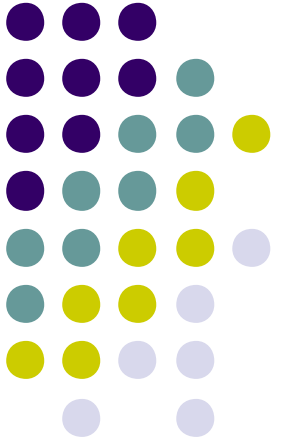
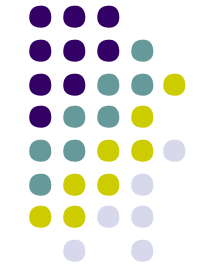
ift 3051 – projet d'informatique

# QUALITÉ DES PROGRAMMES ET PATRONS DE CONCEPTION



# La notion de qualité d'un logiciel



- Notion la plus importante d'un logiciel, 3 définitions connues à ce jour .
- Mesurer  la qualité se mesure à l'aide de métriques
- Patrons de conceptions dégradent la maintenabilté d'un programme, et les métriques orientées objets
- Étudier les liens susceptibles d'exister entre la qualité et les patrons de conceptions

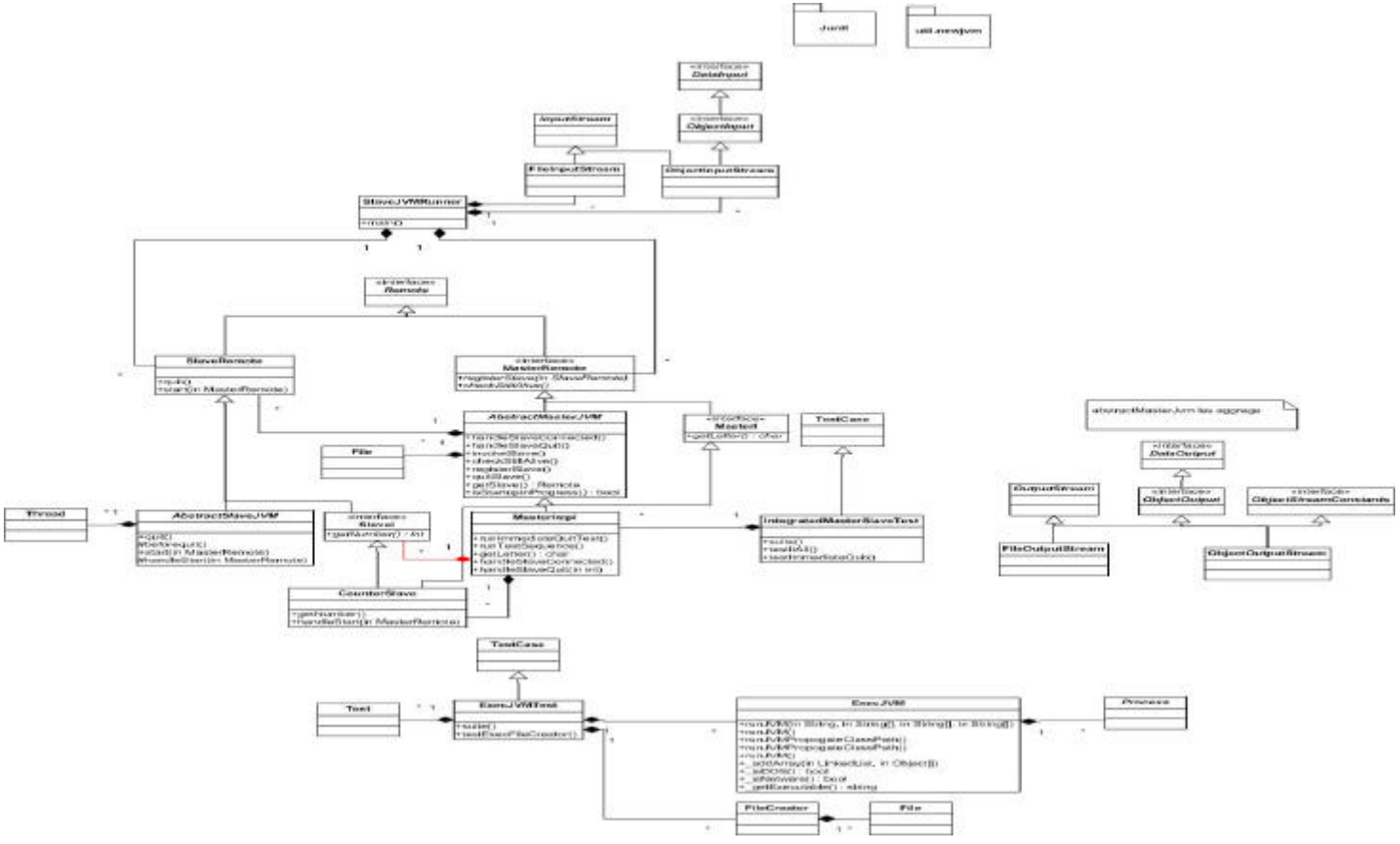
## Objectifs du projet et environnement de travail

- Fichier résultats Xml pour le calcul des métriques
- Environnement de travail : plate Forme Eclipse et Microsoft visio
- Outil de travail: le livre de référence "*Design Patterns Elements of Resusable Object Oriented Software*" de Gamma et AL

# Première méthode

- Réalisation des diagrammes UML à partir de classes d'un seul packge
- Obtention de "pseudo patrons de conceptions "
- Revue de la méthode de recherche

# Présentation d'un pseudo patrons de conception



# Deuxième méthode

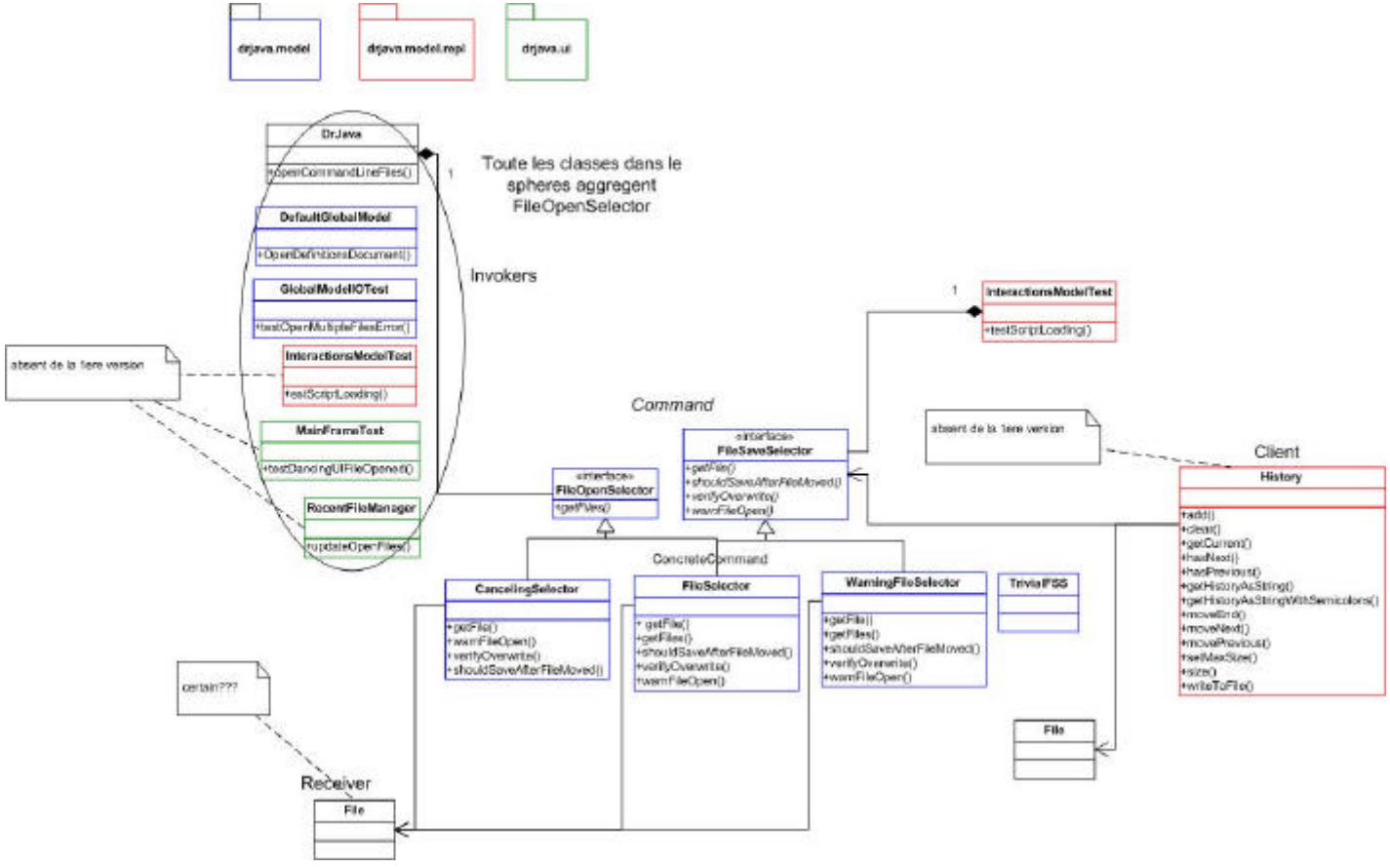
- Étape 1: Recherche du mot “interface ”

- Étape2: Réalisation des diagrammes UML à partir des relations d’aggrégation et d’héritage à l’aide du logiciel Microsoft Visio

- Étape 3: Identification d’un patron de conception à l’aide du livre de référence

  Obtention d’un patron de conception à partir de classes de packages différents.

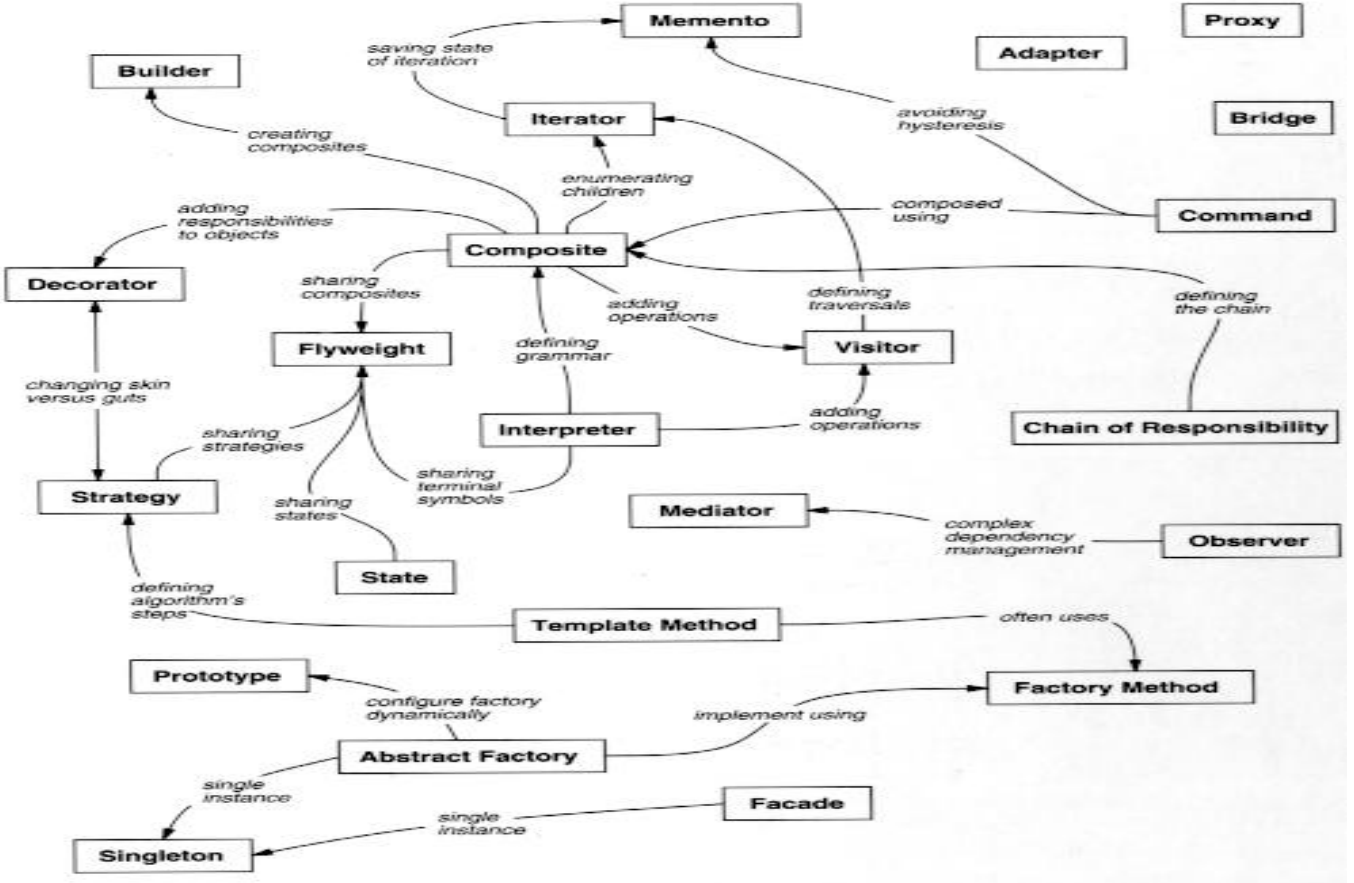
# Réprésentation d'un patron de conception à partir de classes appartenant à des packages différents



# Troisième méthode

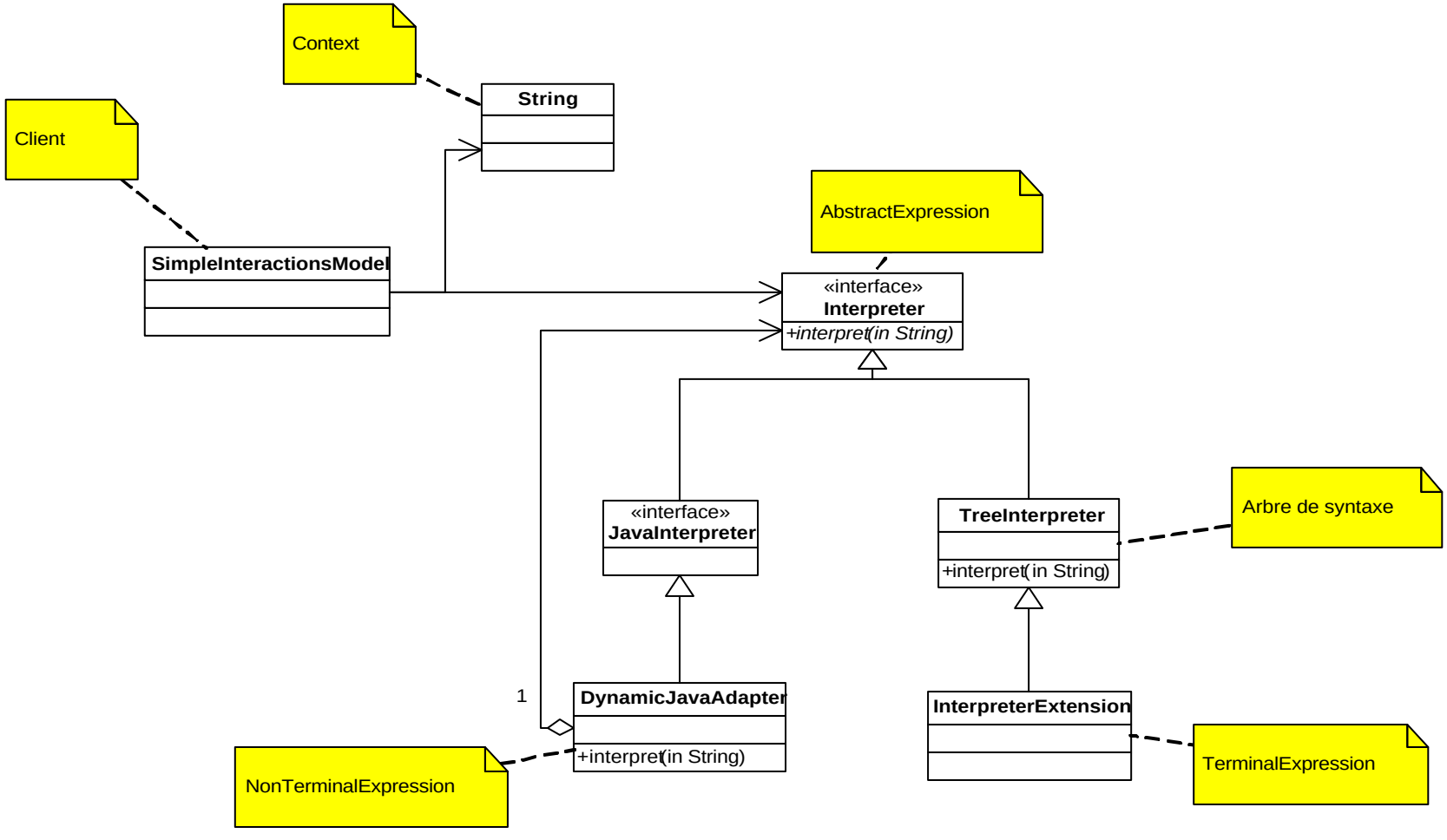
- Elle repose sur le même principe que la deuxième méthode
- Elle est entièrement basée sur le schéma des liens du livre de référence
- Le prochain patron de conception qui est susceptible d'être trouvé dans le programme est identifié
- Aussi sur le nom des méthodes ou des classes
- À l'aide des verbes qui réfèrent à un des groupes de patrons de conception
- Vérification de le structure logique lorsque la structure physique ne correspond à aucun patron connu.

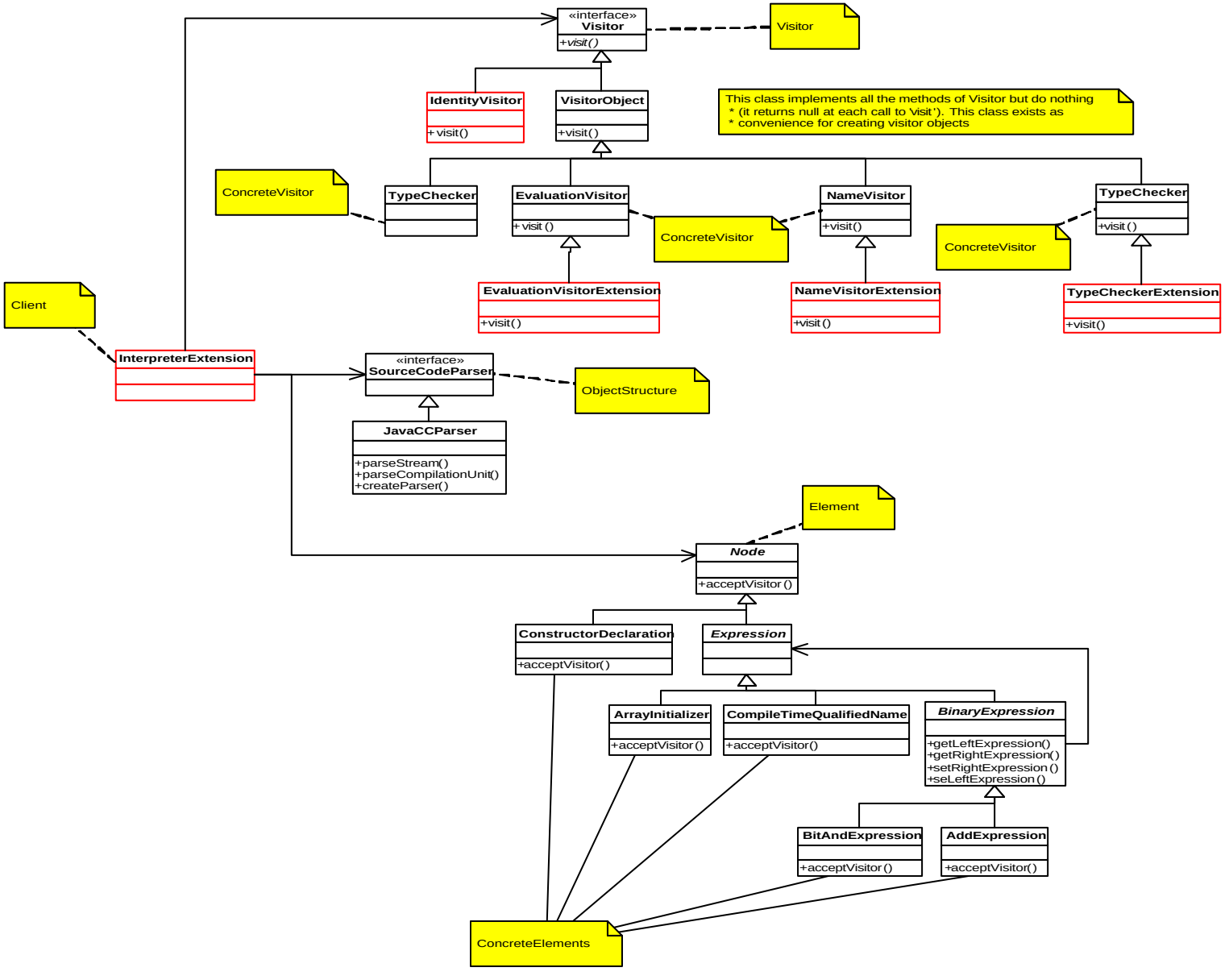
# Schéma des liens du livre de Gamma et AL



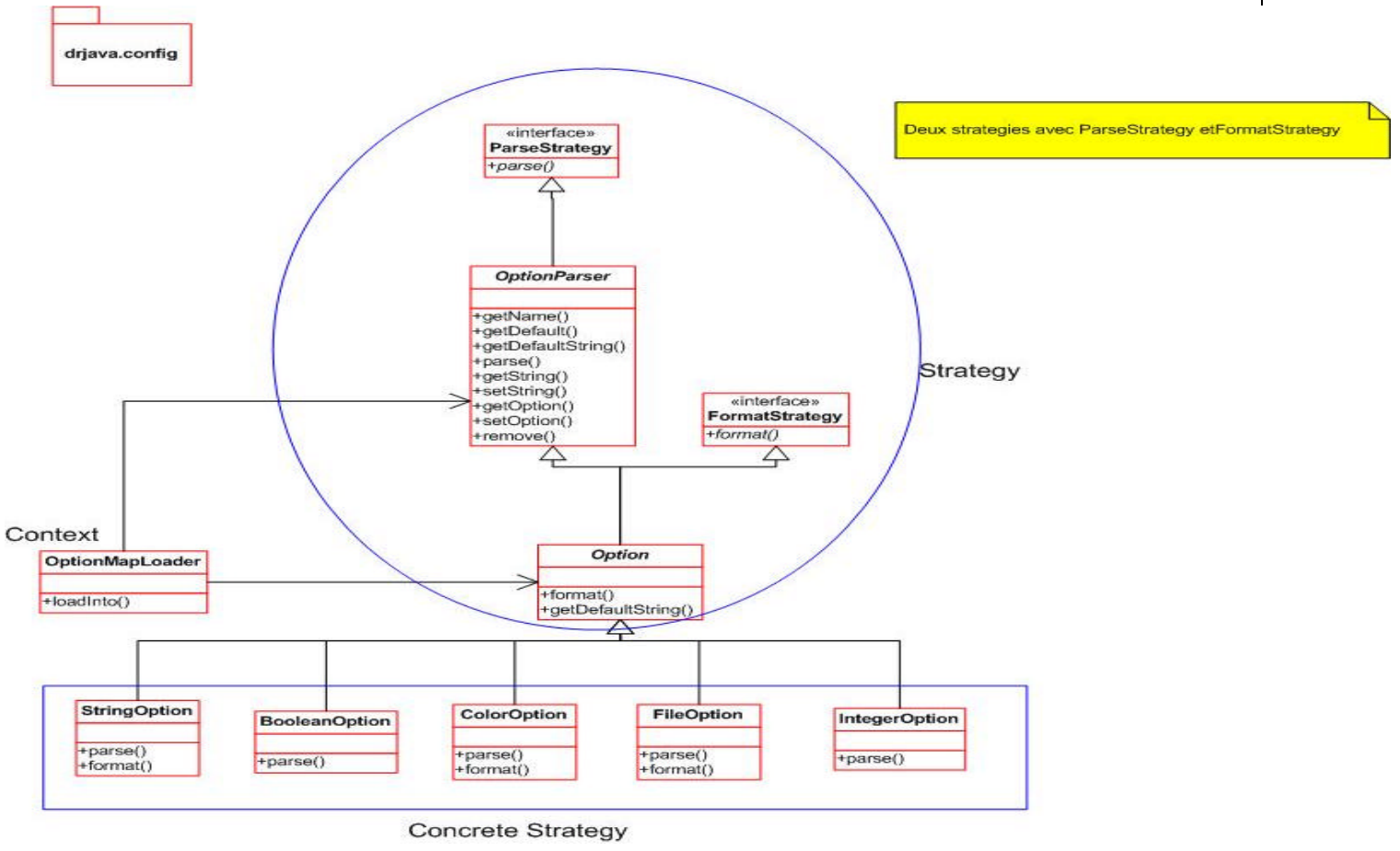

Design Pattern Relationships

# Exemple de l'application de la troisième méthode (Patron INTERPRETER vers Patron VISITOR)

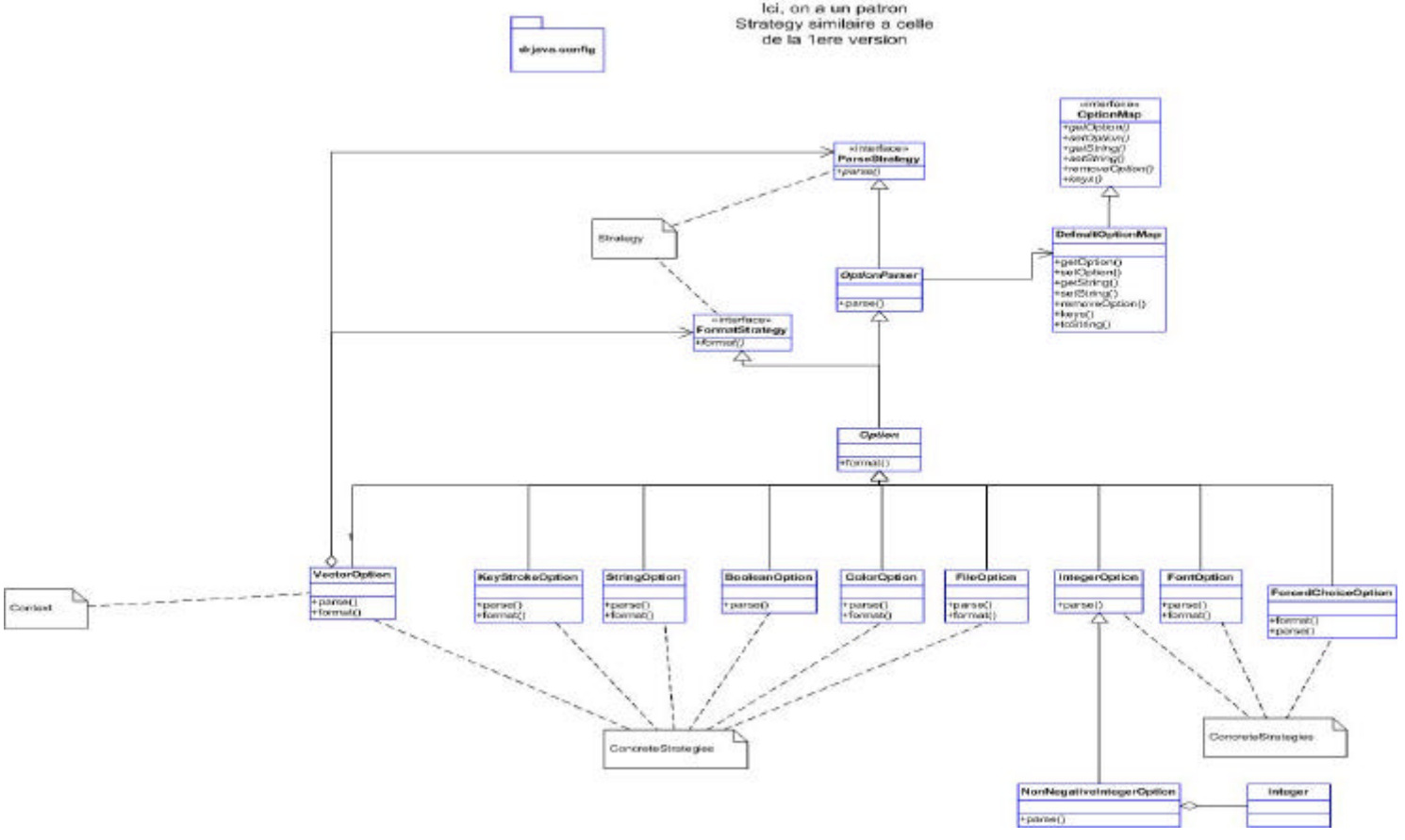



«interface»
Visitor
+visit()
Visitor
IdentityVisitor
+ visit()
VisitorObject
+visit()
This class implements all the methods of Visitor but do nothing
* (it returns null at each call to 'visit'). This class exists as
* convenience for creating visitor objects
ConcreteVisitor
TypeChecker
EvaluationVisitor
+ visit ()
ConcreteVisitor
NameVisitor
+visit()
ConcreteVisitor
TypeChecker
+visit ()
Client
EvaluationVisitorExtension
+visit()
NameVisitorExtension
+visit ()
TypeCheckerExtension
+visit()
InterpreterExtension
«interface»
SourceCodeParser
ObjectStructure
JavaCCParser
+parseStream()
+parseCompilationUnit()
+createParser()
Element
Node
+acceptVisitor ()
ConstructorDeclaration
+acceptVisitor()
Expression
ArrayInitializer
+acceptVisitor ()
CompileTimeQualifiedName
+acceptVisitor()
BinaryExpression
+getLeftExpression()
+getRightExpression()
+setRightExpression ()
+seLeftExpression ()
BitAndExpression
+acceptVisitor ()
AddExpression
+acceptVisitor()
ConcreteElements

# Présentation de l'identification par les noms des classes d'un patron de conception similaire au patron de conception STRATEGY de la version 20020619



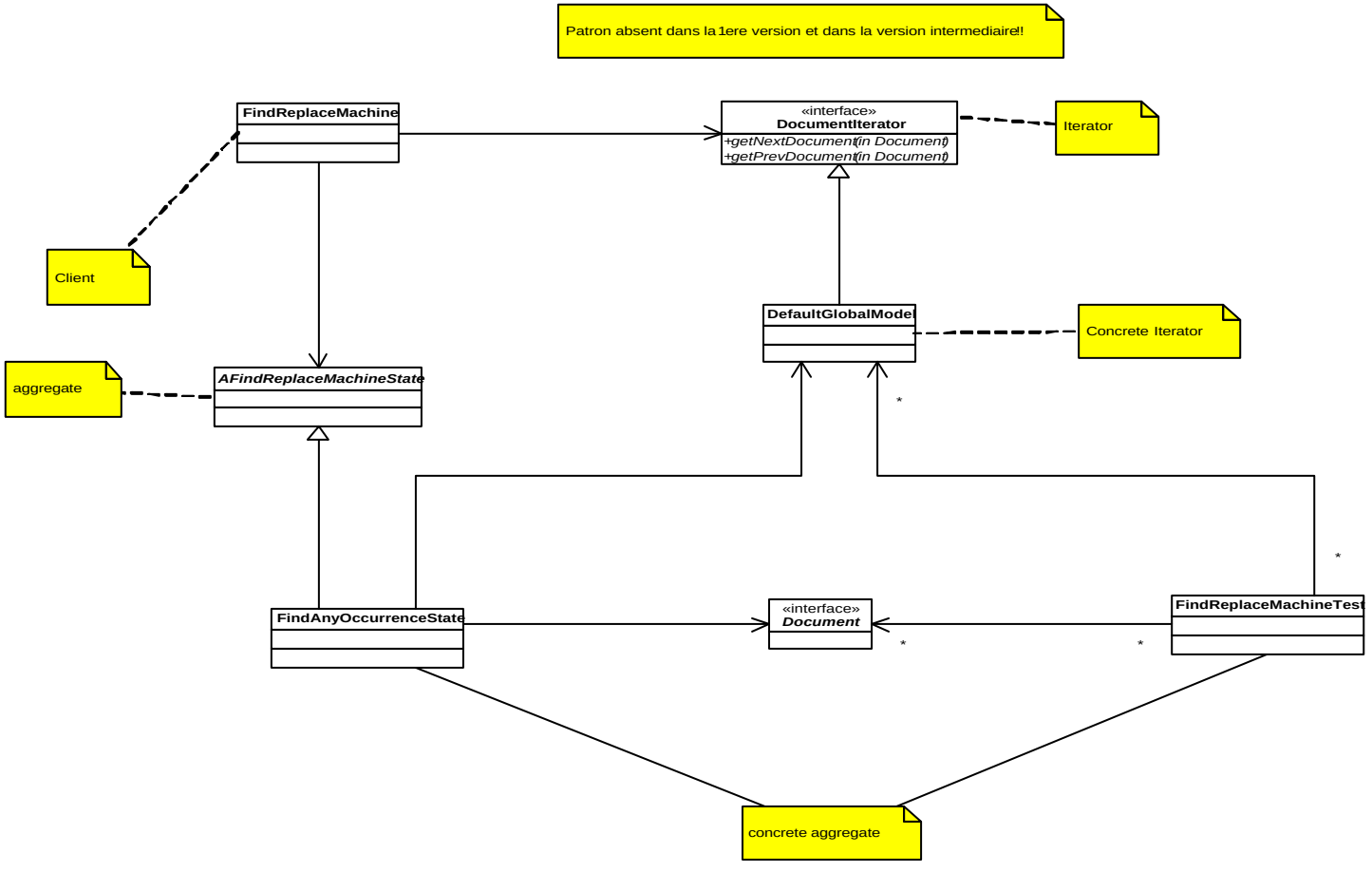
# Présentation de l'identification par les noms des classes d'un patron de conception similaire au patron de conception STRATEGY de la version 20040326



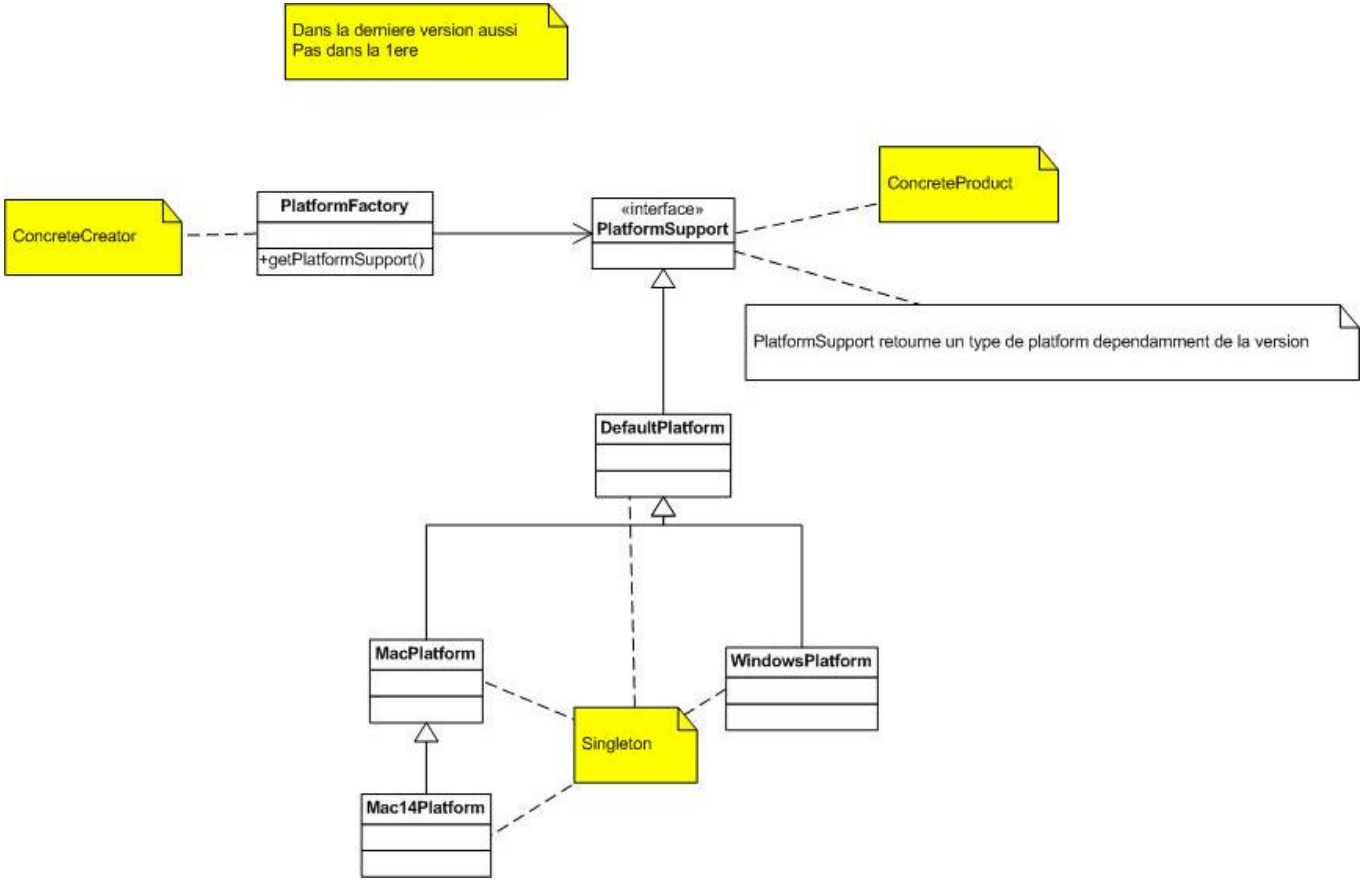
# Tableau des résultats

| Version 2002-06-19 | Version 2003-02-03 | Version 2004-03-26 |
|---|---|---|
| Iterator | Idem | Idem |
| Strategy(2) | Modification | Idem |
| Command(2) | Une modification | Une modification |
| Proxy | Idem | Idem |
| Adapter | Absent | Absent |
| Memento | Idem | Idem |
| Interpreter (incomplet) | Idem | Interpreter (complet) |
| Visitor | Modification | Idem |
| Absent | Factory Method | Idem |
| Absent | Absent | Abstract Factory |
| Absent | Absent | Adapter |
| Absent | Absent | State |
| Absent | Absent | Iterator |

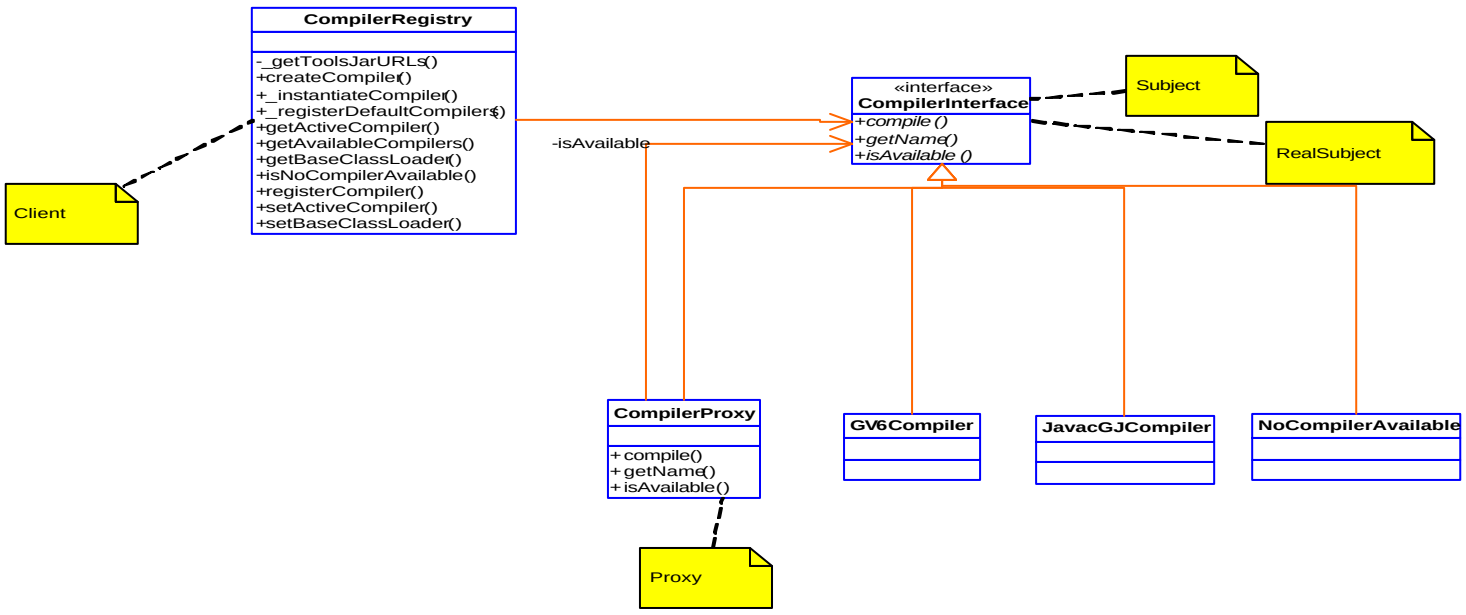
# Présentation d'un patron de conception Iterator différent de celui du livre de référence



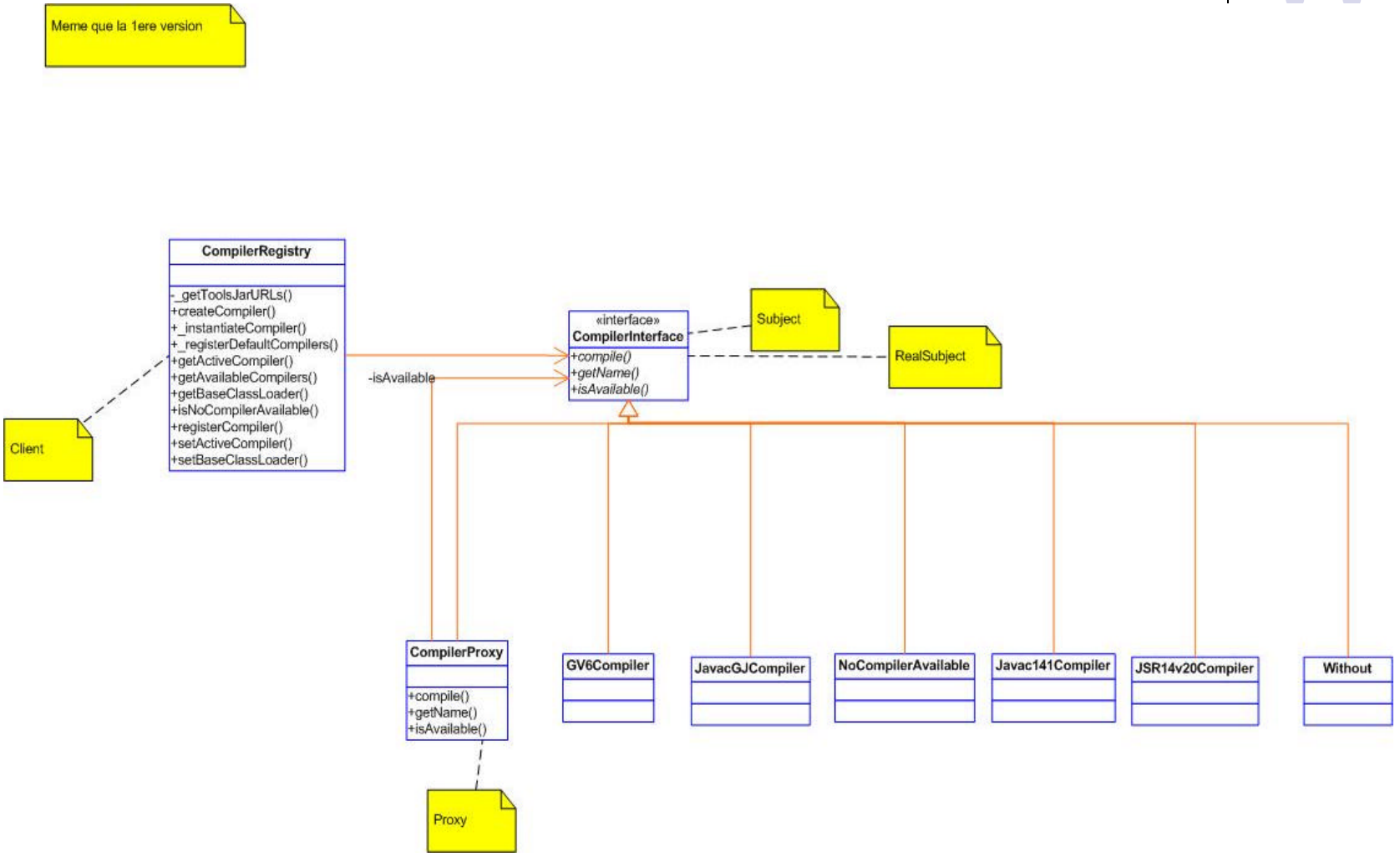
# Présentation d'un patron de conception Factory Method



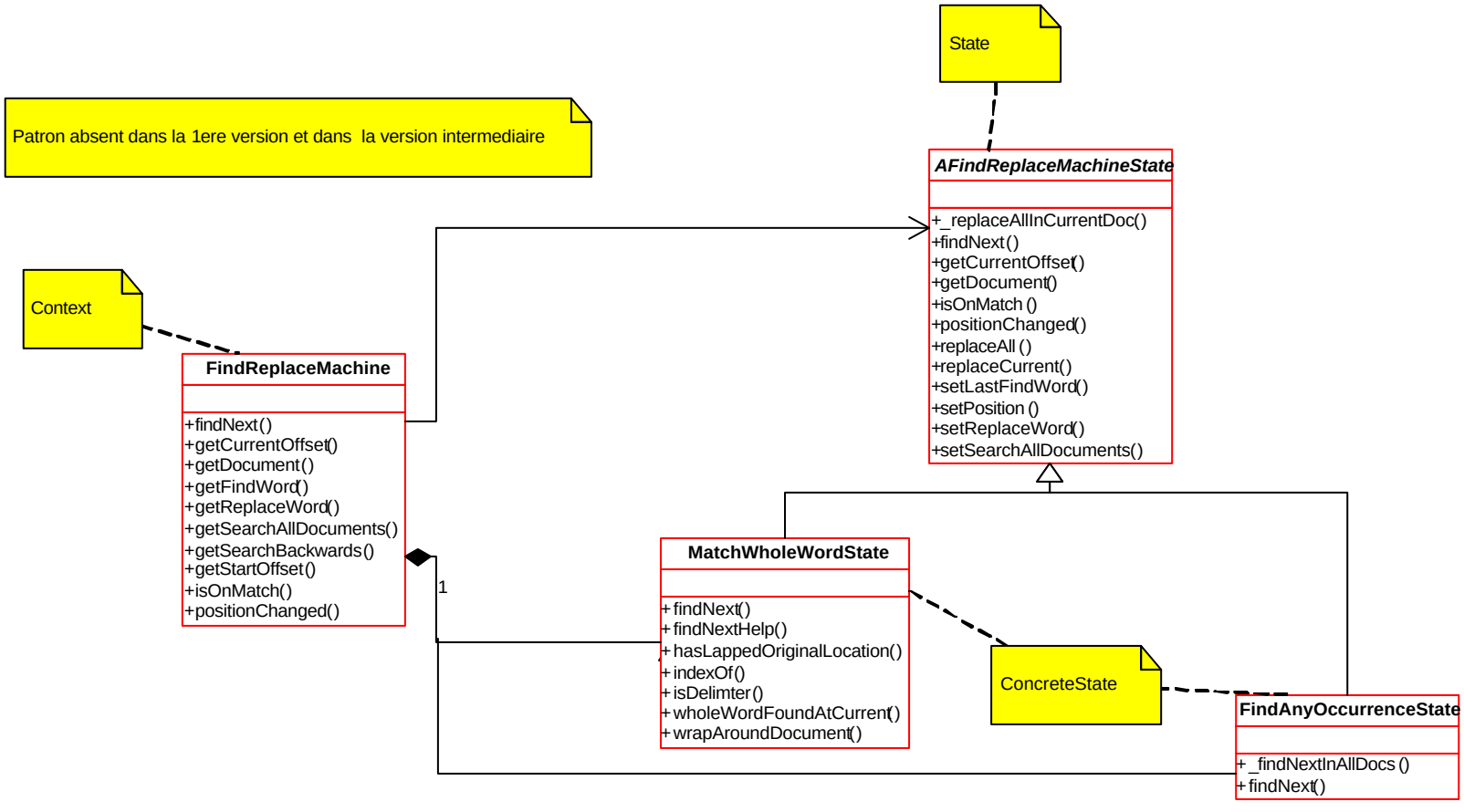
# Présentation d'un patron de conception Proxy de la version 20020619

# Présentation d'un patron de conception Proxy de la version 20040326



# Présentation d'un patron de conception State de la version 20040326



# conclusion

- les patrons de conception rencontrés permettent d'améliorer la compréhension du code
- Le fait que les classes et les méthodes aient des noms significatifs nous a aidé dans l'identification des patrons de conception
- Découverte d'un anti – patron de conception de type blob dans la classe « DefinitionsDocument ».
- L'utilisation de jdk 1.5 a approfondi nos connaissances en Java (découverte de template dans le code)
- Utilisation de visio est plus performante que celle d'omundo ou rational rose