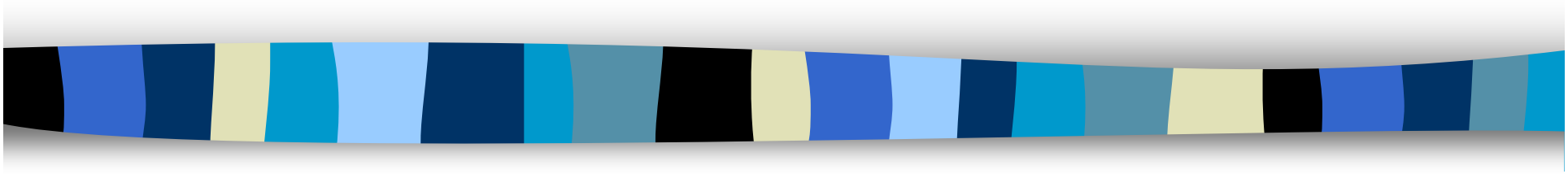
Qualité des programmes et patrons de conception



IFT3051 – Projet été 2005

Auteurs: Vinh Thinh Le
Joseph Vong
Sébastien Boisclair

Ptidej Team – OO Programs Quality Evaluation and Enhancement using Patterns
Group of Open, Distributed Systems, Experimental Software Engineering
Department of Informatics and Operations Research
University of Montreal

Plan

- Énoncé
- Description des outils
  - 

  - 

  - 

- Méthodologies
- Résultats de la recherche de patrons
- Évolution des patrons
- Conclusion

Énoncé

- Tâches à effectuer:
  - Comparer des outils de rétro-conception
  - Analyser différentes versions de Dr Java et identifier les patrons de conception utilisés pour la conception et l'implantation
  - Corréler l'introduction, la suppression ou la modification de patrons de conception avec les demandes de changements
- Résultats concrets:
  - Microarchitectures sous format XML

Critères d'évaluation

Nom du logiciel :

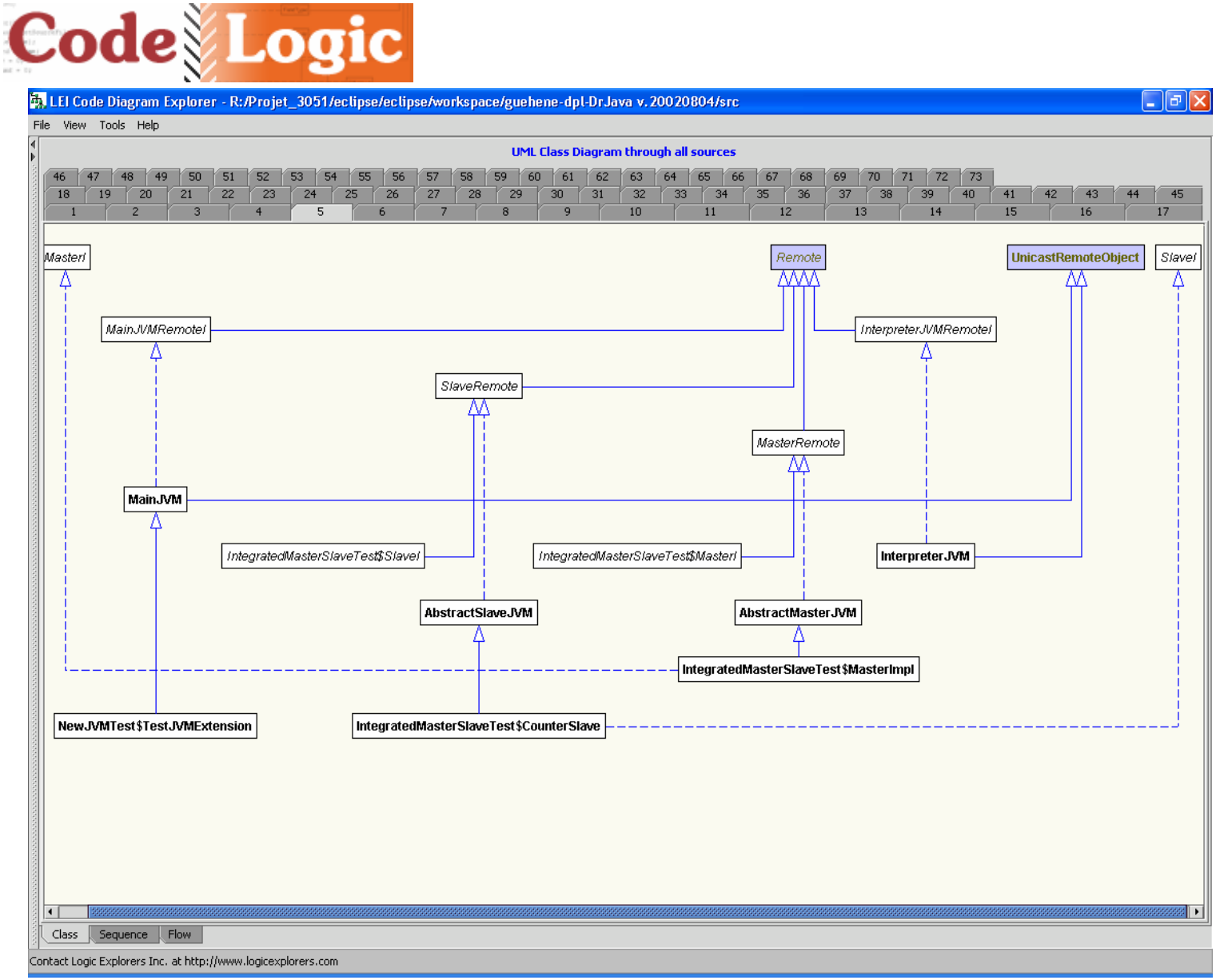
Critères primordiaux	Oui	Non	Évaluation	Commentaires
Diagrammes UML supportés - Diagramme de séquence - Diagramme de classes - Diagramme d'activités - Diagramme d'états - Diagramme de collaboration				
Possibilité de rétro conception				
Pertinence du résultat de la rétro conception			1 2 3 4 5	
Possibilité de définir les associations pertinentes.				
Vitesse pour générer le diagramme de classe			1 2 3 4 5	
Pertinence du résultat des patrons de conceptions			1 2 3 4 5	
Critères secondaires				
Possibilité de faire un réalignement du diagramme de classe				
Non obligation d'avoir un code sans erreur pour générer un diagramme de classe (compilable)				
Aucune nécessité d'avoir certains types de ficher (autre que .java) pour générer les diagrammes				
Possibilité de sauvegarder les diagrammes en différents formats				
Convivialité et facilité d'utilisation du logiciel			1 2 3 4 5	
Génère un API à partir du diagramme de classe				
Esthétique des diagrammes et de rétro conception			1 2 3 4 5	
Synchronisation d'une modification à un diagramme sur les autres diagrammes (refractoring)			1 2 3 4 5	
Supporte Java 1.5				
Supporte UML 2.0				
Limitations du programme				
Langage de programmation supporté (autre que java)				
Qualité de la documentation			1 2 3 4 5	
Indépendance sur différentes plateformes				
Offre les options de Undo / Redo recursif				
Facilité d'installation				
Autres				
Bugs notables				
Commentaires				

- Nom : 

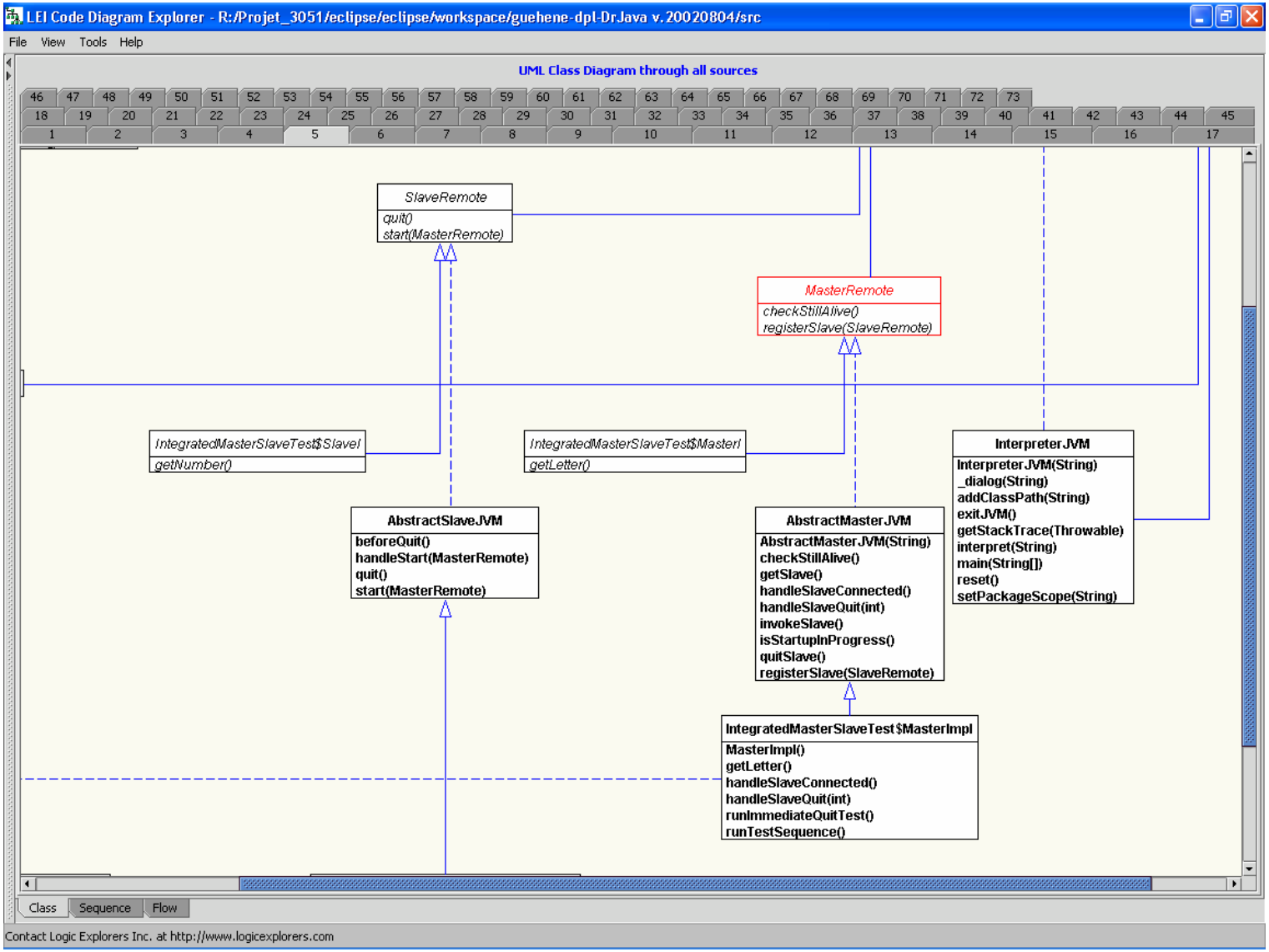
- Retenu par :
  - Joseph Vong

Code Logic (Tableau d'analyse)

Critères primordiaux	Oui	Non	Évaluation	Commentaires
Diagrammes UML supportés - Diagramme de séquence - Diagramme de classes - Diagramme d'activités - Diagramme d'états - Diagramme de collaboration	 X X	 X X X		Dessine de très beaux diagrammes de classe et de séquences facile à comprendre.
Possibilité de rétro conception	X			
Pertinence du résultat de la rétro conception			1 2 3 **4** 5	
Possibilité de définir les associations pertinentes.		X		
Vitesse pour générer le diagramme de classe			1 2 3 **4** 5	
Pertinence du résultat des patrons de conceptions			1 2 3 4 5	
Critères secondaires				
Possibilité de faire un réalignement automatique du diagramme de classe		X		Ne bouge pas (static)
Non obligation d'avoir un code sans erreur pour générer un diagramme de classe (compilable)		X		Ne génère pas les parties avec erreur.
Aucune nécessité d'avoir certains types de ficher (autre que .java) pour générer les diagrammes	X			
Possibilité de sauvegarder les diagrammes en différents formats	X			PHG, Visio.net, XMI,…
Convivialité et facilité d'utilisation du logiciel			1 2 3 4 **5**	
Génère un API à partir du diagramme de classe		X		
Esthétique des diagrammes et de rétro conception			1 2 3 4 **5**	
Synchronisation d'une modification à un diagramme sur les autres diagrammes (refractoring)			1 2 3 4 5	Aucune modification possible
Supporte Java 1.5				
Supporte UML 2.0	X			
Limitations du programme	X			Trial Version
Langage de programmation supporté (autre que java)	X			C#
Qualité de la documentation			1 **2** 3 4 5	
Indépendance sur différentes plateformes		X		
Offre les options de Undo / Redo recursif		X		
Facilité d'installation		X		Licence requis à chaque fois
Autres				
Bugs notables				
Commentaires	Excellent programme qui fait tout ce que l'on a besoin à première vu. Malheureusement le trial nous limite à moins d'options que le programme peut offrir.			

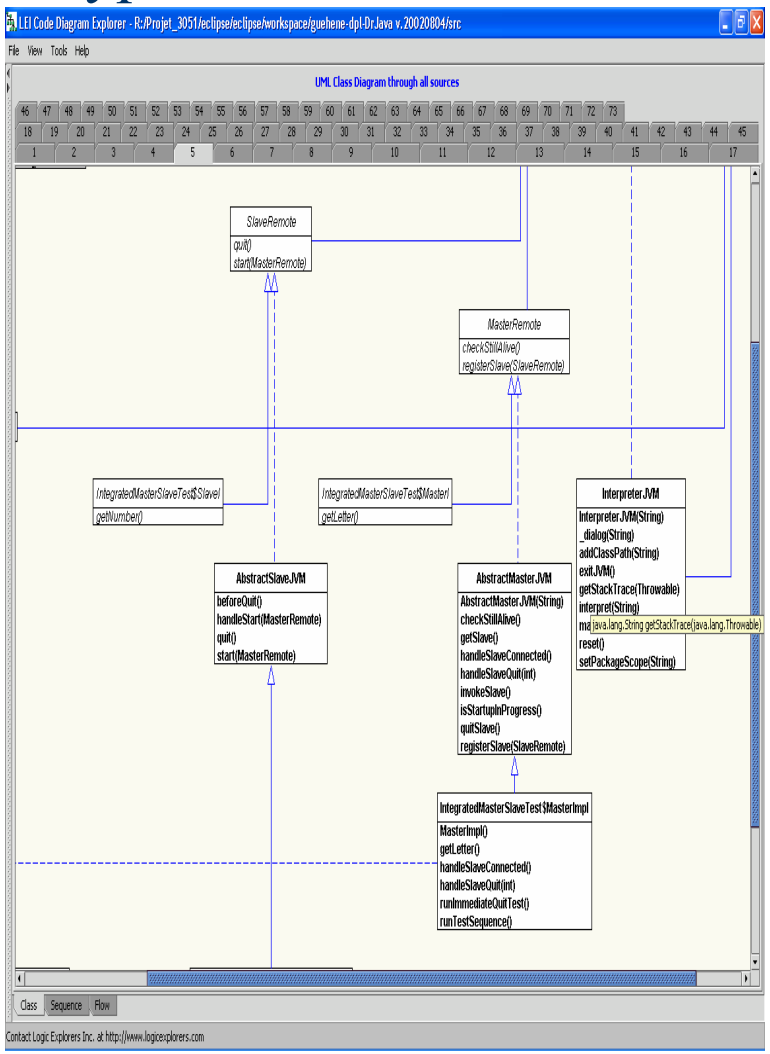

Code Logic
LEI Code Diagram Explorer - R:/Projet_3051/eclipse/eclipse/workspace/guehene-dpl-DrJava v.20020804/src
File View Tools Help
UML Class Diagram through all sources
MasterI
Remote
UnicastRemoteObject
SlaveI
MainJVMRemoteI
InterpreterJVMRemoteI
SlaveRemote
MasterRemote
MainJVM
IntegratedMasterSlaveTest$SlaveI
IntegratedMasterSlaveTest$MasterI
InterpreterJVM
AbstractSlaveJVM
AbstractMasterJVM
IntegratedMasterSlaveTest$MasterImpl
NewJVMTest$TestJVMExtension
IntegratedMasterSlaveTest$CounterSlave
Class Sequence Flow
Contact Logic Explorers Inc. at http://www.logicexplorers.com

Code Logic

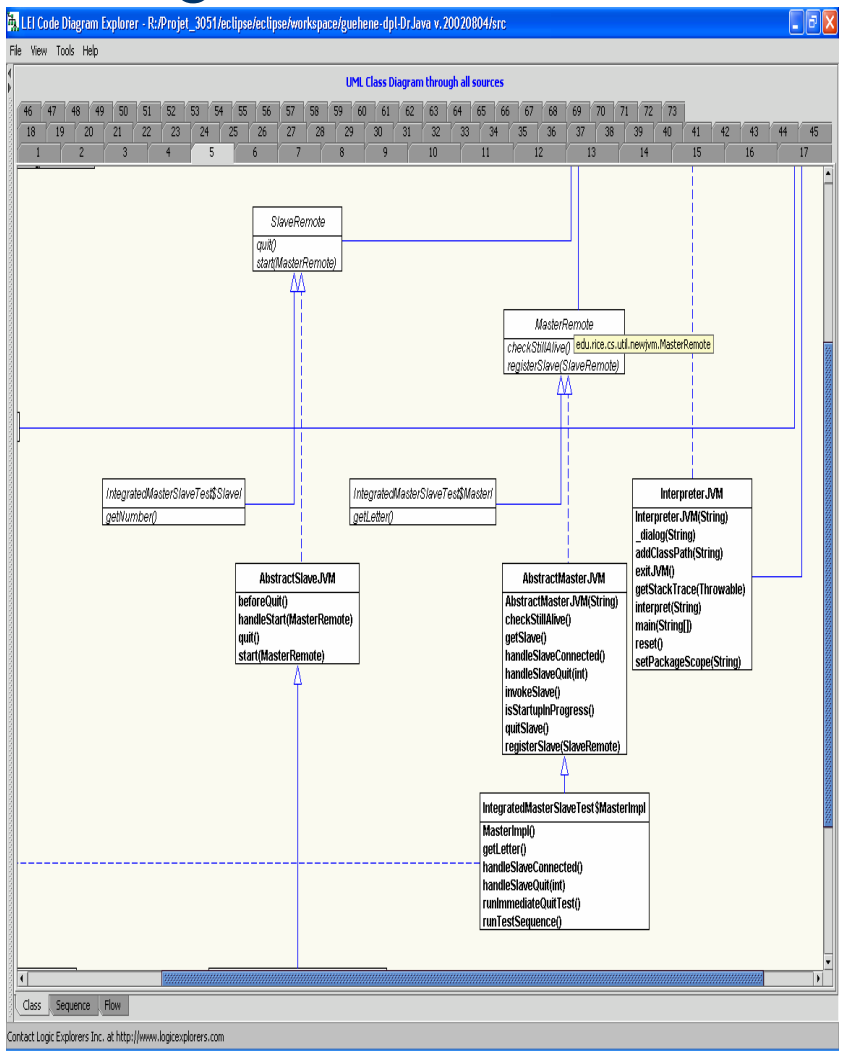

LEI Code Diagram Explorer - R:/Projet_3051/eclipse/eclipse/workspace/guehene-dpl-DrJava v. 20020804/src
File View Tools Help
UML Class Diagram through all sources
SlaveRemote
quit()
start(MasterRemote)
MasterRemote
checkStillAlive()
registerSlave(SlaveRemote)
IntegratedMasterSlaveTest$SlaveI
getNumber()
IntegratedMasterSlaveTest$MasterI
getLetter()
InterpreterJVM
InterpreterJVM(String)
_dialog(String)
addClassPath(String)
exitJVM()
getStackTrace(Throwable)
interpret(String)
main(String[])
reset()
setPackageScope(String)
AbstractSlaveJVM
beforeQuit()
handleStart(MasterRemote)
quit()
start(MasterRemote)
AbstractMasterJVM
AbstractMasterJVM(String)
checkStillAlive()
getSlave()
handleSlaveConnected()
handleSlaveQuit(int)
invokeSlave()
isStartupInProgress()
quitSlave()
registerSlave(SlaveRemote)
IntegratedMasterSlaveTest$MasterImpl
MasterImpl()
getLetter()
handleSlaveConnected()
handleSlaveQuit(int)
runImmediateQuitTest()
runTestSequence()
Class Sequence Flow
Contact Logic Explorers Inc. at http://www.logicexplorers.com

Code Logic

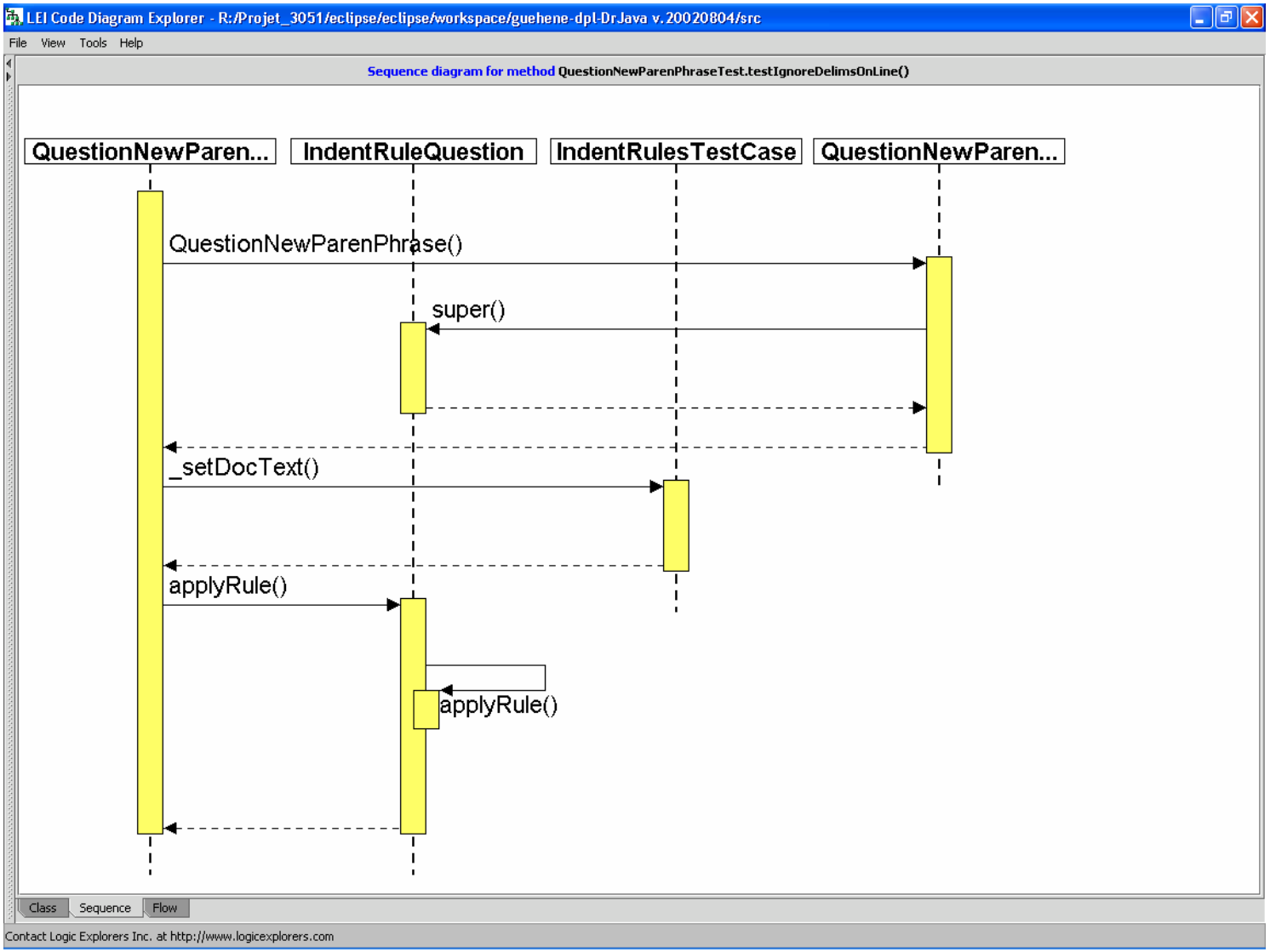
-Type de retour



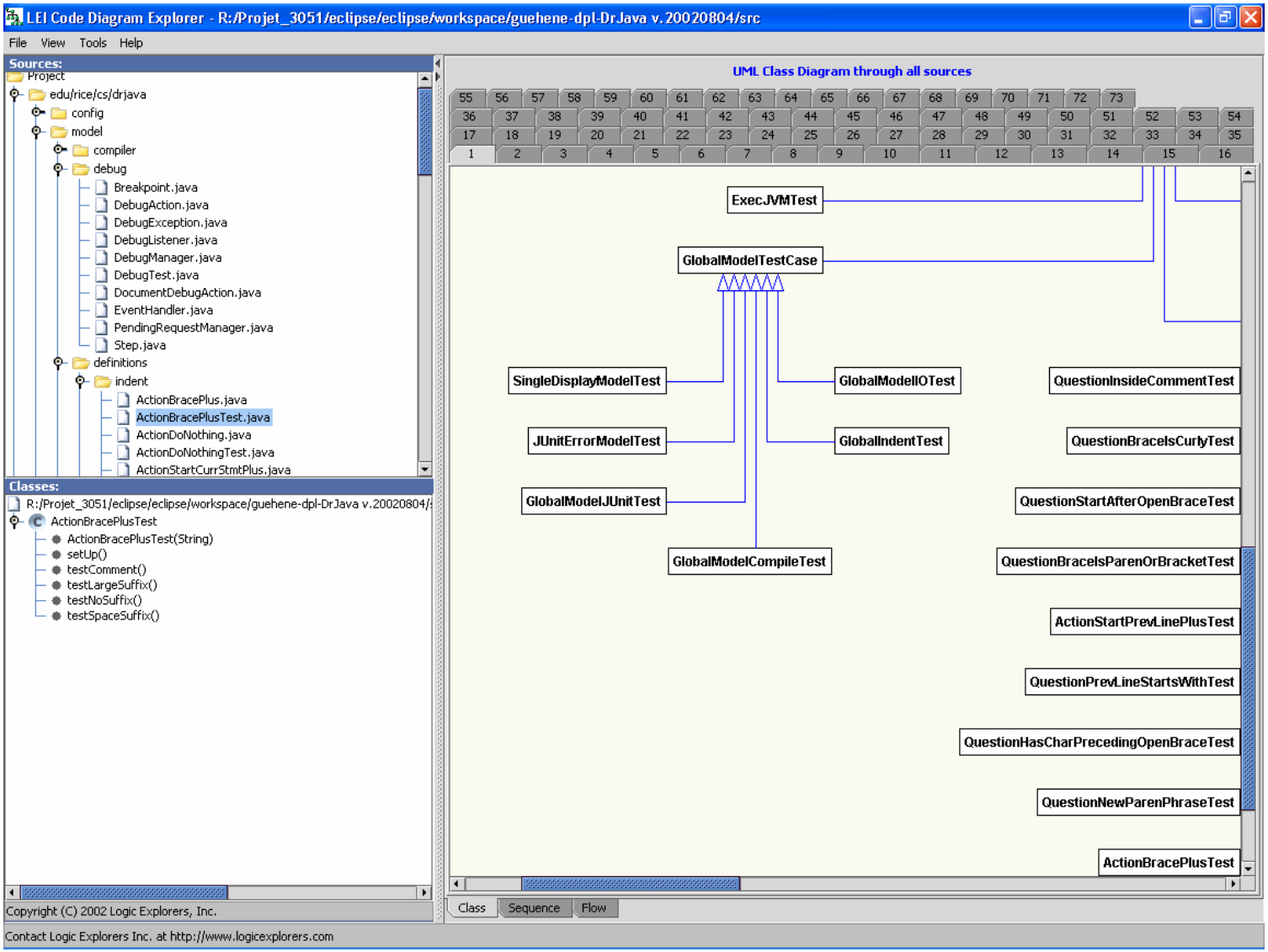
-Package de la classe



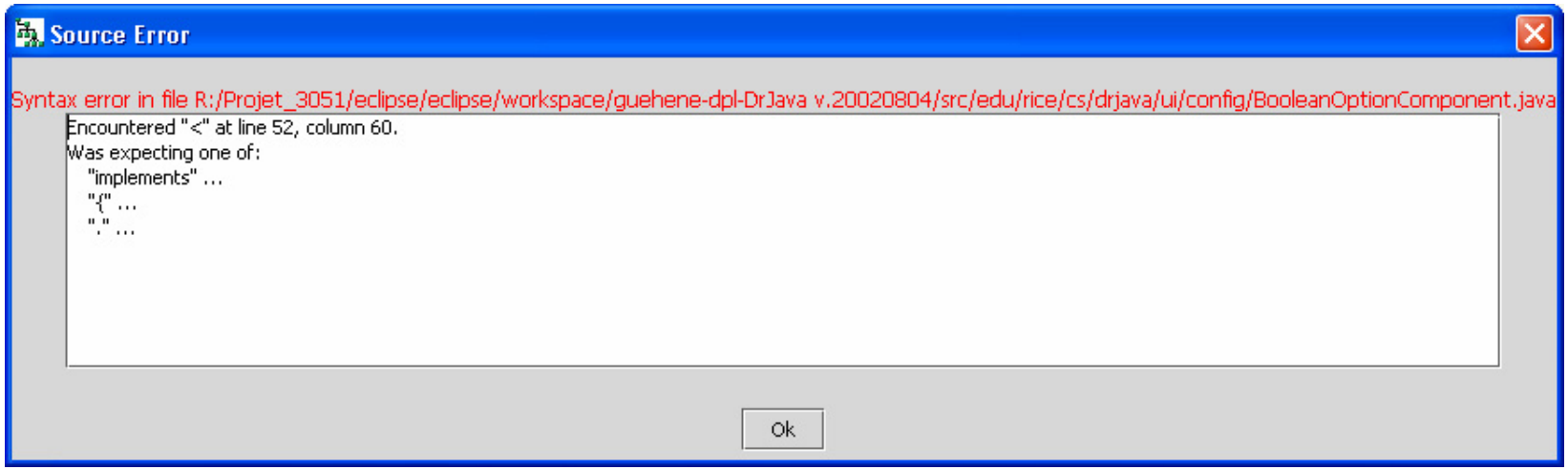
Code Logic Diagramme de séquence

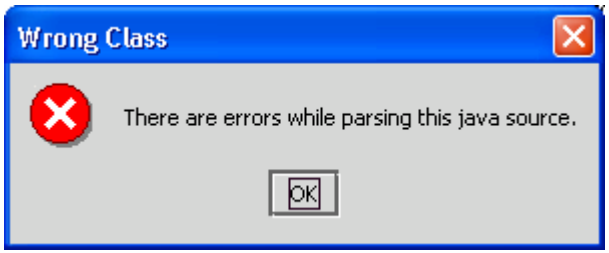


CodeLogic Bar de navigation



Erreurs générées





- **Avantages**
  - Facilité d'utilisation
  - Diagrammes faciles à comprendre
  - Considère tous les packages lors de la rétro-conception
  - Offre une extension ("plug-in") à Eclipse

- **Désavantages**
  - Ne supporte pas Java 1.5
    - Génère une fenêtre d'erreur pour **chaque** erreur de syntaxe trouvée lors de la rétro-conception
    - Rien est dessiné lorsqu'il retrouve une seule erreur de syntaxe.
  - Ne permet pas d'éditer les diagrammes
  - Ne dessine pas les agrégations et les appelles de fonctions.
  - Très peu documenté

- Nom : omondo THE LIVE UML COMPANY
- Retenu par :
  - Sébastien Boisclair

Nom du logiciel : Omondo

Critères primordiaux	Oui	Non	Evaluation	Commentaires
Diagrammes UML supportés - Diagramme de séquence - Diagramme de classes - Diagramme d'activités - Diagramme d'états - Diagramme de collaboration	 X X X X X			
Possibilité de rétro conception	X			
Pertinence du résultat de la rétro conception			1 2 3 **4** 5	
Possibilité de définir les associations pertinentes.	X			
Vitesse pour générer le diagramme de classe			1 2 3 **4** 5	
Pertinence du résultat des patrons de conceptions		X	1 2 3 4 5	Ne détecte pas de patron automatiquement
Critères secondaires				
Possibilité de faire un réalignement du diagramme de classe	X			
Non obligation d'avoir un code sans erreur pour générer un diagramme de classe (compilable)	X			Fonctionne même s'il y a des petites erreurs
Aucune nécessité d'avoir certains types de ficher (autre que .java) pour générer les diagrammes	X			
Possibilité de sauvegarder les diagrammes en différents formats	X			Limiter par la version gratuite
Convivialité et facilité d'utilisation du logiciel	X		1 2 3 **4** 5	
Génère un API à partir du diagramme de classe		X		
Esthétique des diagrammes et de rétro conception	X		1 2 3 **4** 5	
Synchronisation d'une modification à un diagramme sur les autres diagrammes (refractoring)	X		1 2 3 4 **5**	
Supporte Java 1.5	X			
Supporte UML 2.0	X			
Limitations du programme	X			Sur les prints et sauvegarde
Langage de programmation supporté (autre que java)		X		
Qualité de la documentation			1 2 3 **4** 5	
Indépendance sur différentes plateformes	X			Linux - Windows
Offre les options de Undo / Redo récursif		X		Dans le code oui mais pas sur les diagrammes
Facilité d'installation	X			
Autres				
Bugs notables				
Commentaires			Excellent extension pour éclipse	

Vue globale du logiciel

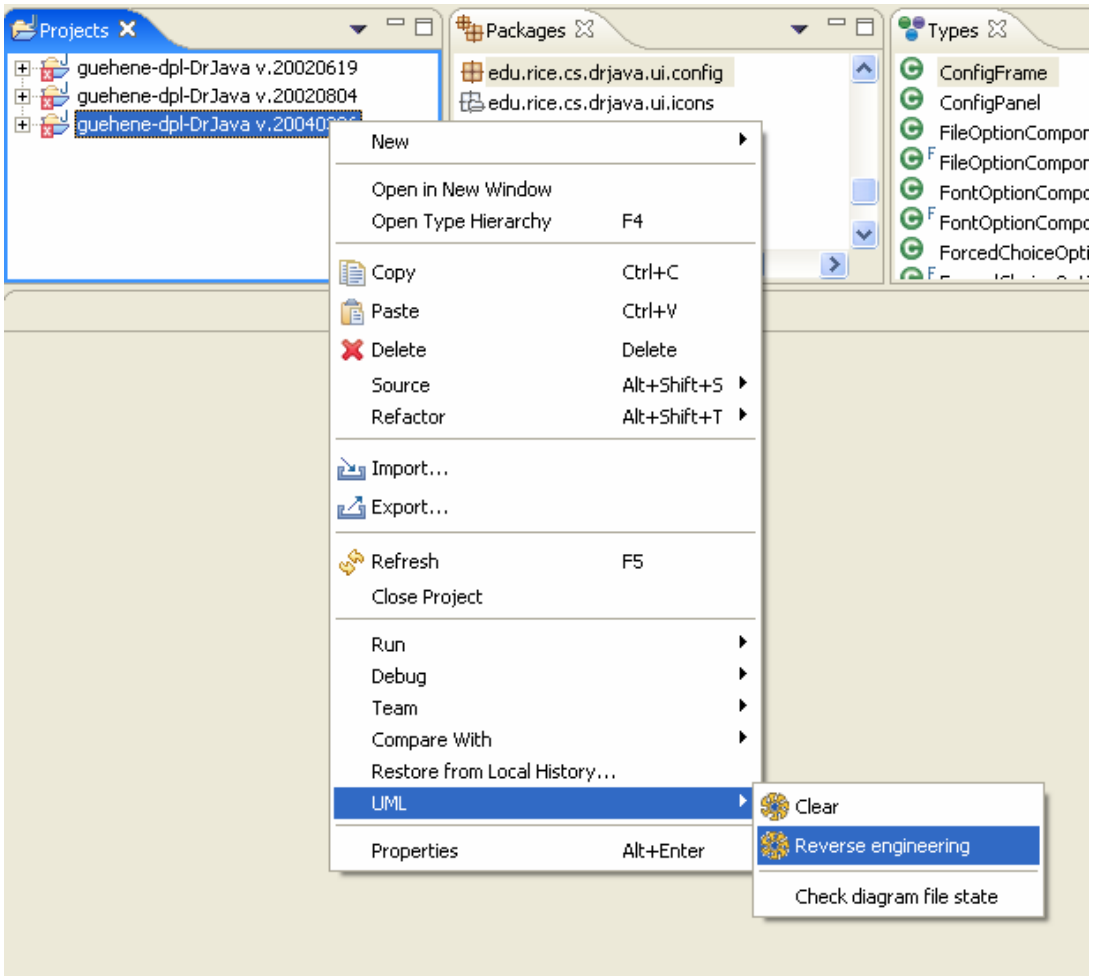
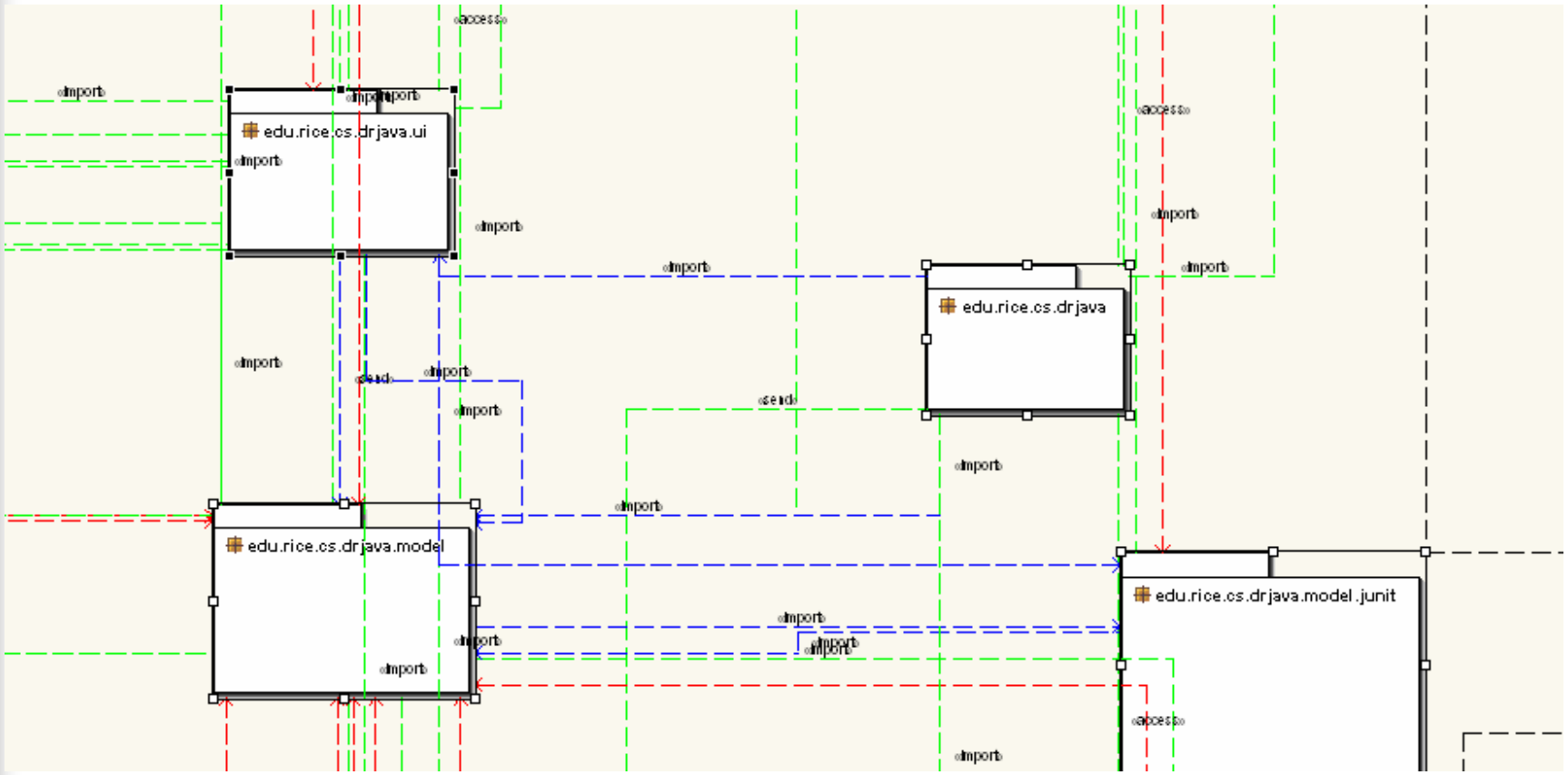
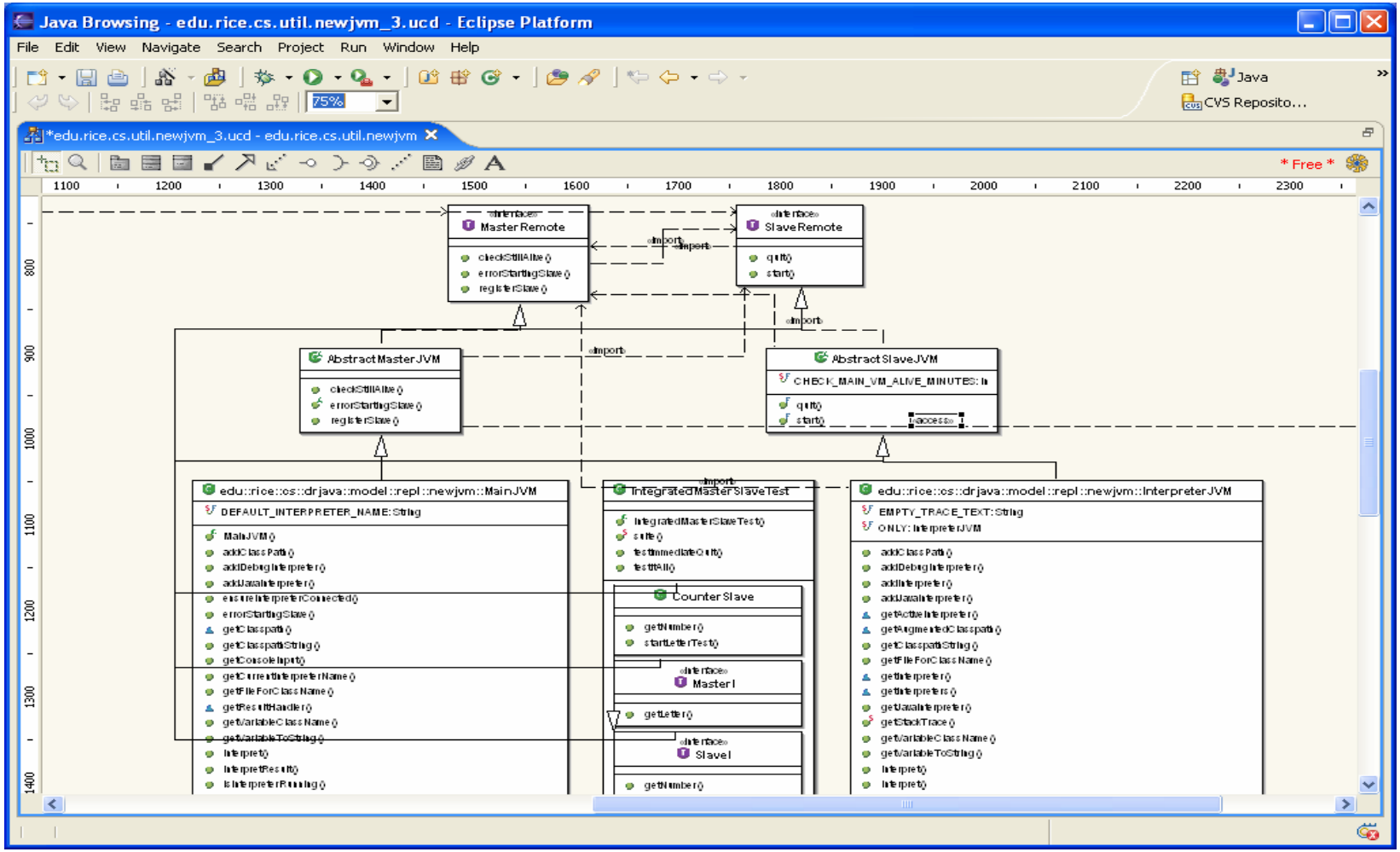


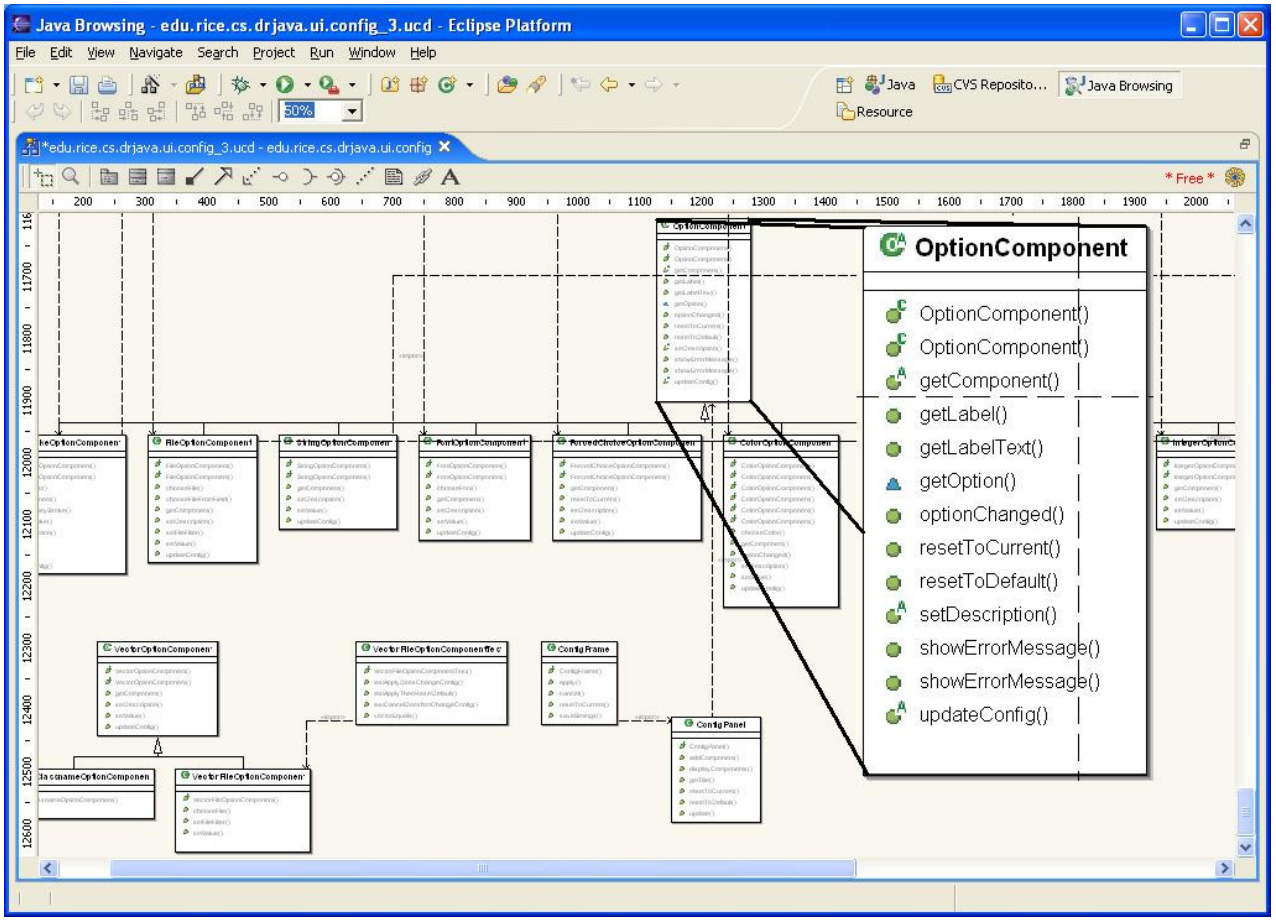
Diagramme des packages



Vue d'un diagramme de classes

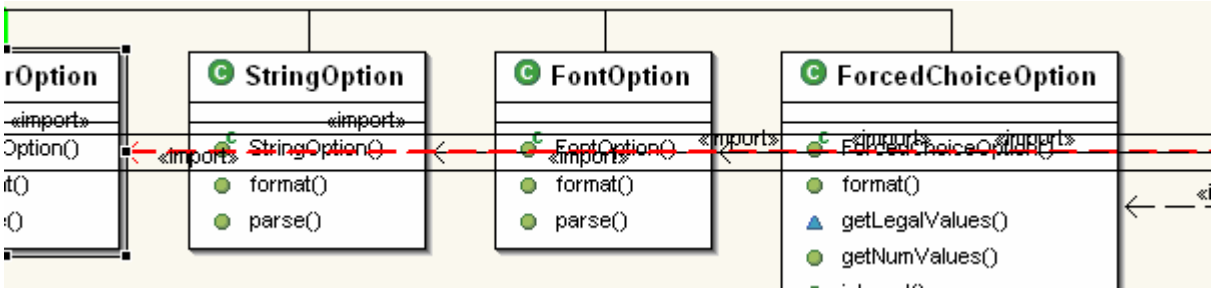


Support de Java 1.5



- **Avantages**
  - Facilité d'utilisation
  - Éditeur combiné avec Eclipse
  - Synchronisation automatique
  - Facilite grandement la recherche
  - Support de java 1.5
  - Gratuit pour les universités

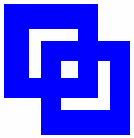
- Désavantages:
  - Disposition des liens parfois ambiguës et liens difficiles à suivre



  - Diagramme avec notation non conviviale
  - Difficulté à imprimer un diagramme

- Nom : 

- Retenu par :
  - Vinh Thinh Le

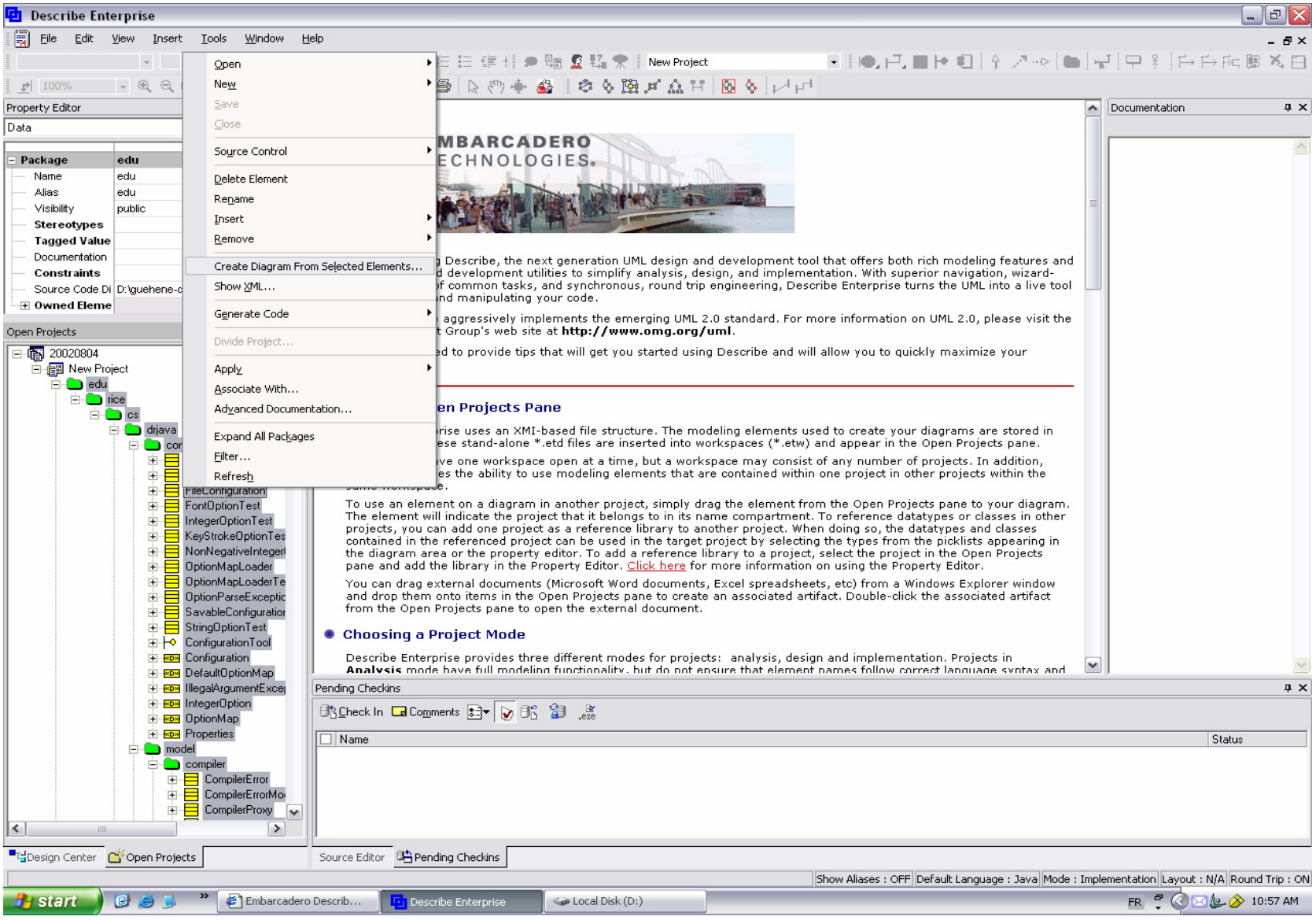


Describe

Nom du logiciel : Describe Entreprise 6.1.7

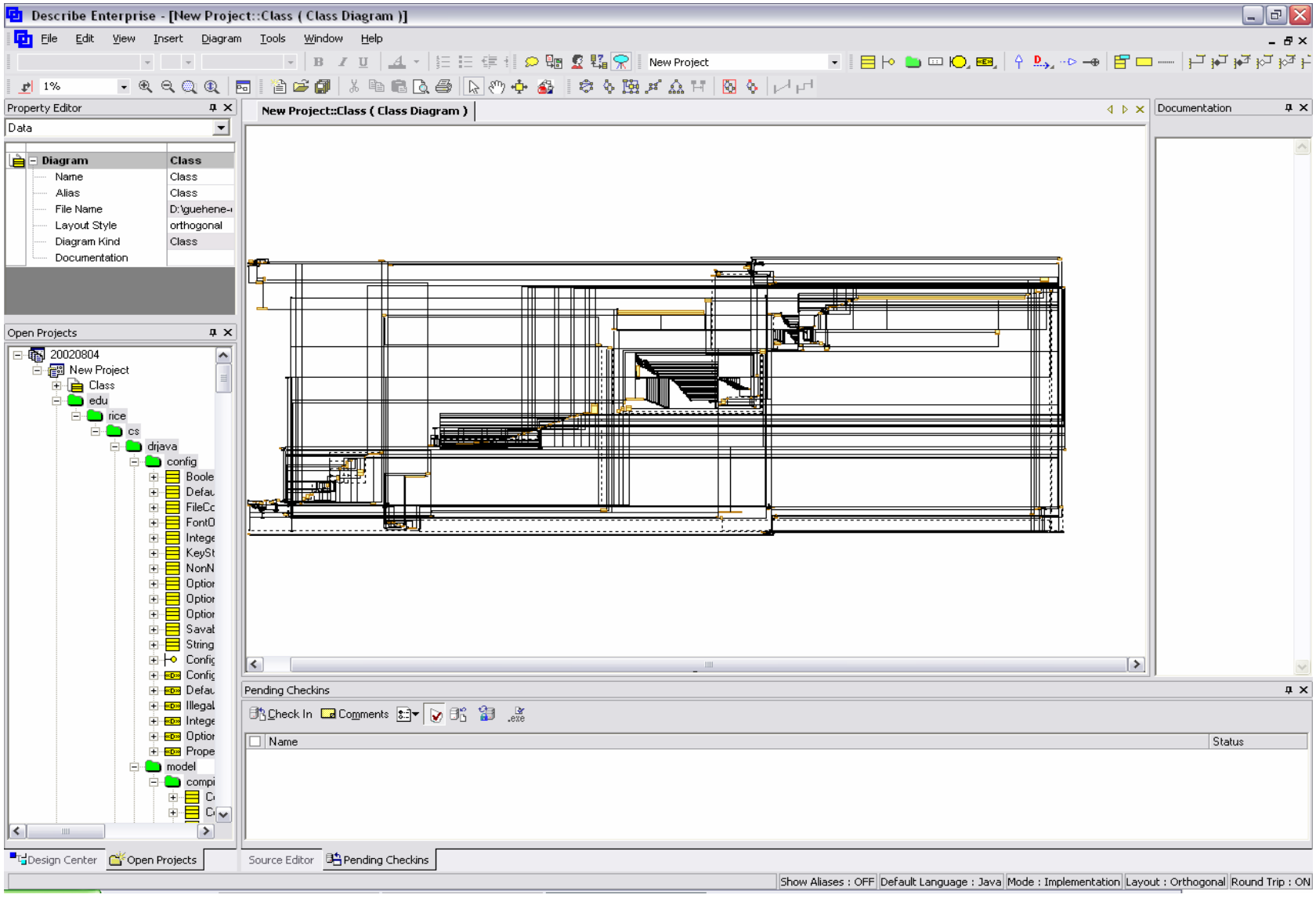
Critères primordiaux	Oui	Non	Évaluation	Commentaires
Diagrammes UML supportés - Diagramme de séquence - Diagramme de classes - Diagramme d'activités - Diagramme d'états - Diagramme de collaboration	 X X X X X			
Possibilité de rétro conception	X			
Pertinence du résultat de la rétro conception			1 2 3 4 5	
Possibilité de définir les associations pertinentes.	X			
Vitesse pour générer le diagramme de classe			1 2 3 4 5	
Pertinence du résultat des patrons de conceptions			1 2 3 4 5	
Critères secondaires				
Possibilité de faire un réalignement du diagramme de classe	X			Circulaire, arbre, symétrique….
Obligation d'avoir un code sans erreur pour générer un diagramme de classe (compilable)	X			
Aucune nécessité d'avoir certains types de ficher (autre que .java) pour générer les diagrammes	X			
Possibilité de sauvegarder les diagrammes en différents formats	X			En théorie, dans la version complète
Convivialité et facilité d'utilisation du logiciel			1 2 3 4 5	
Génère un API à partir du diagramme de classe	X			
Esthétique des diagrammes et de rétro conception			1 2 3 4 5	
Synchronisation d'une modification à un diagramme sur les autres diagrammes (refractoring)	X		1 2 3 4 5	
Supporte Java 1.5		X		Met Data type
Supporte UML 2.0	X			
Limitations du programme	X			30 jours, pas export
Langage de programmation supporté (autre que java)	X			C++, C#, VB
Qualité de la documentation			1 2 3 4 5	
Indépendance sur différentes plateformes				
Offre les options de Undo / Redo récursif		X		Pas undo/redo
Facilité d'installation	X			
Autres				
Bugs notables				
Commentaires	Catalogue de patrons de conceptions et possibilité de les appliquer Agrégation			

Describe


Describe Enterprise
File Edit View Insert Tools Window Help
Open
New
Save
Close
Source Control
Delete Element
Rename
Insert
Remove
Create Diagram From Selected Elements...
Show XML...
Generate Code
Divide Project...
Apply
Associate With...
Advanced Documentation...
Expand All Packages
Filter...
Refresh
Property Editor
Data
Package edu
Name edu
Alias edu
Visibility public
Stereotypes
Tagged Value
Documentation
Constraints
Source Code Di D:\guehene-c
Owned Eleme
Open Projects
20020804
New Project
edu
rice
cs
drjava
Documentation
Describe, the next generation UML design and development tool that offers both rich modeling features and
development utilities to simplify analysis, design, and implementation. With superior navigation, wizard-
of common tasks, and synchronous, round trip engineering, Describe Enterprise turns the UML into a live tool
and manipulating your code.
aggressively implements the emerging UML 2.0 standard. For more information on UML 2.0, please visit the
Group's web site at http://www.omg.org/uml.
ed to provide tips that will get you started using Describe and will allow you to quickly maximize your
Projects Pane
rise uses an XMI-based file structure. The modeling elements used to create your diagrams are stored in
ese stand-alone *.etd files are inserted into workspaces (*.etw) and appear in the Open Projects pane.
ve one workspace open at a time, but a workspace may consist of any number of projects. In addition,
es the ability to use modeling elements that are contained within one project in other projects within the
To use an element on a diagram in another project, simply drag the element from the Open Projects pane to your diagram. The element will indicate the project that it belongs to in its name compartment. To reference datatypes or classes in other projects, you can add one project as a reference library to another project. When doing so, the datatypes and classes contained in the referenced project can be used in the target project by selecting the types from the picklists appearing in the diagram area or the property editor. To add a reference library to a project, select the project in the Open Projects pane and add the library in the Property Editor. Click here for more information on using the Property Editor.
You can drag external documents (Microsoft Word documents, Excel spreadsheets, etc) from a Windows Explorer window and drop them onto items in the Open Projects pane to create an associated artifact. Double-click the associated artifact from the Open Projects pane to open the external document.
Choosing a Project Mode
Describe Enterprise provides three different modes for projects: analysis, design and implementation. Projects in
Analysis mode have full modeling functionality, but do not ensure that element names follow correct language syntax and
Pending Checkins
Check In Comments
Name Status
Design Center Open Projects
Source Editor Pending Checkins
Show Aliases : OFF Default Language : Java Mode : Implementation Layout : N/A Round Trip : ON
start
Embarcadero Describ...
Describe Enterprise
Local Disk (D:)
FR
10:57 AM

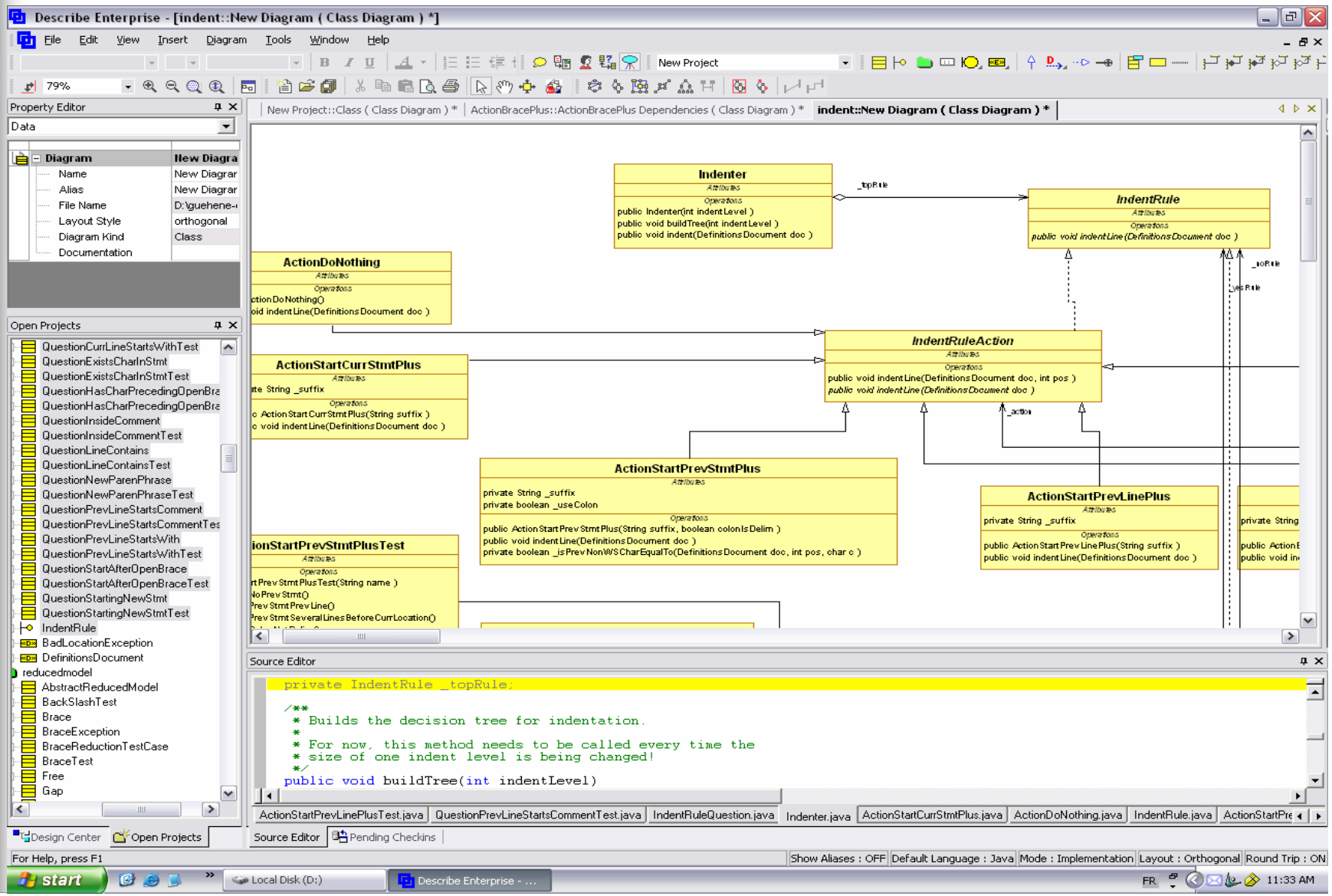
Describe

Vue globale

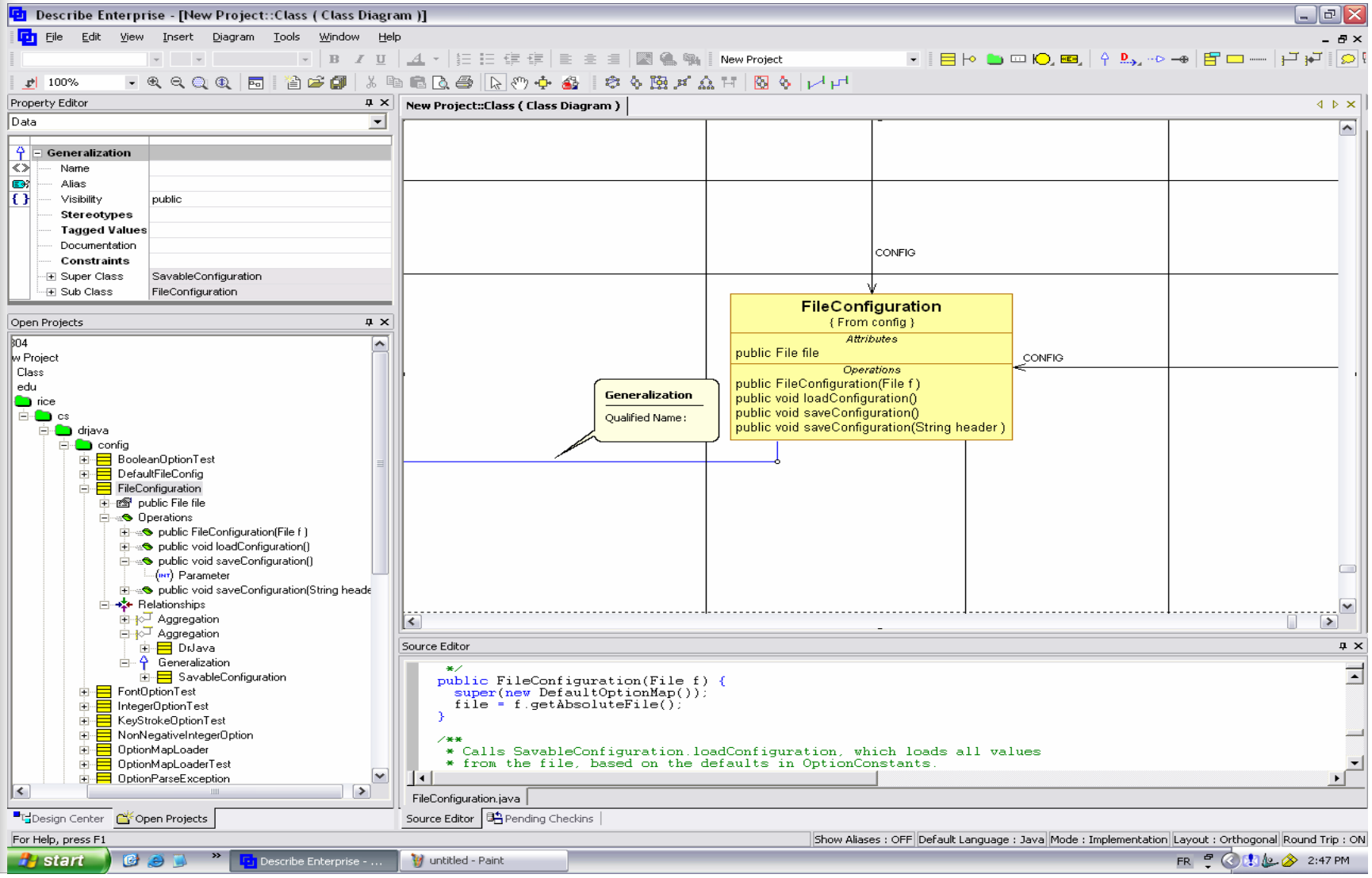


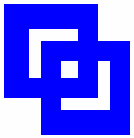
Describe

State



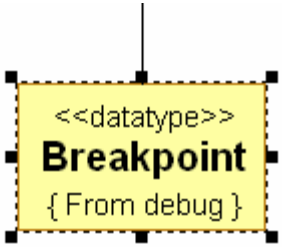
Navigation


Describe Enterprise - [New Project::Class (Class Diagram)]
File Edit View Insert Diagram Tools Window Help
New Project
Property Editor
Data
Generalization
Name
Alias
Visibility public
Stereotypes
Tagged Values
Documentation
Constraints
Super Class SavableConfiguration
Sub Class FileConfiguration
Open Projects
New Project::Class (Class Diagram)
CONFIG
FileConfiguration
{ From config }
Attributes
public File file
Operations
public FileConfiguration(File f)
public void loadConfiguration()
public void saveConfiguration()
public void saveConfiguration(String header)
CONFIG
Generalization
Qualified Name:
config
BooleanOptionTest
DefaultFileConfig
FileConfiguration
public File file
Operations
public FileConfiguration(File f)
public void loadConfiguration()
public void saveConfiguration()
Parameter
public void saveConfiguration(String heade
Relationships
Aggregation
Aggregation
DrJava
Generalization
SavableConfiguration
FontOptionTest
IntegerOptionTest
KeyStrokeOptionTest
NonNegativeIntegerOption
OptionMapLoader
OptionMapLoaderTest
OptionParseException
Source Editor
*/
public FileConfiguration(File f) {
super(new DefaultOptionMap());
file = f.getAbsoluteFile();
}
/**
* Calls SavableConfiguration.loadConfiguration, which loads all values
* from the file, based on the defaults in OptionConstants.
FileConfiguration.java
Design Center
Open Projects
Source Editor
Pending Checkins
For Help, press F1
Show Aliases : OFF
Default Language : Java
Mode : Implementation
Layout : Orthogonal
Round Trip : ON
start
Describe Enterprise - ...
untitled - Paint
2:47 PM

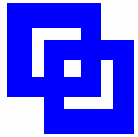


- **Avantages**
  - Présence des agrégations
  - Vue d'ensemble du diagramme de classes
  - Liens inter packages
  - Assez précis, on voit même certains patrons directement
  - Synchronisation automatique

- Désavantages:
  - Rétro-conception avec le diagramme de séquence ne fonctionne pas bien
  - Met DataType quand Java 1.5



  - Impossible de voir les classes internes sur le diagramme de classes

Eclipse vs  Describe

- Describe n’offre pas de fonctions telles que:
  - Call hierarchie
  - Type hierarchie
- Describe offre une version plug-in avec Eclipse
- Describe > aux autres outils testés

Les méthodologies utilisées et développées

- Recherche par vue globale
- Recherche naïve
- Recherche par héritage
- Recherche par relations des patrons

Méthodologie par vue globale

- Étapes
  1. Générer par l'outil de rétro-conception le diagramme de classes
  2. Naviguer dans le code en espérant trouver des micro architectures similaires à un patron de conception
- Simple
- Faible taux de succès

Méthodologie naïve

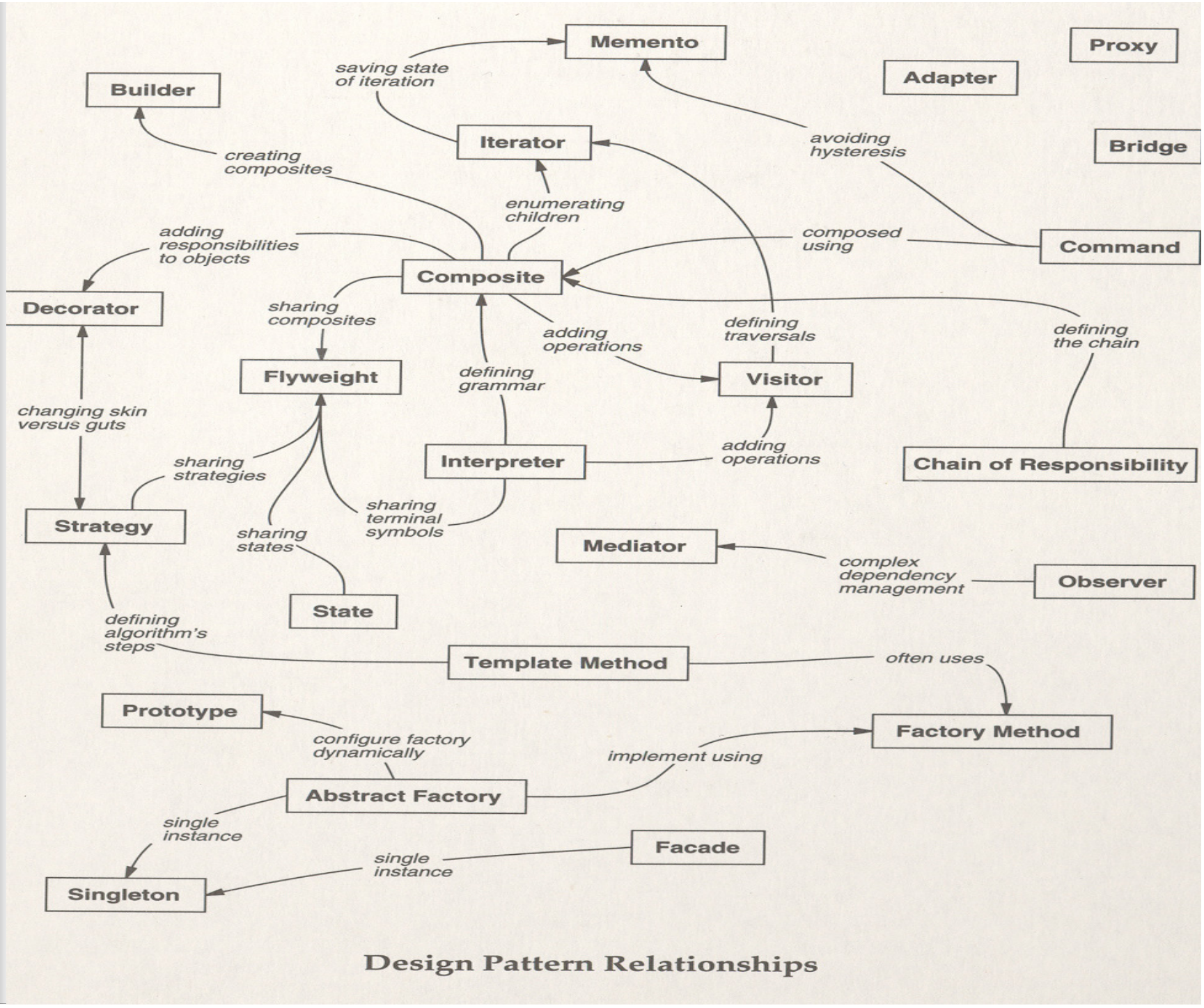
- Étapes
  1. Rechercher parmi toutes les classes, les noms liés aux patrons de conception
  2. Graviter autour des classes qui portent un nom significatif et vérifier si un patron de conception lui est associé
- Méthodologie grandement exploitée par nos prédécesseurs
- Efficacité restreinte

Méthodologie par héritage

- Étapes
  1. Rechercher les classes abstraites et les interfaces
  2. Graviter autour de ces classes afin de trouver des liens entre d'autres classes qui pourraient être intéressantes
  3. Lorsque la structure nous donne l'idée d'un ou de plusieurs patrons de conceptions, en discuter en groupe
  4. Passer à d'autres classes abstraites après être satisfait des recherches autour de cette classe
- Méthodologie grandement efficace
- Cerne rapidement les patrons de la classes « Behavioral »

Méthodologie par relations des patrons

- Étapes
  1. Sélectionner un patron de conception déjà trouvé
  2. Identifier des liens potentiels entre le patron sélectionné et des classes avoisinantes, basé sur le schéma des relations
  3. Ensuite, sélectionner un autre lien potentiel jusqu'à l'épuisement des liens
  4. Continuer ainsi sur tous les patrons que nous avons identifiés
- Méthodologie efficace, complémentaire aux autres



Design Pattern Relationships

Méthodologie par relations des patrons

- Étapes
  1. Sélectionner un patron de conception déjà trouvé
  2. Identifier des liens potentiels entre le patron sélectionné et des classes avoisinantes. Basé sur le schéma des relations
  3. Ensuite, sélectionner un autre lien potentiel jusqu'à épuisement des liens
  4. Continuer ainsi sur tous les patrons que nous avons identifiés
- Méthodologie efficace, complémentaire aux autres

Résumé des méthodologies

- Méthodologie par vue globale
  - Très faible compréhension des patrons de conception et limitée par le potentiel des outils utilisés à bien modéliser les classes
  - Très peu de trouvailles
  - Aucune base solide

Permettre de réaliser qu'il nous faut développer une méthodologie plus robuste

Résumé des méthodologies

- Méthodologie naïve
  - Déduction facile des emplacements de patrons de conception grâce aux noms
  - Non fiable dans des programmes sans noms représentatifs
  - Méthode peu utilisée

Méthodologie plus constructive que la précédente mais peu exploitée du fait que les trouvailles étaient peu constructives et a été exploitée par les équipes de recherches antérieures

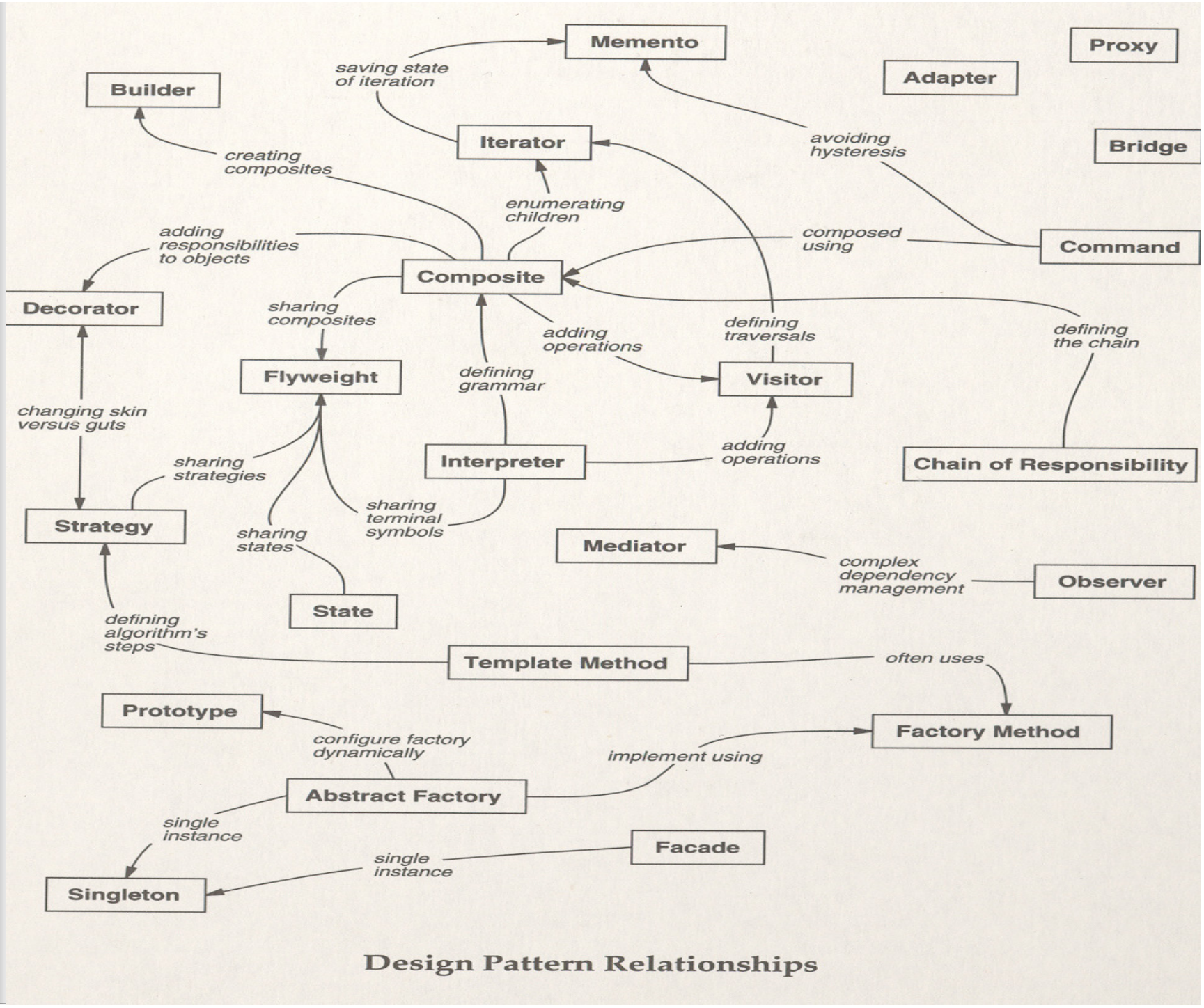
Résumé des méthodologies

- Méthodologie par héritage
  - La méthodologie exploitée grâce à nos outils de rétro-conception
  - Compréhension plus profonde des patrons de conceptions
  - Trouvailles plus concentrées dans la section « Behavioral Pattern »
  - Donne les résultats les plus valorisants

Cette méthodologie est la première que nous considérons assez robuste pour l'appliquer dans n'importe qu'elle recherche

Résumé des méthodologies

- Méthodologie par relations des patrons
  - Approfondie notre compréhension des patrons de conception et de leurs relations
  - Résultat de recherche de patrons de conception autre que dans la section « Behavioral Pattern »
  - Limite
    - Recherche de patrons proportionnels:
      - Aux patrons trouvés précédemment
      - Aux relations possibles des patrons trouvés précédemment (parfois nulle)
    - Recherche parfois peu constructive du fait que les relations entre patrons ne sont pas utilisées



Design Pattern Relationships

Deux versions de Dr Java et une vingtaine de patrons

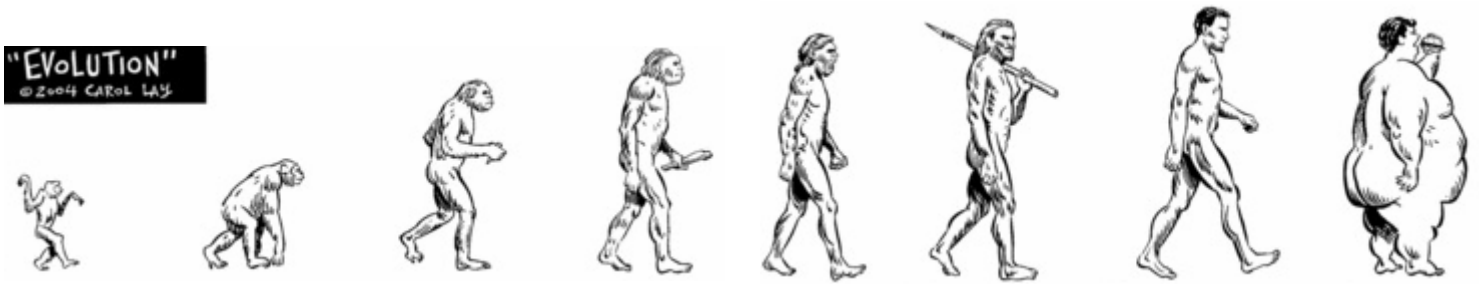
20020703	20020804
Factory method	Factory method
Adapter	Adapter
Bridge	Bridge
Proxy	Proxy
	Mediator
State (3)	State (3)
Strategy (2)	Strategy (2)
Template method (7)	Template method (9)

Résultats dans les plus anciennes versions

20020619-20020703-20020804	Nombre de patrons
Factory method	1
Singleton	8
Adapter	2
Bridge	1
Proxy	1
Command	2
Iterator	1
Mediator	1
Momento	1
State	3
Strategy	3
Template method	9

Évolution des patrons

- Résultat des modifications notées entre les versions:
  - 2002-06-19
  - 2002-08-04
  - 2004-03-26



Résultat des modifications

- **Mediator (3901):**
  - La première version pas de patron
  - Dans la version intermédiaire, le concreteMediator et le Mediator étaient la même classe
  - Dans la dernière version, DebugManager s'est divisé en deux: Debugger et JPDADebugger
- **State (2085) :** Ajout de 3 concreteState
- **Strategy (2070) :** Ajout de 2 concreteStrategy
- **State (2079) :** Ajout d'un concreteState

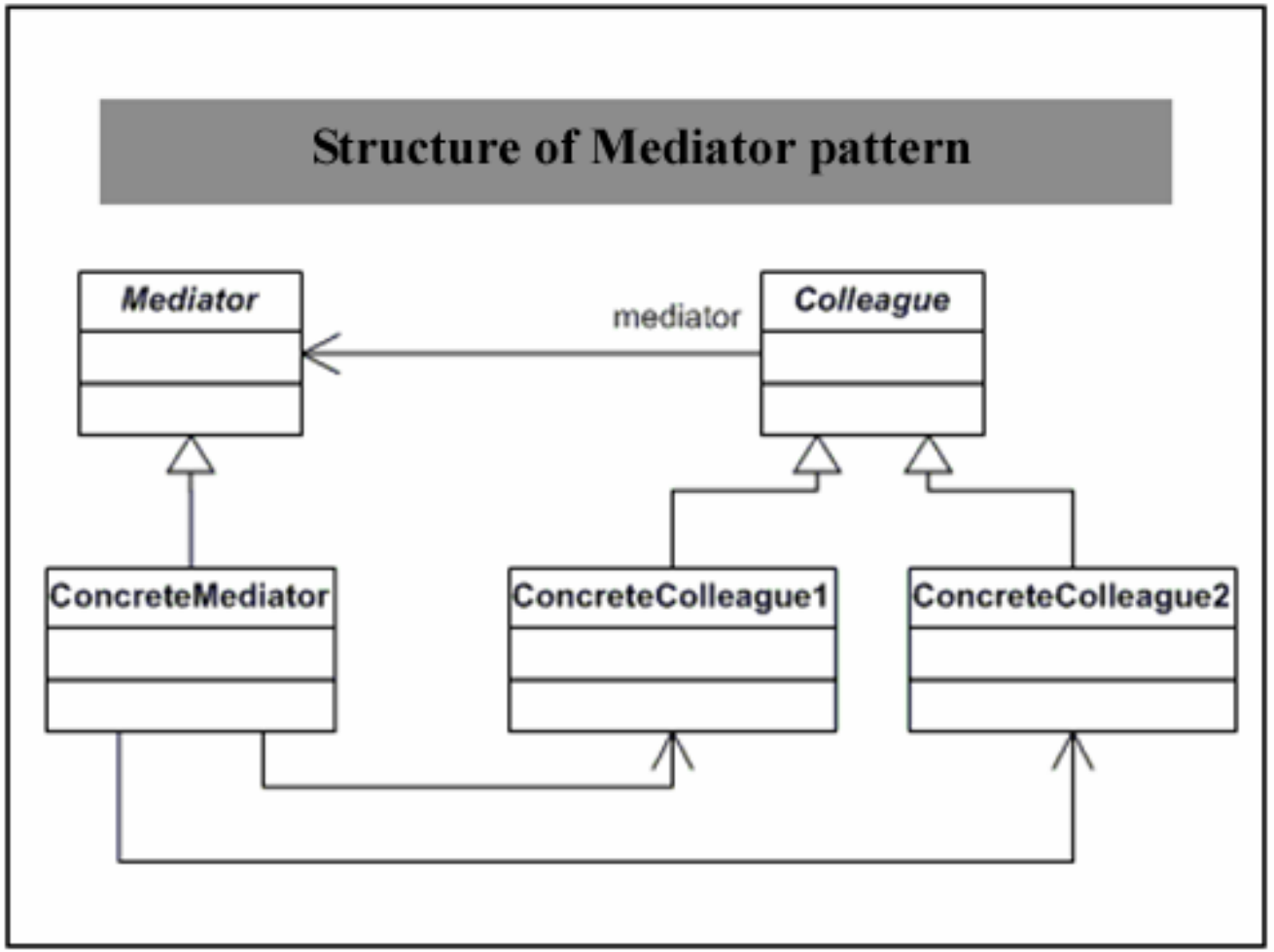
Présentation des évolutions

- Médiateur **(3901)**
  - Changement dans les 3 versions
- State **(2085)**
  - Changement entre les 2 dernières versions

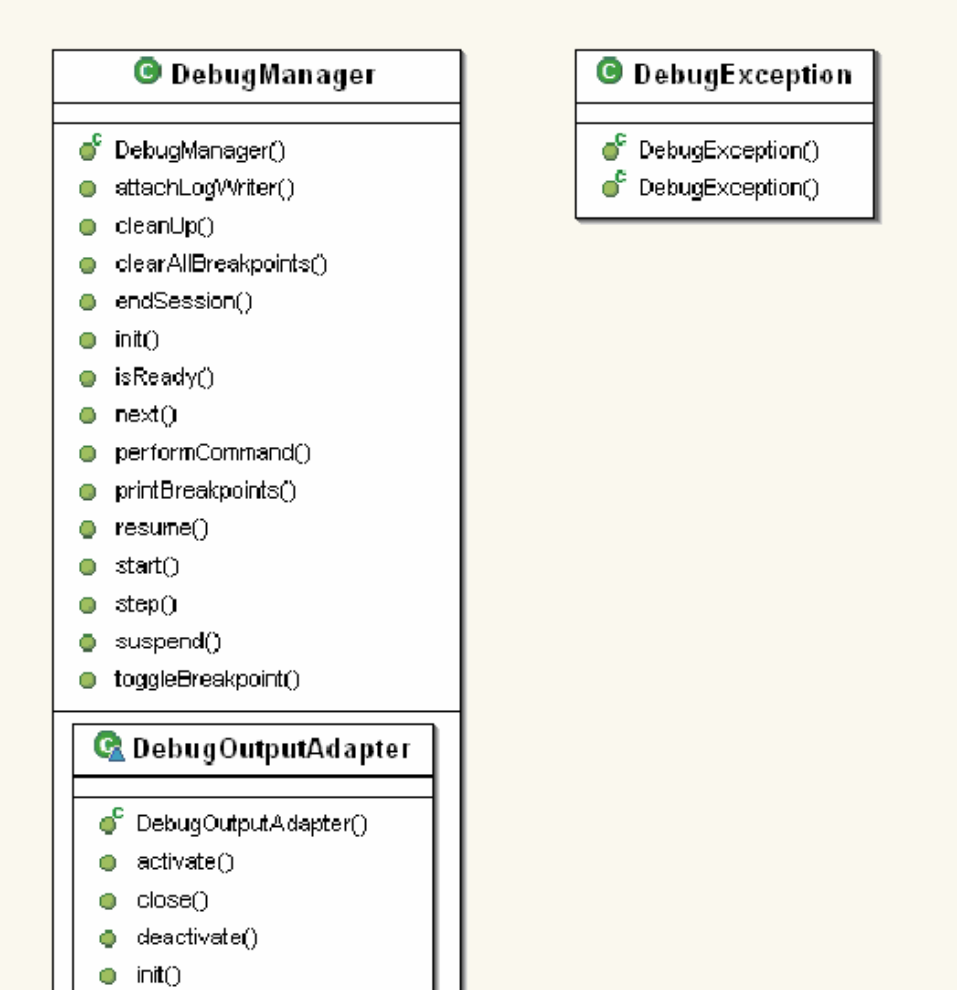
Évolution d'un patron de conception

- Médiateur
  - Qu'est-ce qu'un Médiateur ?
    - Utilités
  - Dans notre analyse
    - Évolution de la version 2002-06-19 à 2004-03-26
    - Spécialisation des classes

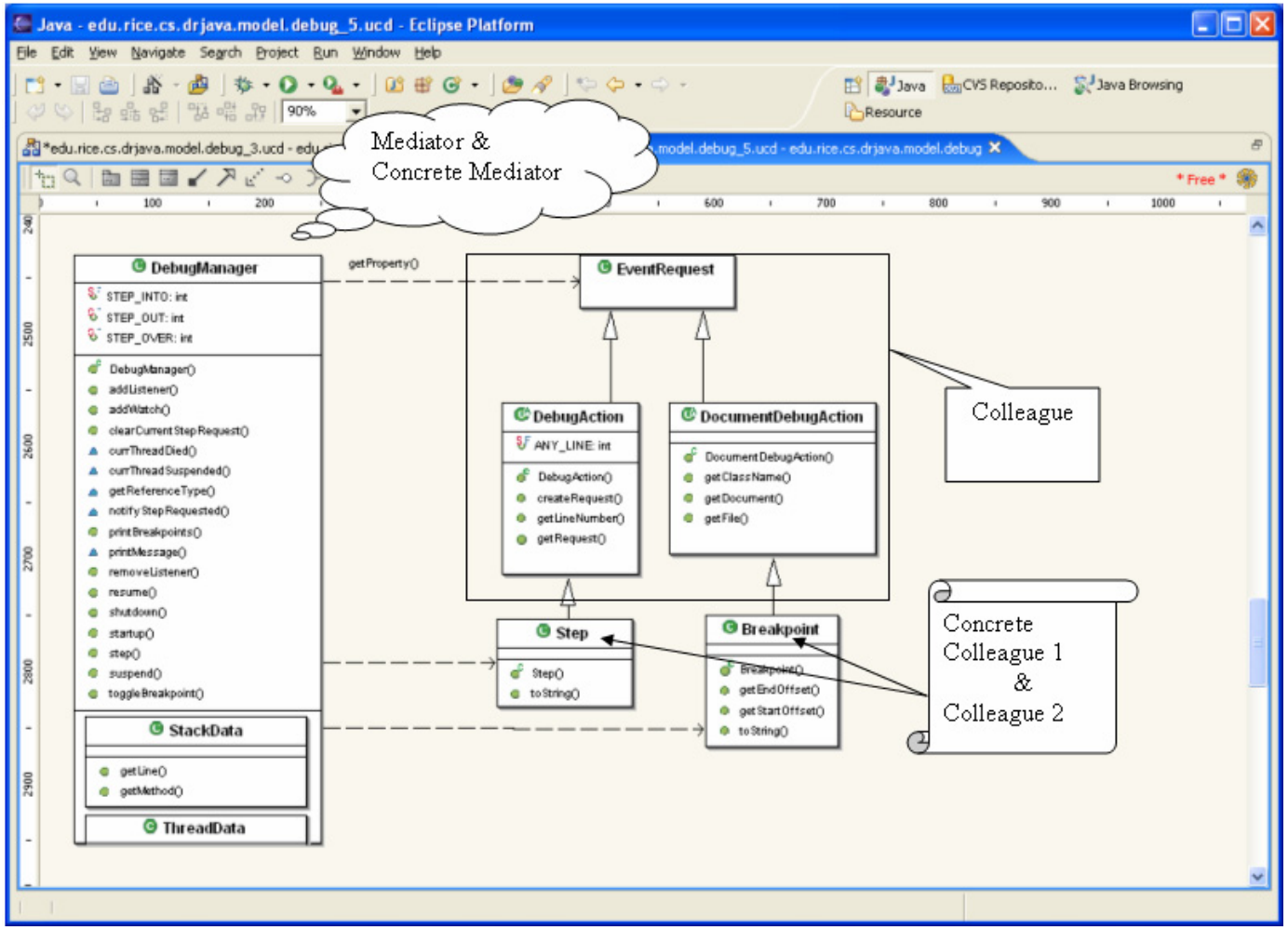
Qu’est-ce qu’un Médiateur



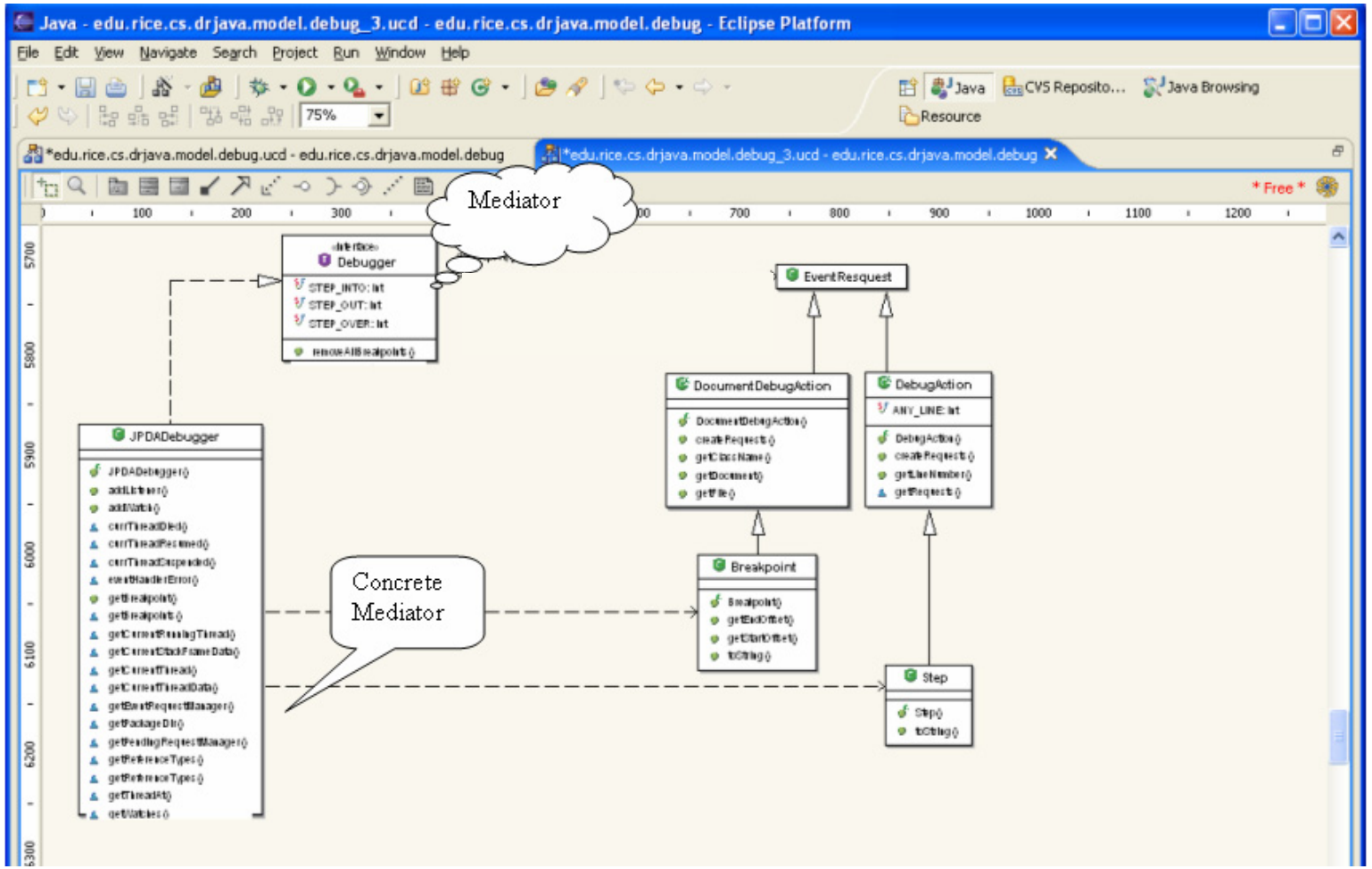
Dr Java 2002-06-19 Médiateur manquant



Dr Java 2002-08-19 Médiateur Mediator & ConcreteMediator regroupés dans la même classe



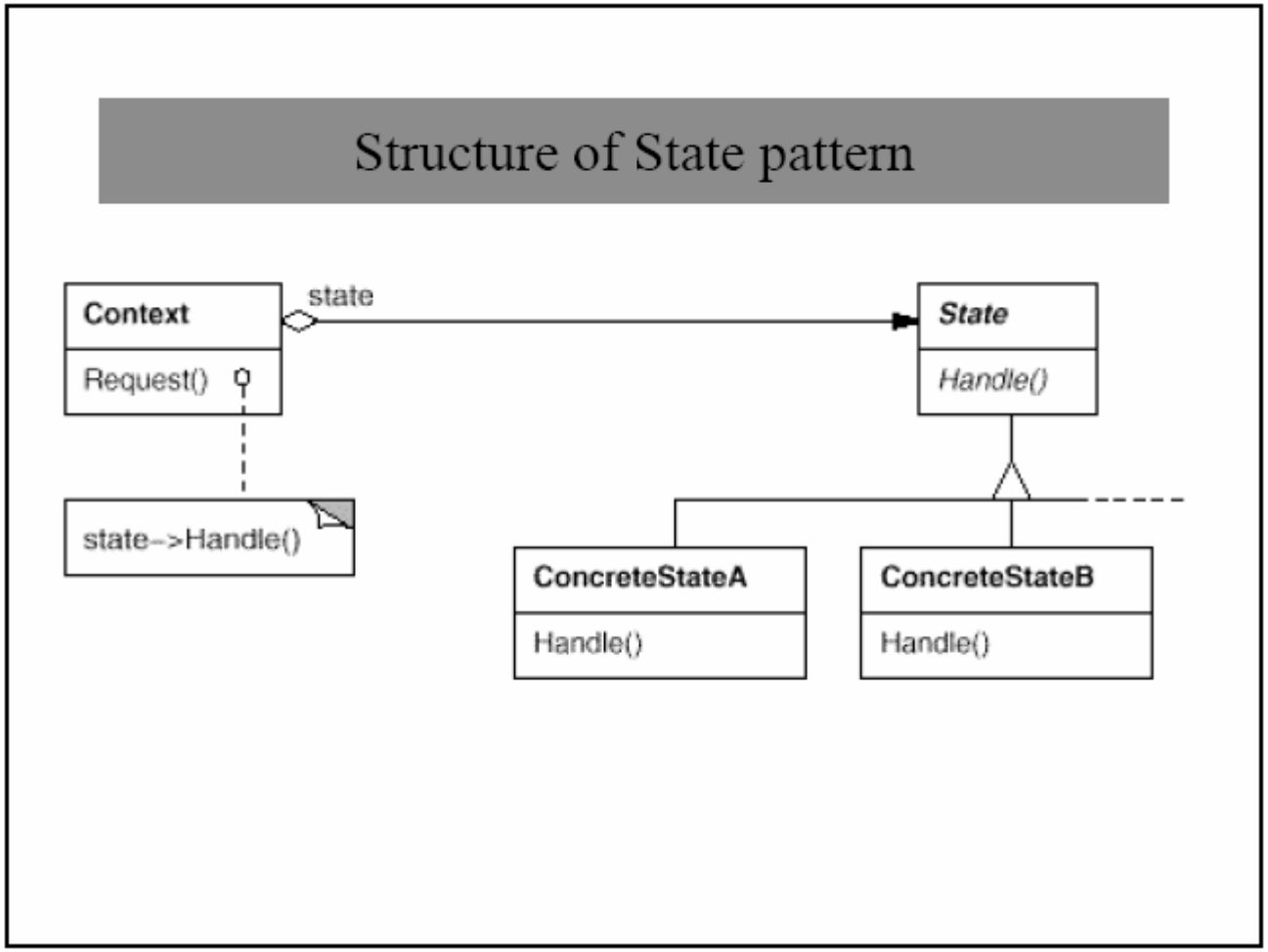
Dr Java 2004-03-26 Médiateur Scindement du Mediator et du ConcreteMediator



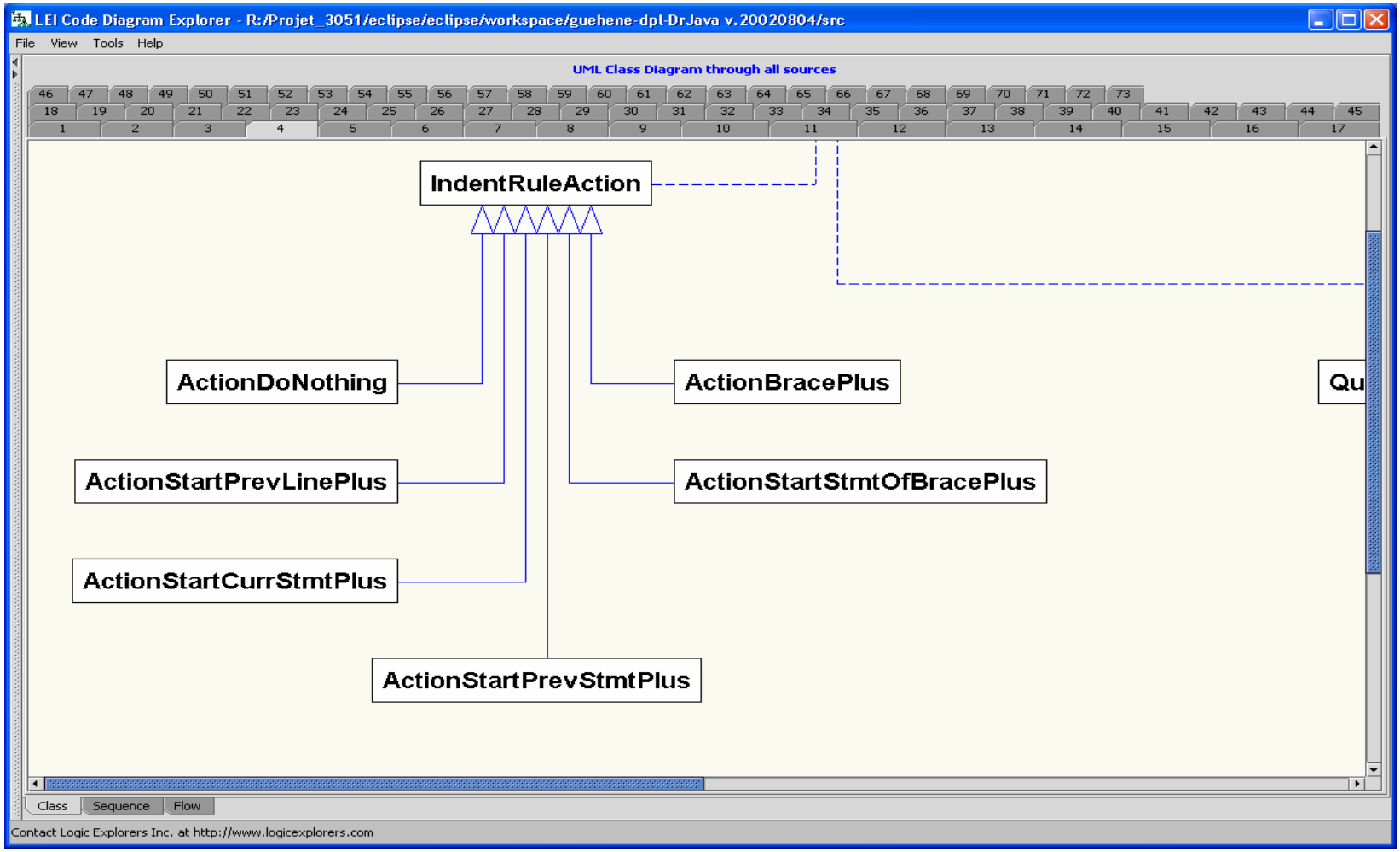
Évolution d'un patron de conception

- State
  - Qu'est-ce qu'un State ?
    - Utilités
  - Dans notre analyse
    - Évolution de la version 2002-08-04 à 2004-03-26
    - Ajout de « Concrete States »

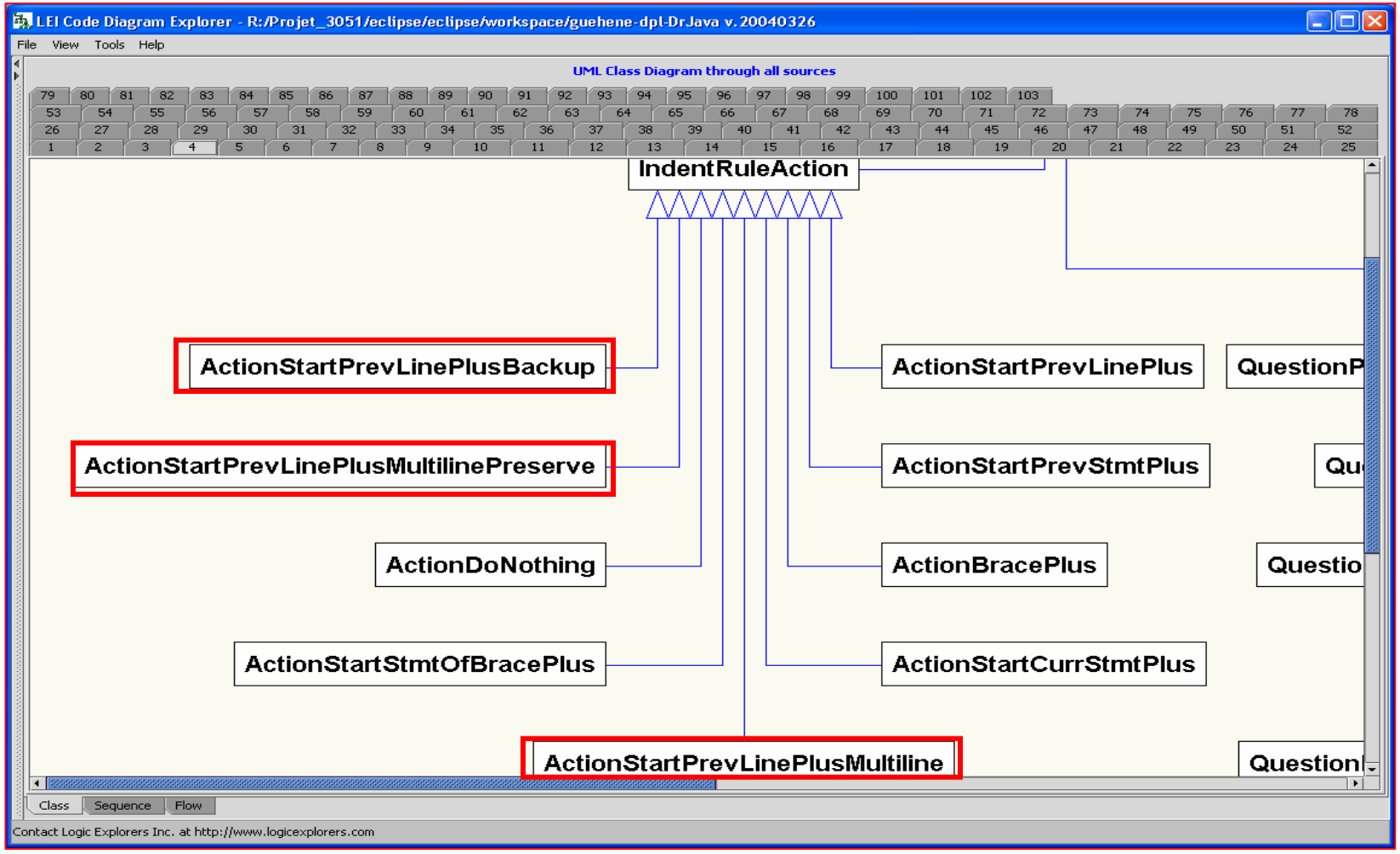
State : Qu'est-ce qu'un state



Version 2002-08-04



Version 2004-03-26



Dans notre analyse

- Réussite de l'utilisation du patron de conception
  - Ajout de « Concrete State » d'une version à une autre
  - Intuition :
    - Nouvelle fonctionnalité du programme (état de ligne multiple – « Multiline »)
    - Ajout de 3 nouveaux états pour gérer cette fonctionnalité

Conclusion

- Les outils utilisés
- La méthodologie utilisée
  - Générale
    - Par vue Globale
    - Naïve
    - Par héritage
    - Par relations des patrons
  - Meilleur méthode
    - Opportuniste !!!!!

Conclusion

- Appréciation personnel
  - Apprentissage lors du projet
  - Difficultés rencontrées
    - Support de Java 1.5
    - Licences des outils
  - Utilité des résultats XML
    - Calibrer Ptidej
- Période de questions

Fin