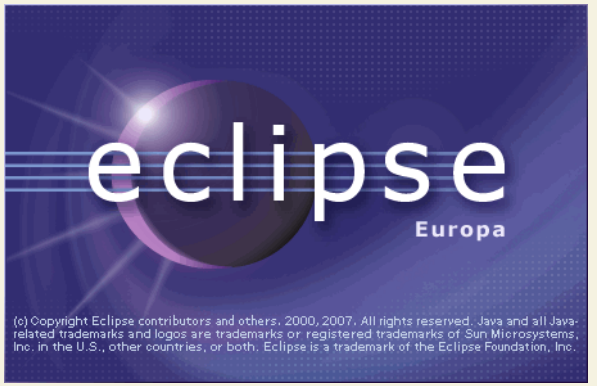
Université de Montréal

EFREI
ECOLE D'INGENIEURS DES TECHNOLOGIES DE L'INFORMATION ET DU MANAGEMENT

Projet IFT3150 : Eclipse Connector

PEPS – Ptidej for EcliPSe



Nelson Cabral - CABN09078501
nelson.cabral@gmail.com

Plan

- Sujet
- Ptidej & Eclipse
- Précédent essai
- Présentation de la solution
- Architecture générale
- Créer une extension
- Difficultés rencontrées
- Perspectives et améliorations

Contexte : Ptidej

- Suite d'outils de rétro-conception
- Modélisation de programmes
- Analyses sur ces modèles : détection des design patterns, des anti-patterns...
- Interface graphique Swing qui permet d'utiliser cette suite d'outils

Contexte : Eclipse

- Plate-forme de développement de référence en Java
- Multi-langages : Java, C/C++, Cobol
- Multi-plateformes : Mac OS X, Linux, Windows
- Repose sur un mécanisme d'extensions => conçu pour être modulaire et extensible

Sujet

- Etablir une interface entre Eclipse et Ptidej pour pouvoir appeler les fonctionnalités de Ptidej depuis Eclipse
- Après étude, le plus simple était de créer un plugin

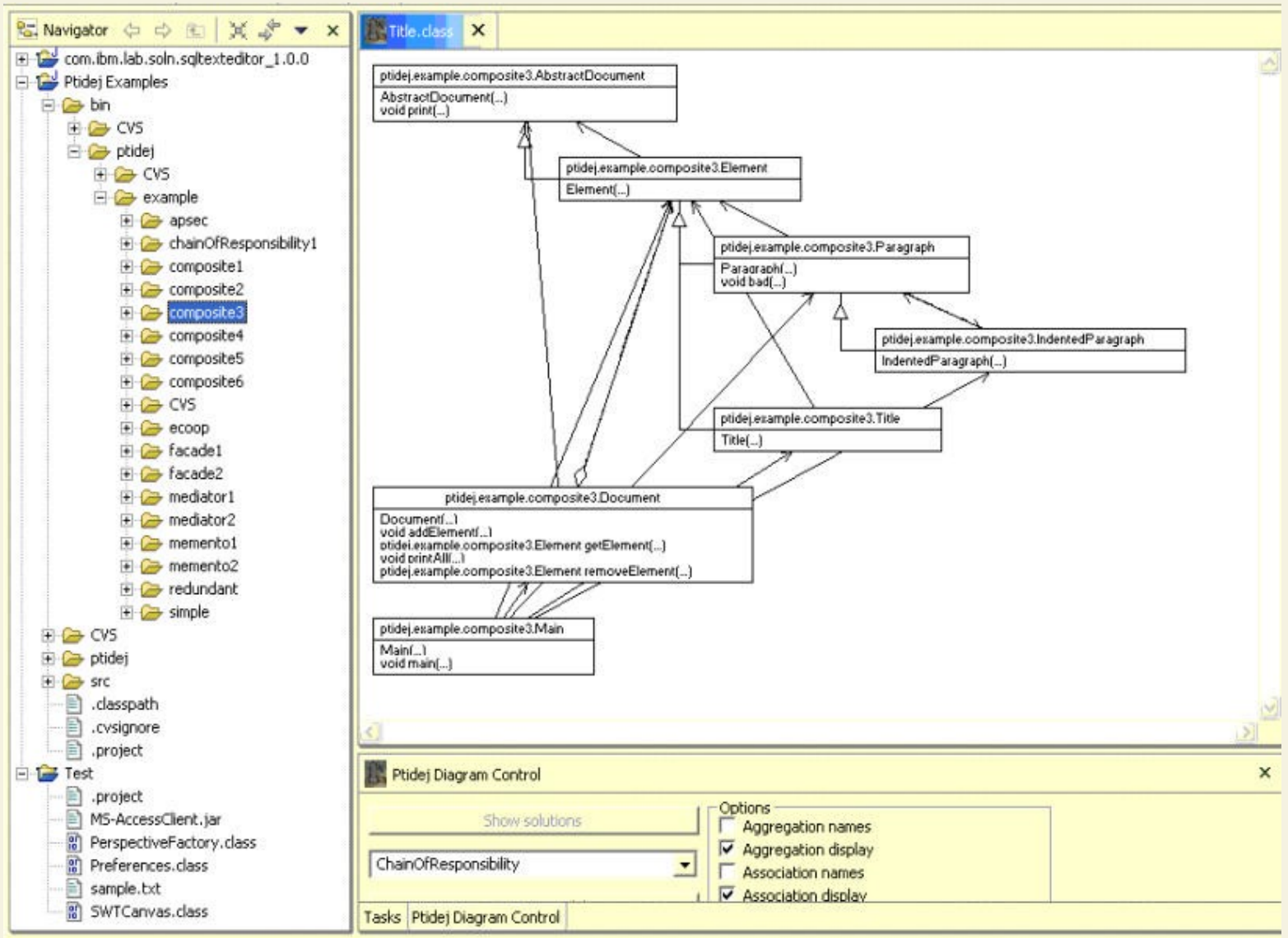
Précédent essai

- En 2004, Driton Salihu et Lulzim Laloshi ont aussi cherché a effectuer ce projet
- Résultat : un plugin pour Eclipse 2 qui n'assurait que très peu de fonctionnalités, seulement le chargement des classes marchait
- L'utilité de Ptidej réside dans l'analyse du modèle généré
- Cf. Interview de Y.-G. Guéhéneuc

Précédent essai

- Difficultés qu'ils avaient rencontrées :
  - Planifier les tâches
  - Trouver de la documentation
  - Manque de temps
  - Manque d'expérience sur le fonctionnement interne d'Eclipse et de Ptidej
- Cependant leur travail m'a été d'une grande utilité, me servant de point de départ sur l'orientation à prendre

Précédent essai

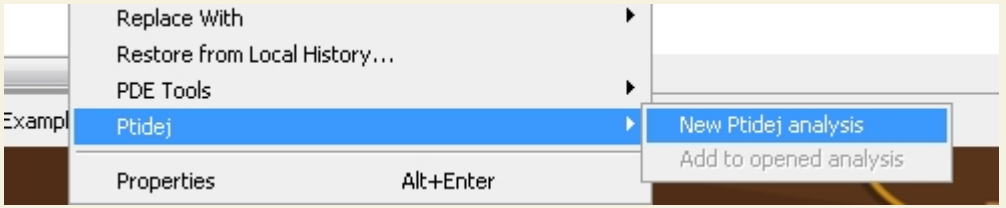


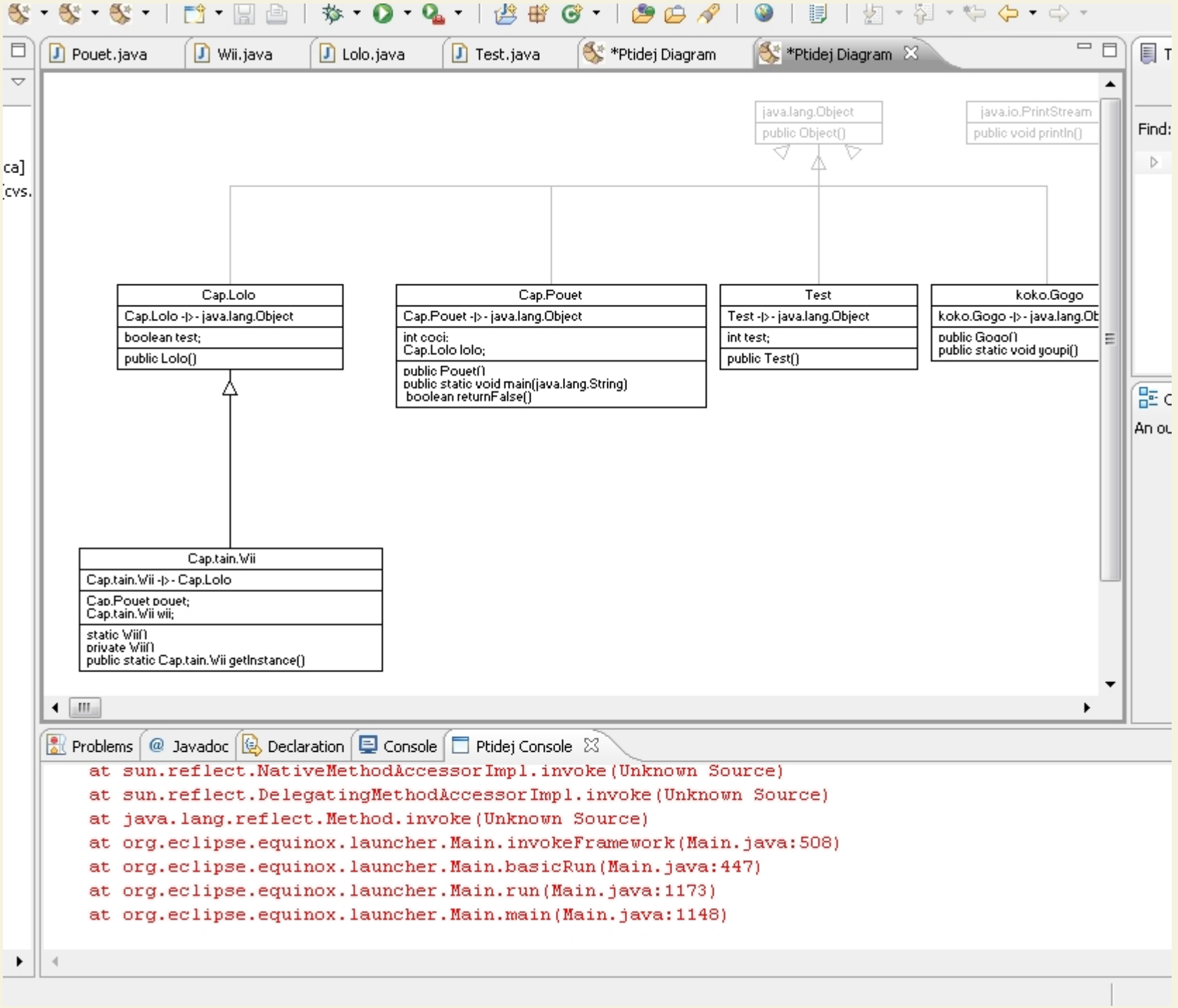
PEPS – Ptidej for EcliPSe

- Création d'un nouveau plugin : PEPS – Ptidej for EcliPSe (en français : Ptidej pour EcliPSe)
- Deux projets :
  - PEPS : gestion du modèle du plugin, outil de retro-ingénierie
  - PEPS Extensions : ensemble des outils d'analyses

Fonctionnalités (démonstration)

- Création de projet Ptidej
  - sélectionnez des classes, des packages ou des projets dans le Package Explorer d'Eclipse
  - lancez une analyse dessus!



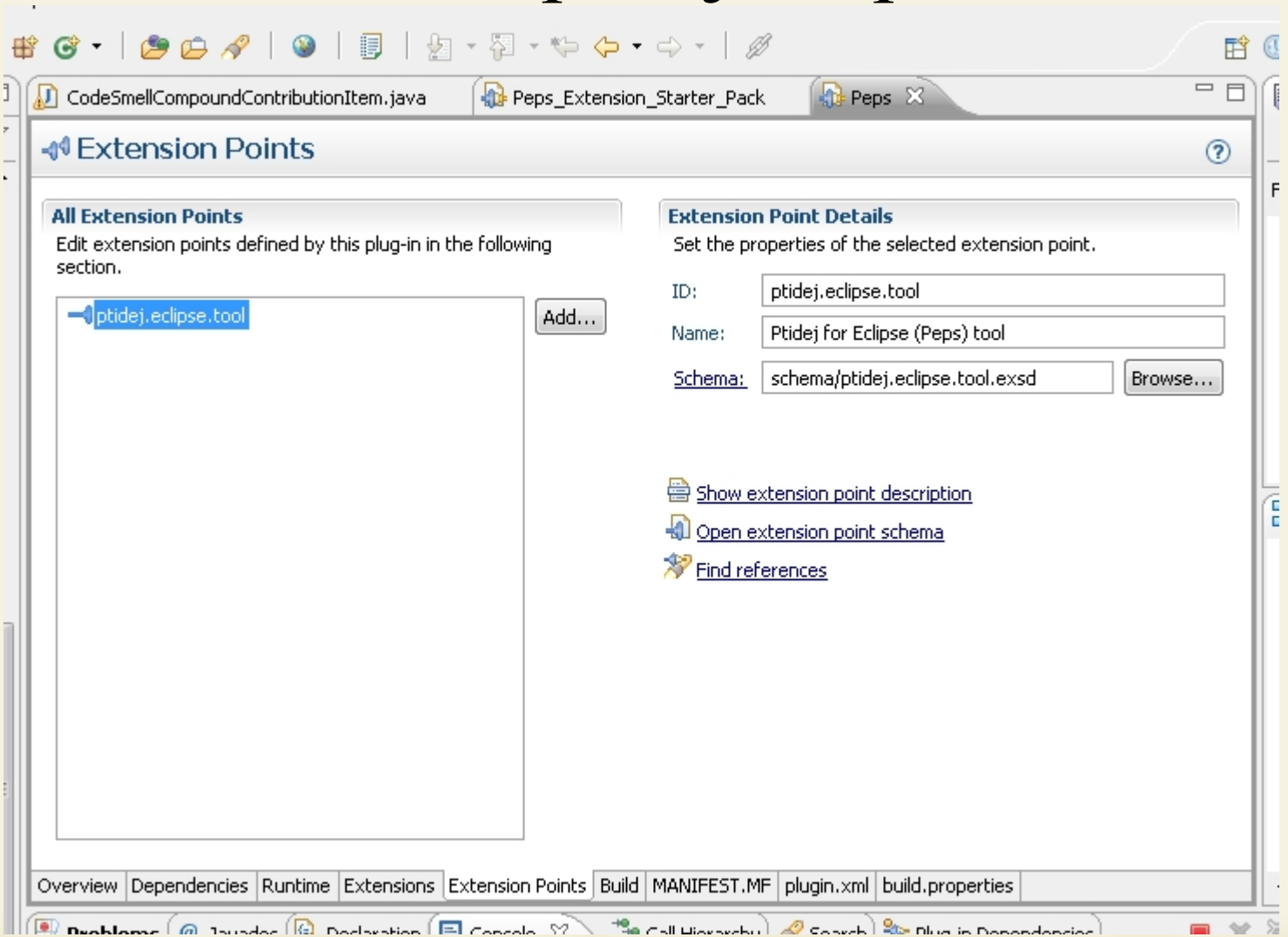

Pouet.java
Wii.java
Lolo.java
Test.java
*Ptidej Diagram
*Ptidej Diagram
java.lang.Object
public Object()
java.io.PrintStream
public void println()
Cap.Lolo
Cap.Lolo -|> - java.lang.Object
boolean test;
public Lolo()
Cap.Pouet
Cap.Pouet -|> - java.lang.Object
int coci;
Cap.Lolo lolo;
public Pouet()
public static void main(java.lang.String)
boolean returnFalse()
Test
Test -|> - java.lang.Object
int test;
public Test()
koko.Gogo
public Gogo()
public static void youpi()
Cap.tain.Wii
Cap.tain.Wii -|> - Cap.Lolo
Cap.Pouet pouet;
Cap.tain.Wii wii;
static Wii()
private Wii()
public static Cap.tain.Wii getInstance()
Find:
Problems
Javadoc
Declaration
Console
Ptidej Console
at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
at java.lang.reflect.Method.invoke(Unknown Source)
at org.eclipse.equinox.launcher.Main.invokeFramework(Main.java:508)
at org.eclipse.equinox.launcher.Main.basicRun(Main.java:447)
at org.eclipse.equinox.launcher.Main.run(Main.java:1173)
at org.eclipse.equinox.launcher.Main.main(Main.java:1148)

Fonctionnalités (démonstration)

- Deux vues :
  - Editeur : représentation graphique du modèle
  - Console : messages envoyés par le plugin
- Ajout de classes à un projet déjà ouvert
- Sauvegarde de projet Ptidej (.ptidej)
- Chargement de projet (.ptidej)

Fonctionnalités (démonstration)

- Point d'extension : ptidej.eclipse.tool

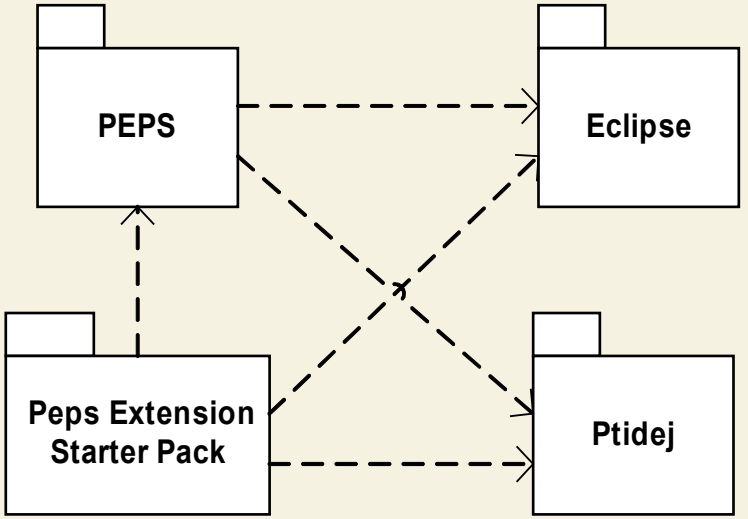


Fonctionnalités (démonstration)

- Premier set d'extensions : Ptidej Extension Starter Pack
  - Gestion des éléments visibles
  - Visiteurs
  - Micro-patterns
  - Code smells
  - Outils d'analyse PADL
  - Design Patterns

Architecture (démonstration)

- Deux projets : PEPS et Peps Extension Starter Pack



Architecture (démonstration)

- PEPS :
  - Une partie indépendante d'Eclipse, surcouche de Ptidej
  - Une partie dépendante d'Eclipse, ptidej.eclipse.*
- Peps Extension Starter Pack
  - Implémentations du point d'extension ptidej.eclipse.tool

Créer une extension (Démonstration)

- Configurer l'extension pour dépendre de ptidej.eclipse.tool
- Implémentation du code métier : IptidejTool, point d'accès unique au modèle et au canvas
- Intégration dans Eclipse : Commands, Handlers & Menus

Difficultés rencontrées

- Faire un planning + manque de temps
- Eclipse 2 => manque de documentation
- Java 1.4 => habitué à Java 6
- Ptidej : manque de documentation (Javadoc? Snippets d'exemple?)

Solutions

- Etablissement d'un planning serré (prototype fonctionnel en deux mois)
- Eclipse 3 : derniers outils + documentation => gain de productivité
- Java 6 : gain de productivité
- Ptidej : lecture du code du viewer
- Bonus : création de documentation sur le plugin (Javadoc + rapport)

Perspectives et améliorations

- Déboguer et consolider PEPS :
  - Ajouts de vues supplémentaires
  - Revue du code existant et amélioration
- PEPS Extensions
  - Intégrer les outils restants
- Cf. Notes en fin de rapport

Conclusion