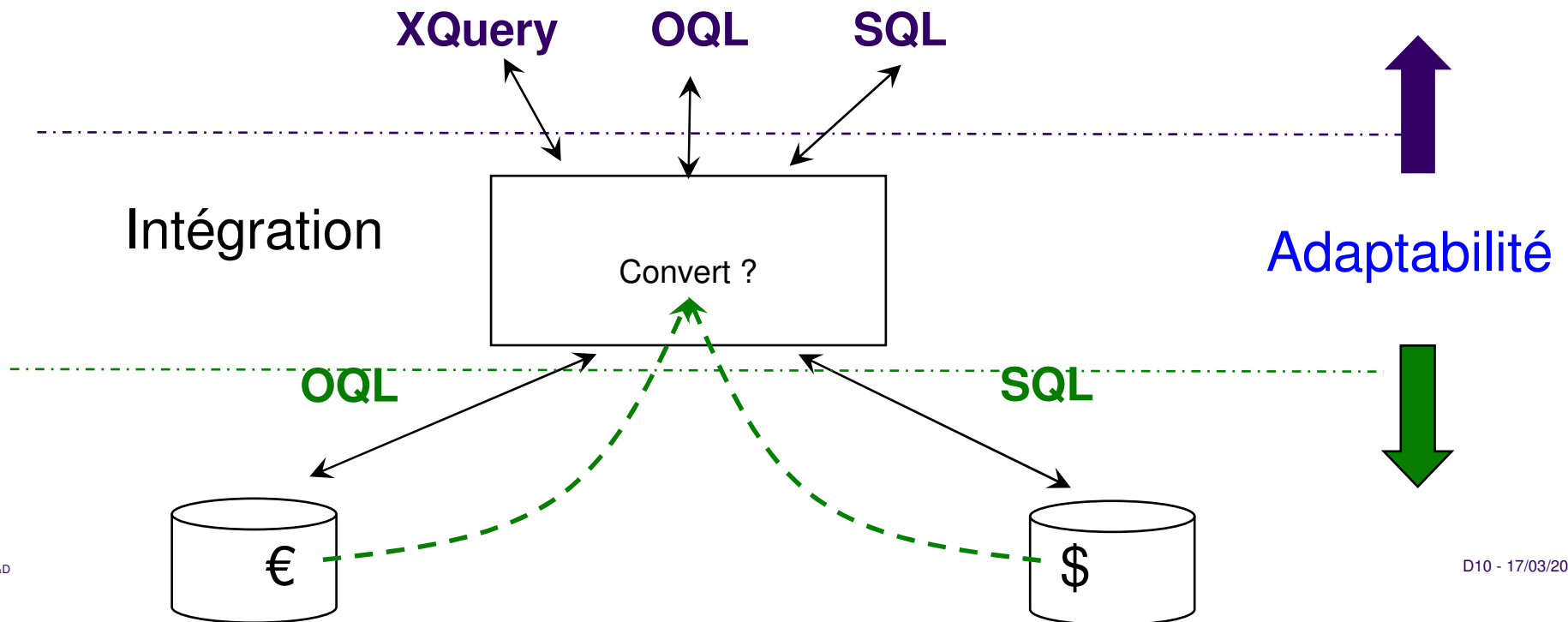
Plan

- Introduction
 - Architectures Existantes
 - Problèmes
- Notre approche
 - Un Système Adaptable
 - Modèle Architectural à Composant
- Les canevas
- Conclusion et perspective

Adaptabilité (1/2)

→ Adaptabilité vers le haut et vers le bas

- ⑦ Adapter le système aux besoins de l'application dans un domaine particulier
- ⑦ Adapter le système à l'hétérogénéité des sources de données





Adaptabilité (2/2)

A trois niveaux

- ⑦ Au niveau langage de requêtes
 - Possibilité d'interrogation par différents langages (indépendance)

- ⑦ Au niveau des données
 - Prise en compte de nouveaux types de données

- ⑦ Au niveau fonctionnel et calcul
 - Ajout d'agrégats, fonctions de calcul, etc.



Notre approche

→ Approche minimale

- ⑦ Gestion de l'accès aux sources
- ⑦ Prise en compte de l'hétérogénéité des données
- ⑦ Traitement efficace des requêtes

→ Approche extensible

- ⑦ Indépendance vis-à-vis des langages de requêtes
- ⑦ Cache, rajout de nouvelles fonctionnalités, nouveaux algorithmes d'évaluation

→ Approche architecturale à composants

- ⑦ Construction des systèmes par composition
- ⑦ Facilite le contrôle, le déploiement, la réutilisabilité

→ Approche canevas logiciel

- ⑦ Points de flexibilité et d'extension



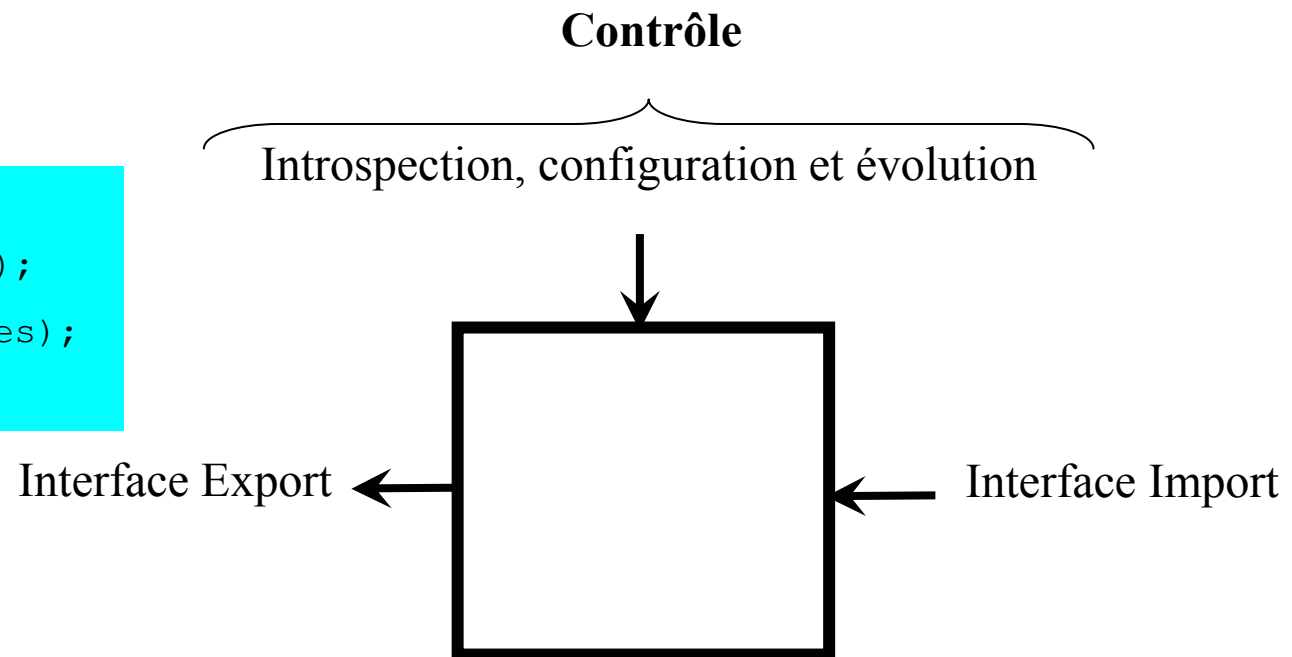
Modèle (1/2)

→ Basé sur Fractal

- ⑦ Des composants « computationnels » aux composants « informationnels »

→ Domaine de données

```
View getView();  
Collection evaluate(HashMap hints );  
void optimize(RuleConfigurator rules);  
...
```



Le modèle de composants Fractal

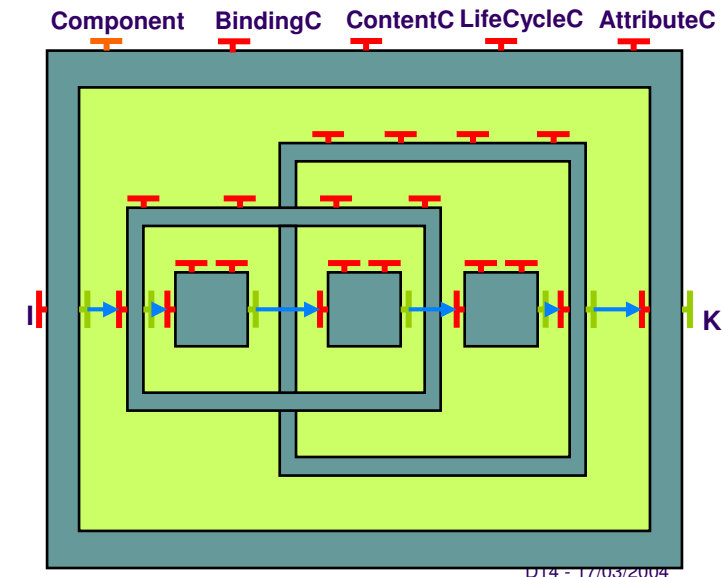


→ Modèle

- ⑦ **Composant = contrôleur [contenu]**
- ⑦ **Interface**
 - Seuls point d'accès à un composant par invocations d'opérations
- ⑦ **Liaison**
 - Canal de communication local entre deux composants
 - Composants pour les liaisons réparties ("connecteurs")

→ Principes architecturaux

- ⑦ **Séparation entre interfaces et implantations**
- ⑦ **Liaisons manipulables programmatiquement (et non « noyées » dans le code)**
- ⑦ **Récurtivité (*auto-similarité*)**
 - Interface cliente/Interface server
 - Composants de liaisons pour liaisons distantes, sécurisées...
- ⑦ **Contrôle arbitraire du contenu au runtime**
 - Introspection (réification de la topologie),
 - Contrôles prédéfinis : liaisons, contenu, attributs, cycle de vie





Modèle (2/2)

→ **Domaine de données = Composant**

- ⑦ Regroupe un ensemble de vues structurelles (collection d'objets)
- ⑦ Traitement des requêtes
- ⑦ N'impose pas de langage de requêtes ni de schéma intégré

→ **Unité d'intégration : vue structurelle (= requête)**

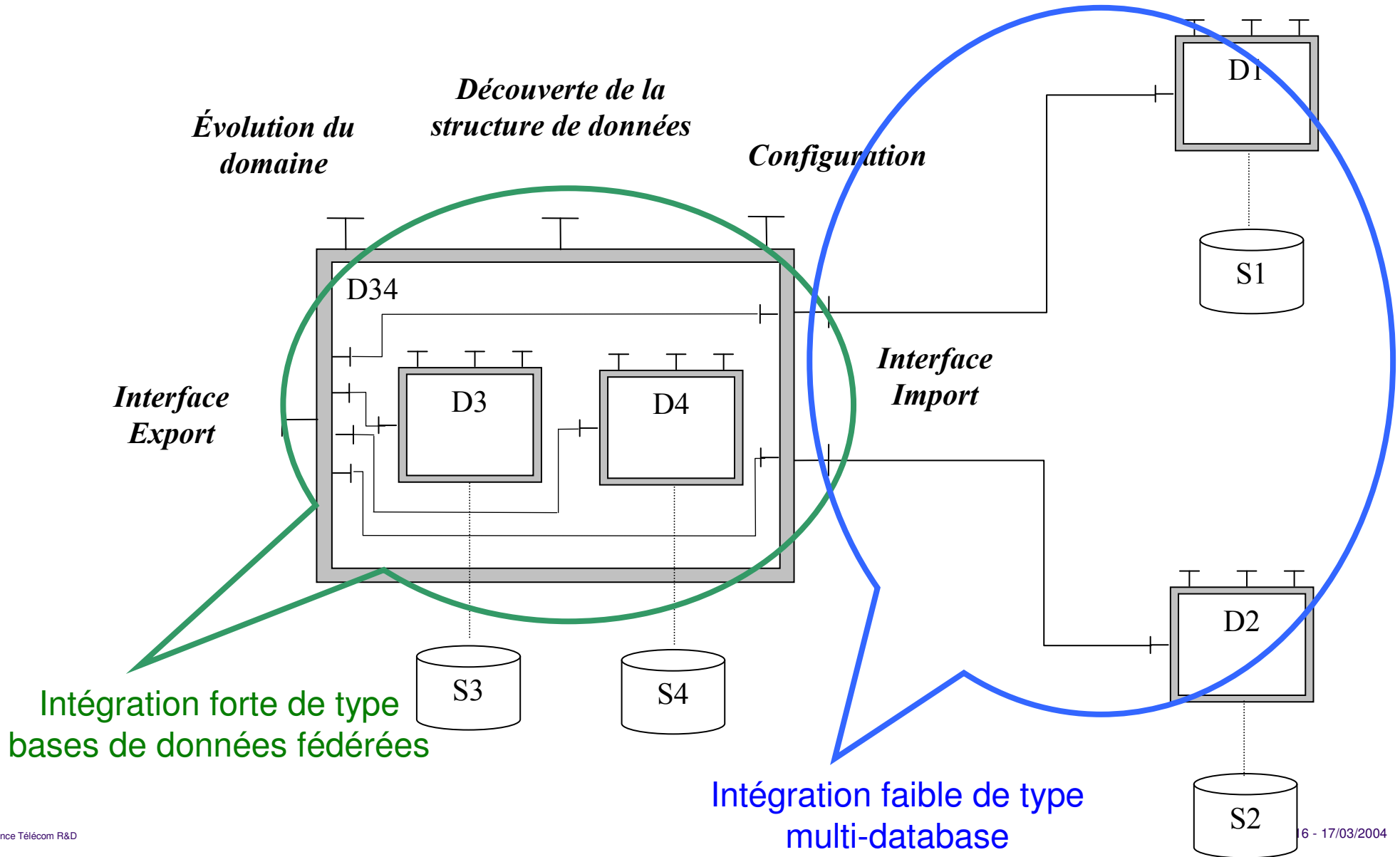
- ⑦ Partage de données entre domaines
- ⑦ Transformation des données

→ **Interopérabilité entre les domaines de données**

- ⑦ Désignation des interactions de base (Import/Export)
- ⑦ Composition pour le partage entre les domaines de données
 - Intégration de données par composition de domaines



Un exemple



Formes et granularités des domaines de données



→ Des domaines de données primitifs

- ⑦ Directement rattaché à une source de données : Wrapper
- ⑦ Domaines primitifs (1 table dans 1 SGBDR)

→ Des domaines de données semi-structurées Vs domaine de données structurées

- ⑦ Attention aux performances !

→ Des domaines de données légers Vs Domaine de données lourds

- ⑦ Nombre de sources de données, nombre de liaisons, ...
 - Un domaine de données peut se résumer à un seul opérateur !

→ Des domaines qui reformatent le résultat

→ Des domaines qui renomment les éléments (langue particulière)

- ⑦ Composition des services information



Modèle de données

→ Pour la prise en compte

- ⑦ De l'hétérogénéité des données
- ⑦ Des références
- ⑦ D'échange des données entre domaines

→ Modèle structurel objet

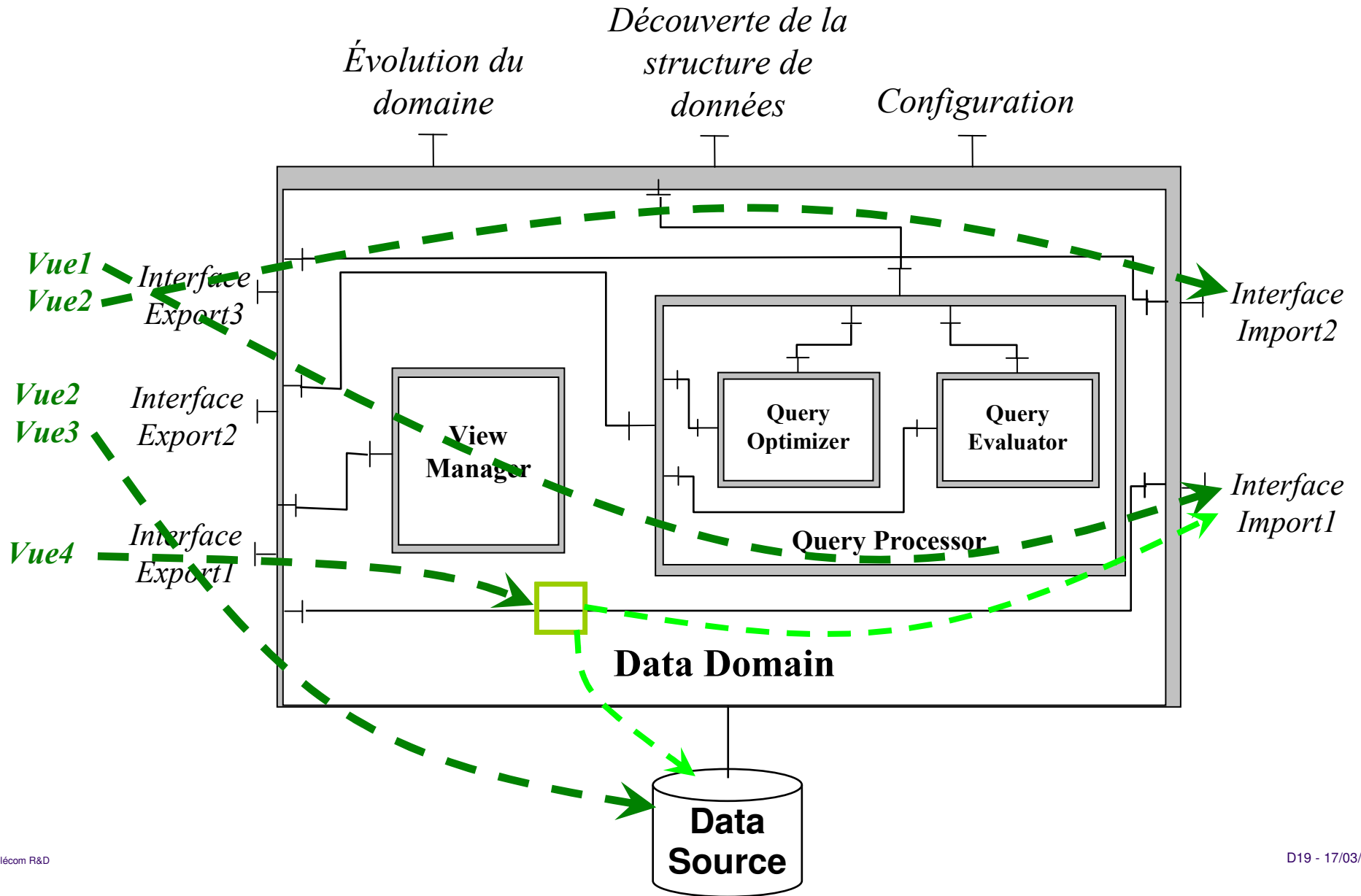
- ⑦ Identification et références

→ Deux types de collections d'objets : structurés / non structurés

- ⑦ Objets structurés : TupleCollection (~~ResultSet)
 - ⑦ Objets non structurés : TreeList (OEM)
 - ⑦ Prise en compte de la structure si elle existe, sinon découverte
 - ⑦ Une référence doit indiquer si l'objet référencé est structuré ou non
-
- ⑦ Projeté sur le langage de programmation Ex: Java



Architecture d'un domaine de données





Plan

- Introduction
 - Architectures Existantes
 - Problèmes
- Notre approche
 - Un Système Adaptable
 - Modèle Architectural à Composant
- Les canevas
- Conclusion et perspective



Les canevas

→ **Des points d'extension et de flexibilité pour la construction du système**

❖ **Canevas de nommage**

❖ **Canevas d'expression de requêtes**

❖ **Canevas de traitement des requêtes**



Canevas de nommage

→ Identification des espaces d'objets à travers des expressions de chemins

⑦ Résolution des noms (Matching)

→ Pas de langage d'expression de chemins

⑦ D'une manière programmatique sous forme d'un arbre d'opérateurs

- Opérateur de navigation, opérateurs d'expression régulières, prédicats sur les types ... (extensibilité pour une personnalité XPath !)

→ Exemple

⑦ Récupérer toutes les infos sur l'adresse des personnes de type string ?

`personne . * . Adresse . ?(type=string)`



Canevas d'expression des requêtes

→ Indépendance des langages de requêtes

- ⑦ Représentation canonique des requêtes
- ⑦ Requêtes paramétrables

→ Construction des requêtes d'une manière programmatique

- ⑦ Basée sur une algèbre
- ⑦ Les nœuds représentent des opérations et les feuilles, des vues

→ Capacité d'expression de sémantique

- ⑦ Renommage des éléments
- ⑦ Changement/conversion des types des éléments
- ⑦ Transformation des données



Canevas de traitement de requêtes

→ Optimisation

⑦ Globale/locale

- Répartition de l'évaluation entre domaines de données

⑦ Réécriture des requêtes en fonction des vues (MiniCon)

- Délégation de l'évaluation aux sources si possible

⑦ Query/data/hybrid/code shipping

→ Evaluation

⑦ Itérative (pipe line)

⑦ Row blocking

⑦ Parallélisme

- Intra domaine / inter domaines



Plan

- Introduction
 - Architectures Existantes
 - Problèmes
- Notre approche
 - Un Système Adaptable
 - Modèle Architectural
- Les canevas
- Conclusion et perspectives



Conclusion

- ➔ **Analyse des problèmes liés à l'architecture des systèmes d'intégration de données**
- ➔ **Définition des points d'adaptabilité et d'extensibilité**
- ➔ **Proposition d'une architecture à composants, basée sur le modèle Fractal**
- ➔ **Canevas associés pour le nommage, l'expression et le traitement des requêtes**

- ➔ **Première validation dans la cadre du consortium ObjectWeb :**
 - ⑦ **MEDOR : Implantation des canevas d'expression et de traitement de requêtes (en centralisé)**
 - ⑦ **Utilisé dans JOnAS (EJB), Speedo (JDO)**



Perspectives

- Continuer l'implantation
- Définir des critères de choix des architectures
- Développer une bibliothèque de domaines de données spécialisés



L'architecture *logicielle* d'après Matisse et Picasso

C'est en rentrant dans l'objet qu'on rentre dans sa propre peau...

Respecter l'objet !

Tout doit être construit... composé de parties qui forment un tout. Un arbre comme un corps humain, un corps humain comme une cathédrale.

J'ajoute, je retranche, je déplace...

Merci !