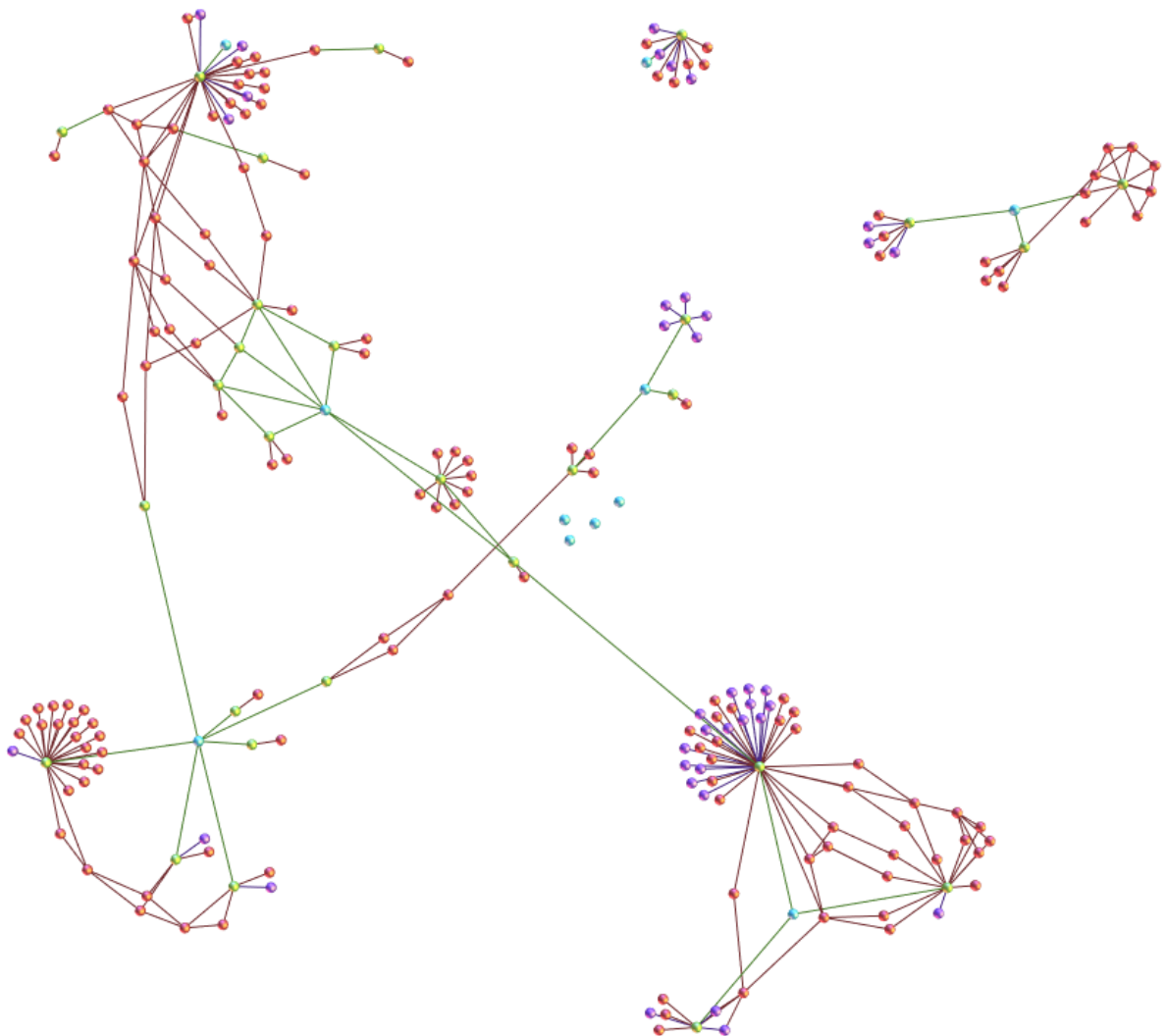


CodeMiner

Microscope numérique pour les logiciels

« Révéler l'organisation interne de ces systèmes complexes »

Rapport final



Elèves-ingénieurs

Jérémy PALMIER

Sébastien HEYMANN

Suiveur UTC

Mr Antoine JOUGLET

Remerciements :

Nous tenons à remercier Mr Antoine Jouglet, notre suiveur, pour avoir eu l'audace d'accepter un sujet de projet venant d'étudiants, et pour nous avoir donné carte blanche sur son déroulement.

Un grand merci aussi à Mathieu Bastian, responsable de Gephi actuellement en GI04, pour sa grande disponibilité et le chamboulement de son planning de travail pour nous offrir la possibilité de réaliser un plugin sur un logiciel toujours en version alpha.

Index

I.Contexte.....	5
1.Rappels.....	5
2.Motivations.....	5
3.Objectifs.....	6
II.Réalisation.....	7
1.Etat d'avancement.....	7
Fonctionnalités.....	7
Architecture globale.....	7
Plugin Eclipse.....	8
Plugin Géphi.....	11
2.Méthodologie.....	12
3.Environnement et outils.....	13
4.Retour sur les premiers usages.....	13
III.Gestion du projet.....	16
1.Organisation.....	16
2.Planning.....	17
IV.Conclusion.....	17
1.Rappel des points essentiels.....	17
2.Bilan.....	17
3.Perspectives du projet CodeMiner.....	18
V.Annexes.....	19
1.Glossaire.....	19
2.Annexe B : Contrat de licence GNU/GPL 3.....	20
3.Annexe C : Témoignage d'usage.....	26
4.Annexe D : Liste des relations entre éléments représentées.....	27

Index des illustration

Illustration 1: Vue globale sur l'architecture du système.....	8
Illustration 2: Architecture du système de plugins de Eclipse.....	9
Illustration 3: Sélection d'un projet, ici le projet CodeMiner.....	9
Illustration 4: Bouton de démarrage.....	9
Illustration 5: Paramètres généraux.....	10
Illustration 6: Paramètres d'export GDF des éléments.....	10
Illustration 7: Paramètres du parseur Java.....	11
Illustration 8: Représentation du graphe.....	11
Illustration 9: Panneau de contrôle et recherche.....	12
Illustration 10: Graphe des méthodes de Géphi, vu sous Géphi standard.....	14
Illustration 11: PhotoViz à t + 3 semaines « Intégration des parties ».....	15
Illustration 12: PhotoViz à t=début de développement « chacun travail dans son coin ».....	15
Illustration 13: Fragment principal du plugin Eclipse de CodeMiner, vu avec Géphi standard	15
Illustration 14: Détail de la vue du plugin Eclipse de CodeMiner.....	16
Illustration 15: Diagramme de GANTT simplifié.....	17

I. Contexte

1. *Rappels*

La fabrication d'un logiciel quelle qu'il soit fait appel à des méthodes de représentation de son fonctionnement avant, pendant et après la phase de développement.

La vue générale sur la structure est fournie par des schémas de fonctionnement qui synthétisent et réduisent les informations dans un formalisme de description, dont les plus célèbres sont l'UML (Unified Modeling Language) et le Merise francophone. Ils permettent entre autre de représenter les interactions entre composants, classes ou packages du logiciel pour mettre au point une logique de fonctionnement.

Le découpage en différentes classes et packages a ainsi pour objectif, entre autre, d'en isoler des modules dont le couplage minimal leur assure des "rôles" différents et complémentaires. Cette isolation des variables et fonctions permet de clarifier la logique de fonctionnement, limiter les dépendances et ainsi augmenter considérablement la durée du cycle de développement au bout duquel toute tentative d'évolution provoque une régression. Ce n'est ni plus ni moins qu'une méthodologie pour réduire et maîtriser les interactions fonctionnelles entre variables, regroupées dans des ensembles qui font sens pour l'individu. On parle en effet de "complexité" d'un programme non sans raison, car le produit de ces interactions inscrites dans le code source entraîne à un moment une perte de la vue synoptique (à la fois synthétique et détaillée) sur celui-ci, et donc sa maîtrise. Le reverse engineering UML ne peut alors dessiner qu'une vue globale sur la structure, mais il n'est pas possible de la lire et encore moins de se la approprier dans le détail. D'autre part les méthodologies du génie logiciel ne sont pas toujours strictement respectées pour diverses raisons, nous pensons donc que la visualisation de la structure des logiciels est une piste pour palier aux déficiences de choix de conception et d'erreurs de modularité en cours de développement.

CodeMiner est un projet de chaîne de traitement logiciel visant à représenter un code source sous forme de graphe dirigé (noeuds/arcs) dans un espace 2D. Il a pour but de donner à voir les relations entre packages, classes, méthodes et variables existantes dans un code source Java.

2. *Motivations*

- Une curiosité scientifique : maintenant que la technologie nous permet de visualiser des graphes de plusieurs dizaines de milliers de nœuds et notamment des graphes du Web, nous nous sommes demandé s'il était possible de représenter un code source logiciel sous cette forme. Le code source s'appréhende ainsi en tant que manifestation d'un système distribué (comme en biologie, en physique, en sociologie ou en ingénierie);
- Les méthodologies du génie logiciel ne sont pas toujours strictement respectées pour diverses raisons. Nous pensons que la visualisation de la structure du code source est une piste pour palier aux déficiences de choix de conception et d'erreurs de modularité en cours

de développement, pour les détecter plus tôt et pour mieux les corriger.

3. Objectifs

Il s'agit d'élaborer des outils de construction automatique de graphes représentant les dépendances entre classes et fonctions, entre classes, et entre fonctions, dans le but de rendre de nouveau intelligible aux architectes et développeurs la structure des programmes qu'ils produisent dans leur niveau de détail le plus fin.

Dans ce cadre, l'objectif de cette UV TX est de concevoir et développer un outil d'aide à la réalisation de projet informatique dont l'objectif est de donner à voir des unités du code source de différents niveaux de précision (fichier > classe > fonction > variable : multigranularité) et leurs relations de dépendance sous forme de graphes. L'outil est pensé pour être multilingage de programmation, mais seul le Java est traité. A mi-chemin entre la production d'un logiciel fonctionnel et d'une activité de recherche, l'étudiant a à définir des concepts et une grammaire graphique pour les articuler, en vue de concevoir une visualisation des données cohérente et exploitable aussi bien par des développeurs que par des architectes logiciel. L'accent est mis sur la mise en place d'un système de parsing de documents sources, de rétention des données extraites puis de production d'un fichier de graphe.

Le projet se décompose ainsi en 4 phases de développement :

1. Le parsing des fichiers de code source Java
2. La rétention des données extraites : stockage permanent ou temporaire ?
3. L'export des données dans un format lisible par un logiciel de visualisation (GDF pour GUESS et Géphi)
4. L'adaptation de ce logiciel aux besoins spécifiques d'interaction utilisateur

Le livrable est l'ensemble des codes sources réalisant les trois premières étapes ainsi que l'exécutable de la dernière phase (si possible).

II. Réalisation

1. *Etat d'avancement*

Au terme de la TX, tous les objectifs sont atteints. Le plugin Eclipse peut parser n'importe quel projet Java et produit un fichier GDF qui sera ensuite lu par Géphi et le plugin que nous avons développé. Par la suite nous ne donnons pas les détails de conception. Ils sont consultables dans le cahier des charges fonctionnel. Les choix de représentation ont cependant été mis en annexe D.

Fonctionnalités

- Langages étudiables : Java
- Sélection libre du type des éléments : packages, classes, méthodes, attributs
- Informations additionnelles exportées : nombre de lignes de code, JavaDoc, cardinalité minimale des relations entre éléments
- Format de fichier produit : [GDF](#)
- Langue de l'interface : English

Architecture globale

Architecture Overview

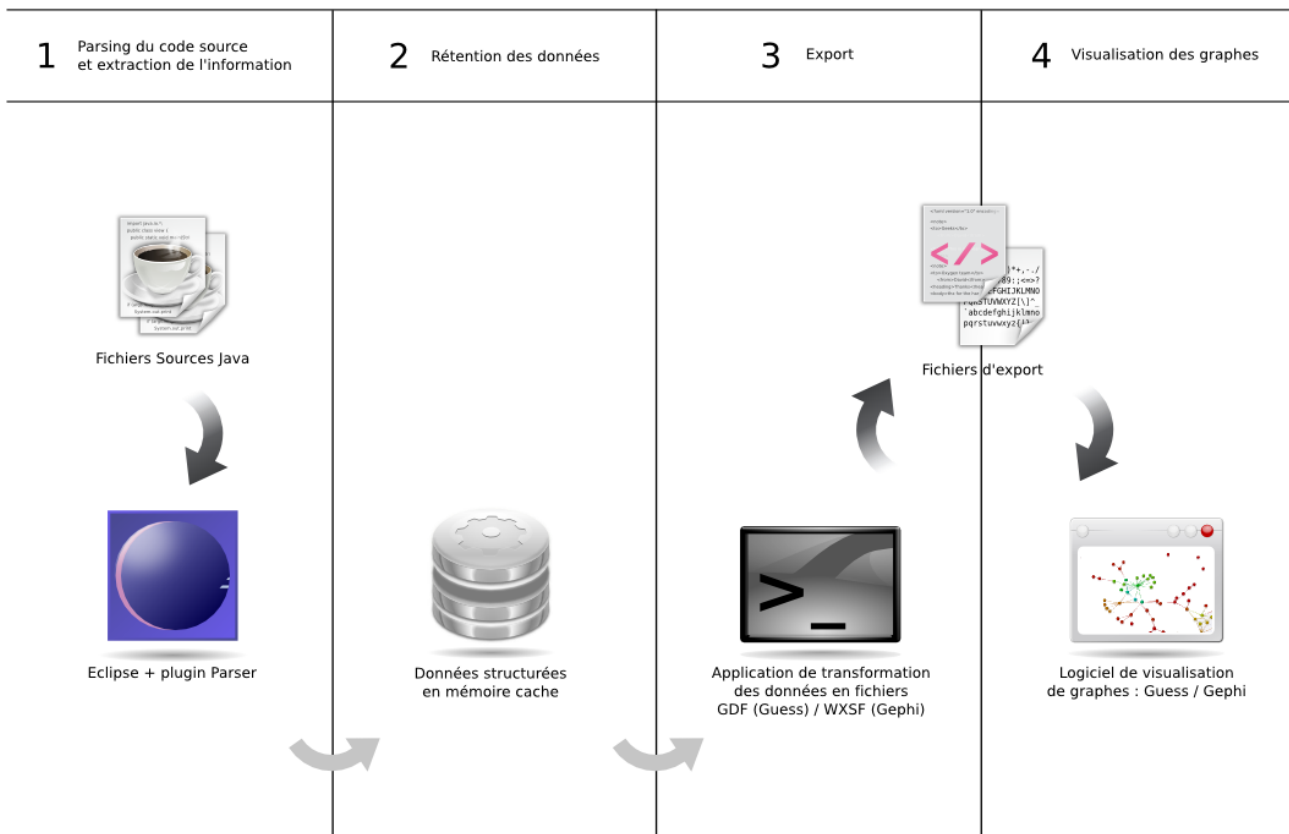
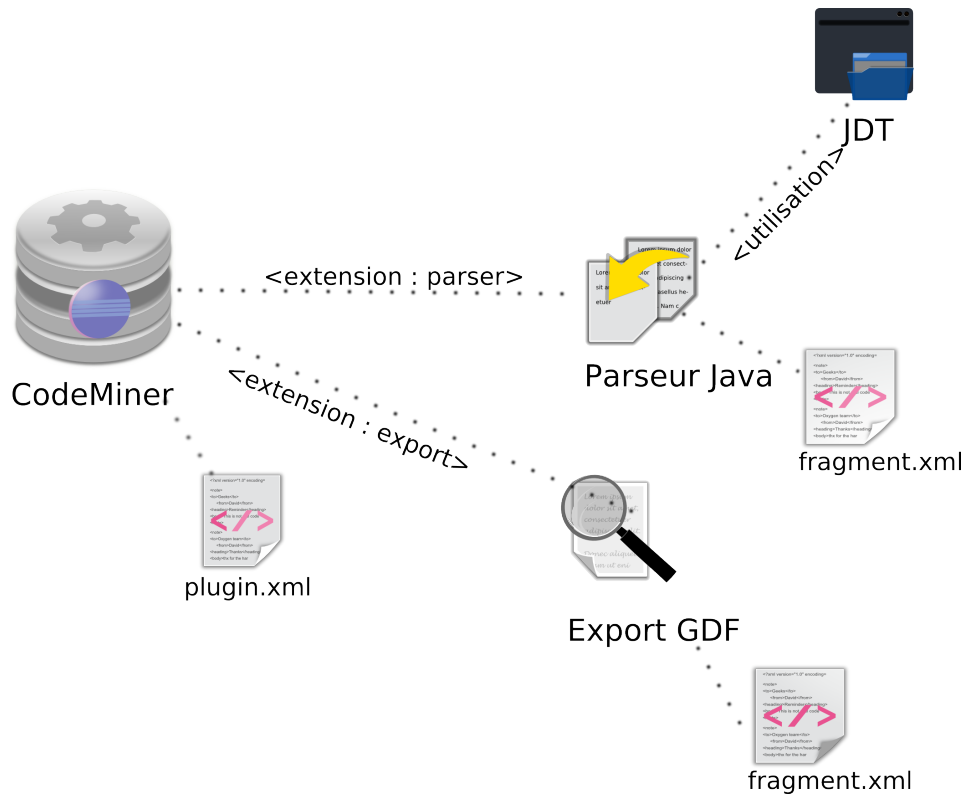


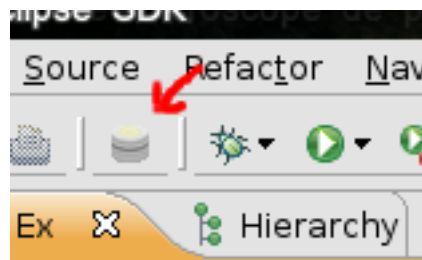
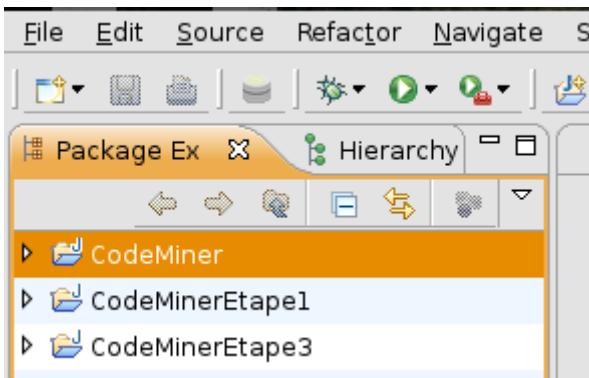
Illustration 1: Vue globale sur l'architecture du système

Plugin Eclipse

Un plugin Eclipse CodeMiner et des fragments de plugin permettent d'ajouter les fonctionnalités d'analyse de programme et d'export au plugin mère. Les fragments étendent les extensions *parser* ou *exporter*. Un même plugin peut définir plusieurs extensions, ainsi un plugin peut exporter plusieurs formats différents s'il étend plusieurs fois le point d'extension *exporter* (cf. illustration suivante).



L'utilisateur règle les paramètres, clic sur un projet puis démarre l'extraction des données à l'aide du bouton de démarrage :



Les paramètres sont réglables dans les préférences du plugin, accessibles au même endroit que

l'ensemble des préférences de Eclipse (menu Window > Preferences...) :



Illustration 5: Paramètres généraux

Export directory : indique le répertoire d'écriture des fichiers produits par le plugin, par défaut dans le répertoire *bin* du projet choisi.

Format : le format d'export des données est uniquement le GDF pour le moment.

Write pre-export files : enregistre les données sérialisées du parseur, avant la construction du fichier formaté. Utile en cas de problème (si vous nous rapportez un bug on vous demandera certainement le fichier .text généré), à décocher pour améliorer les performances sinon.

Debug mode : affiche des messages dans la console, à décocher pour améliorer sensiblement les performances.

L'utilisateur peut choisir les types d'éléments à exporter (sans contrainte) :

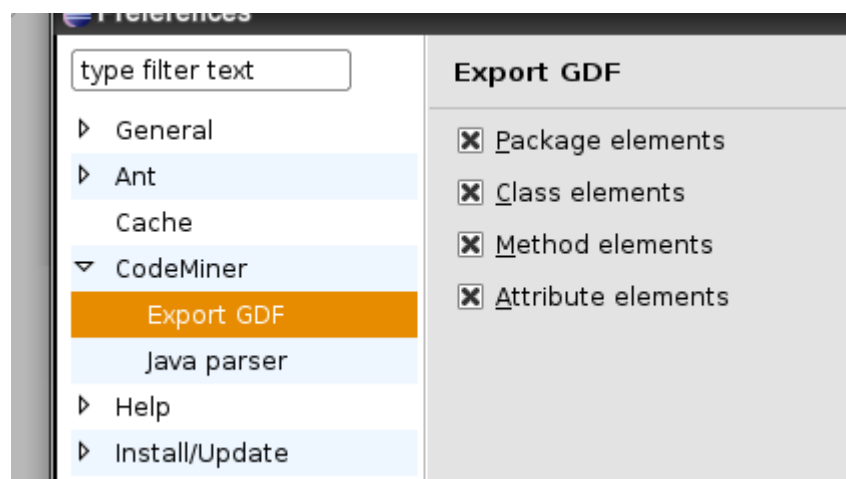


Illustration 6: Paramètres d'export GDF des éléments

Enfin il peut choisir d'intégrer les bibliothèques JAR lors du parsing (exclus par défaut), mais seuls les packages et classes seront détectés :



Illustration 7: Paramètres du parseur Java

Plugin Géphi

Ce plugin adapte la représentation graphique des nœuds du graphe (initialement des boules) en panneaux plus lisibles, et ajoute une interface d'affichage des méta-données des éléments ainsi qu'une fonction de recherche et de parcours des résultats dans le graphe.

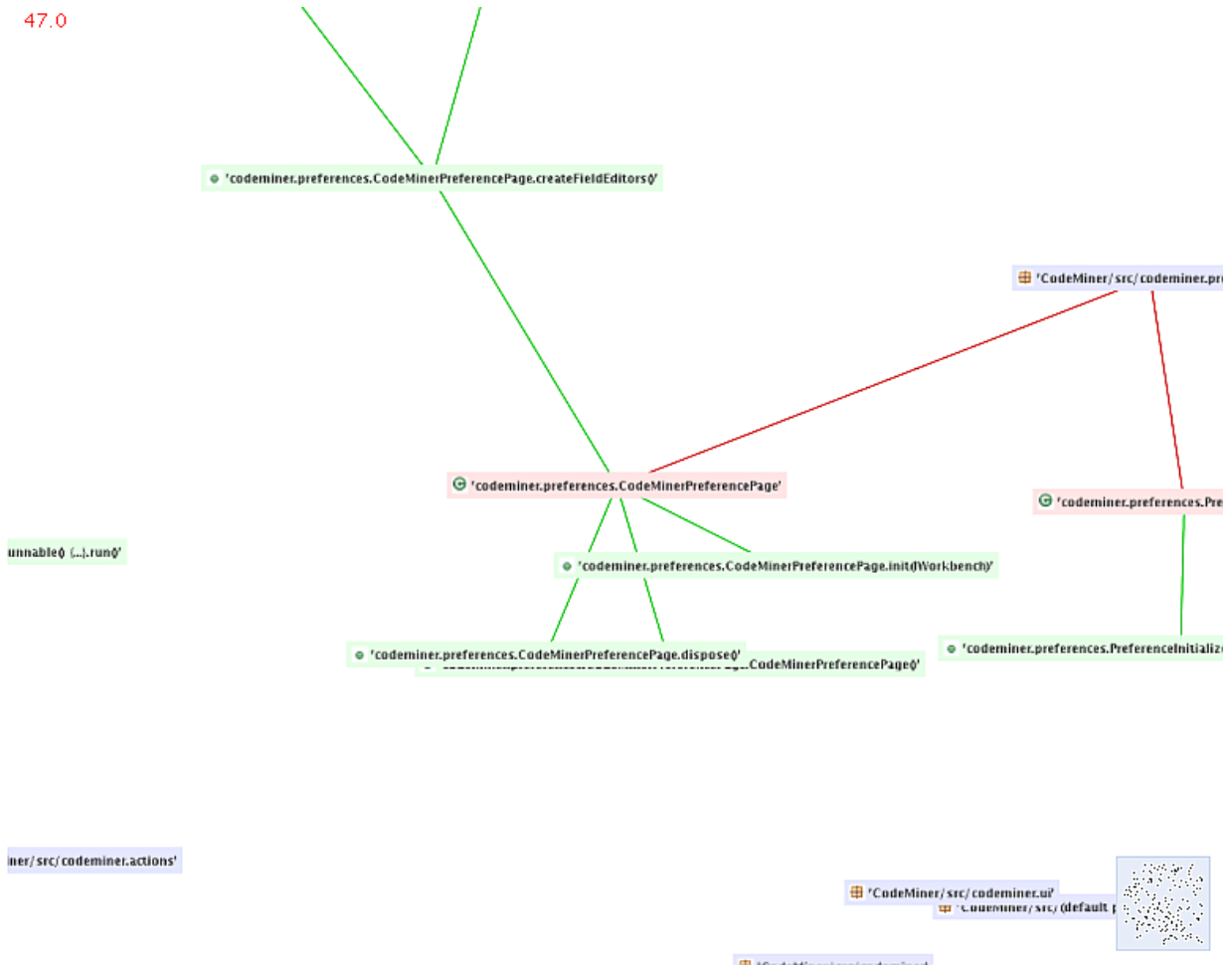


Illustration 8: Représentation du graphe

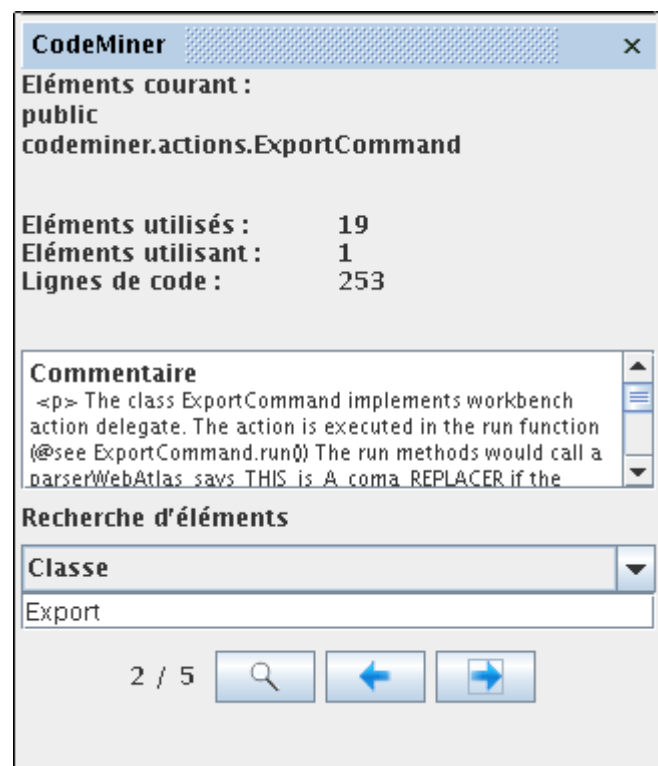
Les couleurs sont fixés par le plugin et dépendent du type de l'élément : bleu pour les package, rouge pour les classes, vert pour les méthodes et gris pour les attributs. A chaque type d'élément est attribué une icône, qui se précise pour les méthodes et attributs en fonction du *modifier* (public/protected/private, static, final). Ces icônes sont celles utilisées par Eclipse lors de la représentation des arbres AST. Grâce à Géphi, ces graphes ne sont pas seulement visualisables, mais aussi manipulables : l'utilisateur peut déplacer des groupes de nœuds, zoomer, cacher des éléments...et choisir et modifier à la volée les paramètres des algorithmes de spatialisation dont il se sert pour associer des coordonnées planaires aux nœuds du graphe. Ces choix sont issus des recherches en sciences cognitives (paradigme é actif), que nous ne détaillerons pas.

Un clic sur un élément du graphe fait afficher ses méta-données dans ce panneau :

- Son *modifier* et son nom complet depuis le package.
- Quelques statistiques sont données, ainsi que la JavaDoc de l'élément si elle est présente.

La recherche d'élément peut se faire sur son type (package, classe, méthode, attribut, ou tous), et la chaîne de caractère de recherche est évaluée en tant qu'expression rationnelle.

Le nombre de résultats est affiché et le parcours des éléments avec les boutons Précédent et Suivant fait positionner la caméra du graphe sur l'élément en question.



2. Méthodologie

Le projet a été réalisé en deux phases : une phase de conception globale, puis une phase de développement en cycle itératif. Cette dernière consiste dans notre cas à mettre en place une fonctionnalité à la fois, la tester et la valider, avant de commencer une autre fonctionnalité. L'avancement du projet est ainsi maîtrisé et le logiciel exploitable à chaque fin de cycle.

Le cœur de l'extraction des données d'un projet Java (plugin Eclipse) a fait l'objet de tests unitaires pour certifier le bon fonctionnement de cette partie pivot du système. Ce sont des tests de bon déroulement d'extraction de l'information sur des cas simples, comme l'existence d'héritage entre deux classes ou encore l'appel d'une méthode.

Ces données ont d'ailleurs fait l'objet d'une normalisation en troisième forme normale (3FN), méthode (issu des SGBD) de réduction des données pour éviter les redondances tout en conservant des performances d'accès optimales.

Des algorithmes ont été créés pour les points clés de l'application : l'extraction des données et leur transformation en fichiers formatés. L'objectif était de maîtriser l'encombrement et de diminuer la durée de ces tâches, où toutes les données sont créées et manipulées.

La conception de l'application a reposé en majeure partie sur la méthodologie UML avec l'usage de diagrammes de composants et de diagrammes de classes.

Enfin la totalité du plugin Eclipse est documenté et sa JavaDoc est accessible en ligne.

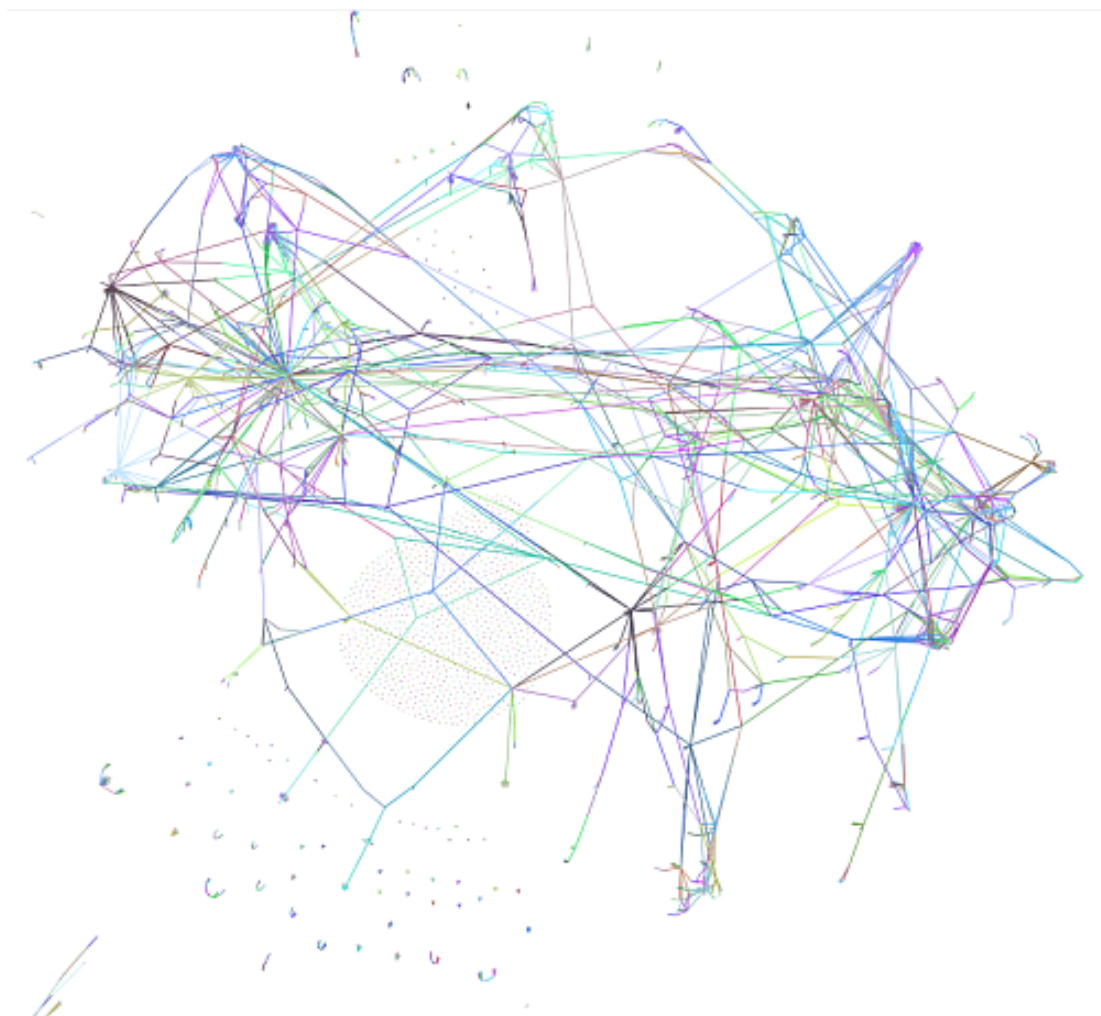
3. Environnement et outils

Le développement a été réalisé sous Windows XP et Linux Ubuntu 7.10, système d'exploitation basé sur Debian, avec le logiciel Eclipse 3.3, exécuté sur des machines virtuelles Java 1.6. Le plugin Géphi est développé pour Géphi nightly build en révision n°253 du SVN (première version stable supportée : 0.5.b5). L'application est publiée sur la plateforme collaborative de technologies Web-mining.fr.

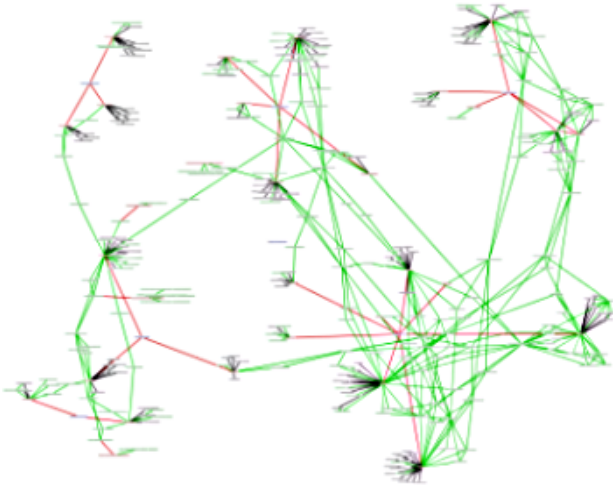
4. Retour sur les premiers usages

CodeMiner a été testé par deux utécéens ayant déjà des connaissances en théorie des graphes et science des réseaux, et ont été formé dans l'UV IC05 de Franck Ghitalla. Nous avons commencé par évaluer le logiciel avec ce type de public car nous savions que l'appropriation serait aisée.

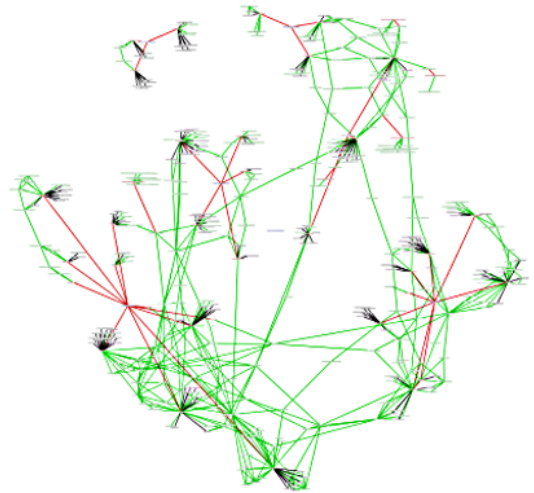
Ainsi Mathieu Bastian, responsable de Géphi, nous a aidé dans un premier temps à valider les graphes produits représentant la structure de Géphi. Il s'en est ensuite servi pour valider des éléments de son architecture, et s'orienter dans ce projet de plus de 20 000 lignes de code.



Julian Bilcke, quant à lui, a pu voir et manipuler deux graphes de son projet d'NF28 PhotoViz. Un peu après le démarrage du développement et la mise en place du SVN, et un autre trois semaines plus tard. Il a ainsi pu détecter du « code mort », composantes détachées de la composante principal (cf. Annexe C) et vérifier l'intégration des parties de ses binômes. Il a été « surpris » la première fois par la complexité de son projet, étant « *habitué à raisonner classe par classe, ou méthode par méthode (pendant la rédaction du code)* ». Il espère pouvoir s'en servir à l'avenir pour surveiller l'évolution de l'architecture de ses projets.



*Illustration 12: PhotoViz à t=début de développement
« chacun travail dans son coin »*



*Illustration 11: PhotoViz à t + 3 semaines
« Intégration des parties »*

Enfin CodeMiner a aussi servi à analyser...le plugin Eclipse de CodeMiner en cours de développement ! Ces images ont été prises avant la création du plugin Géphi pour adapter la visualisation.

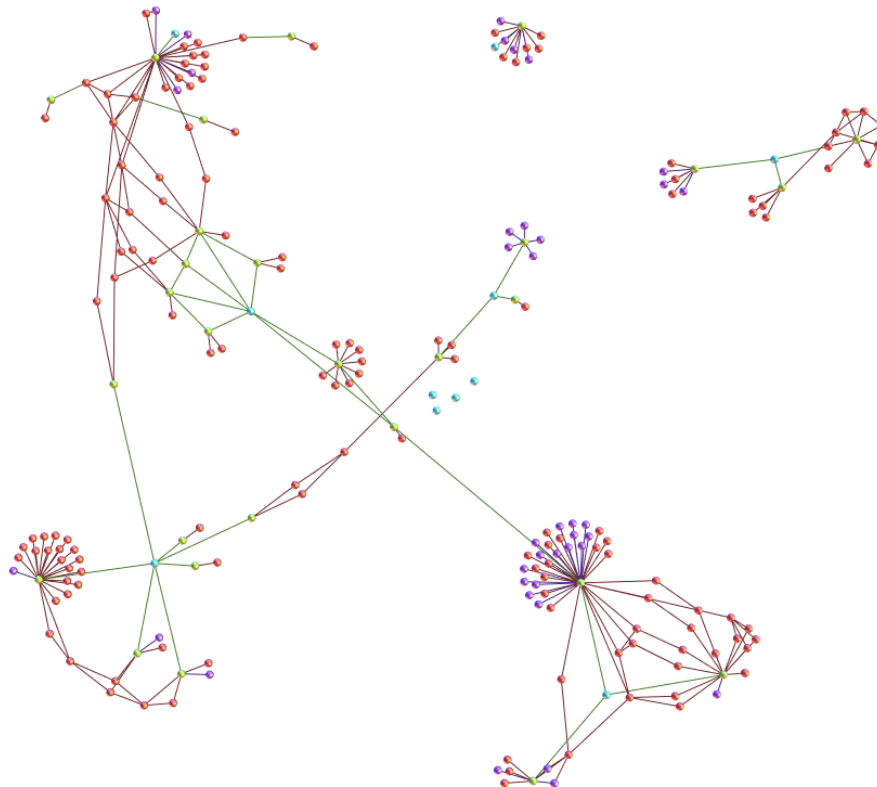


Illustration 13: Fragment principal du plugin Eclipse de CodeMiner, vu avec Géphi standard

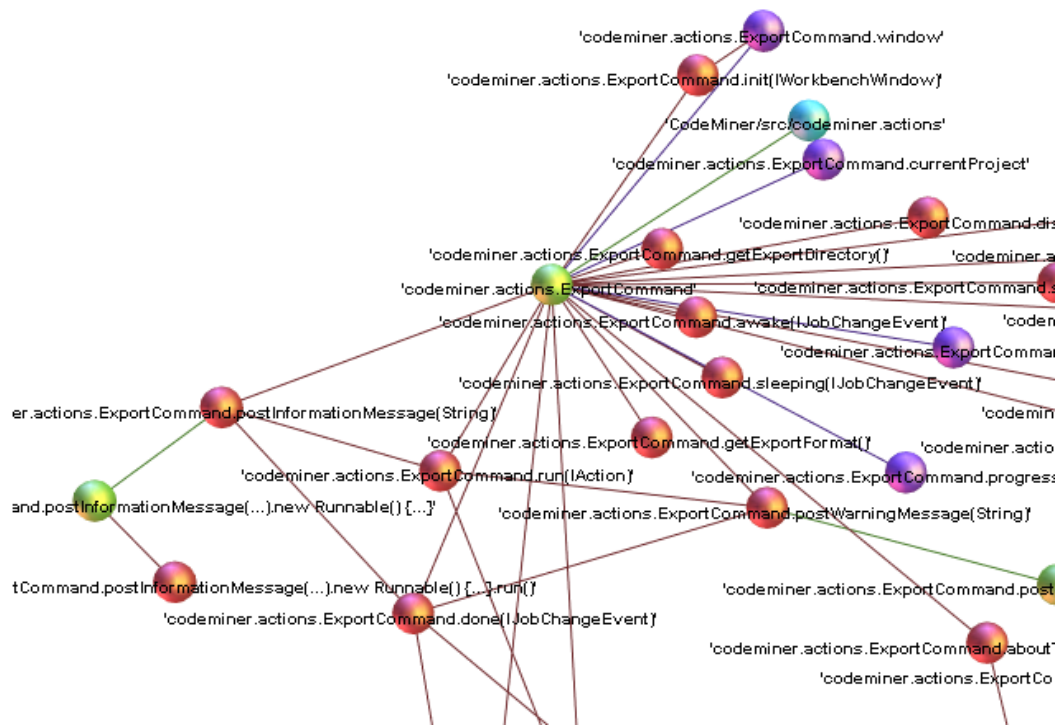


Illustration 14: Détail de la vue du plugin Eclipse de CodeMiner

III. Gestion du projet

1. Organisation

Nous avons des compétences complémentaires, ce qui nous a amené naturellement à la répartition des rôles suivante :

- Jérémy Palmier s'est essentiellement préoccupé de la couche extraction des données : parseur, architecture logicielle et algorithmes.
- Sébastien Heymann s'est principalement chargé de la gestion de projet, de la gestion de la conception, de la communication (interne et externe, écrite et orale) et de la couche de visualisation.

Toutes les étapes de conception ont fait l'objet de réunions et de validations en groupe. Ces réunions ont eu lieu une fois par semaines. D'autres réunions faites avec les équipes de TX IceCrawler et SIGL:CPAN ont servi à l'entretien de la motivation et au pilotage global de chaque projet, environ une fois toutes les trois semaines.

Enfin des documents ont été produit à intervalles réguliers : avant-projet, cadrage théorique, cahier des charges fonctionnel, notes de clarification et ce présent rapport.

2. Planning

Le planning du projet a été globalement respecté.
Le développement a été divisé en 4 parties :

1. Parsing de projet Java
2. Rétenion des données
3. Transformation des données en fichiers formatés
4. Visualisation des données

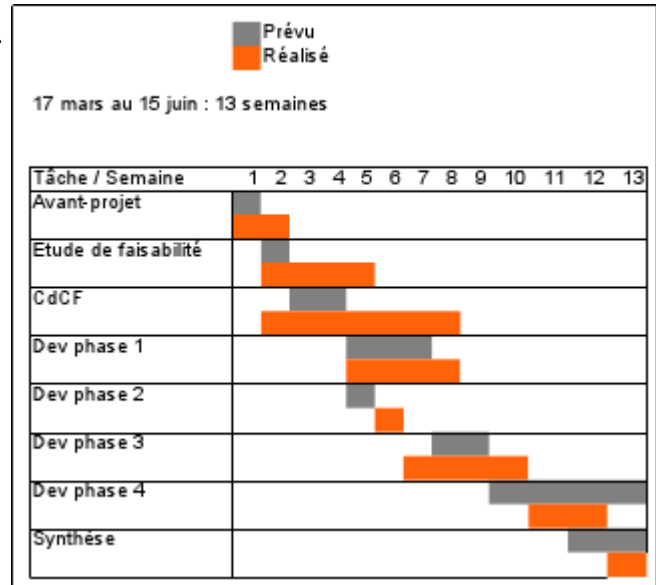


Illustration 15: Diagramme de GANTT simplifié

IV. Conclusion

1. Rappel des points essentiels

CodeMiner est un projet de chaîne de traitement logiciel visant à représenter un code source sous forme de graphe (nœuds/arcs) dans un espace 2D. Il a pour but de donner à voir les relations entre packages, classes, méthodes et variables existantes dans un code source Java.

Il se base sur l'environnement de développement Eclipse pour son plugin d'extraction de données, et sur l'application de visualisation et de manipulation de graphe Géphi. Il est robuste et extensible (100% modulaire).

L'ensemble du projet est publié sous licence libre GNU/GPL 3 publié sur www.web-mining.fr/codeminer. Il fait déjà l'objet d'une utilisation, ce qui nous amène à affirmer qu'il a atteint la règle des « 3U » : Utile, Utilisable, Utilisé !

2. Bilan

Ce projet était un pari risqué : nous devions réaliser une application innovante, de la conception à la publication, en passant par la recherche de solutions et l'évaluation de technologies. C'est ainsi l'ensemble du processus d'élaboration d'un nouveau produit qu'il nous a fallu maîtriser pour notre usage, et ainsi toucher aux multiples compétences d'un ingénieur en Recherche & Développement, le tout en moins de 3 mois. Ce pari a largement été remporté grâce à une bonne rigueur, beaucoup de communication interne, une grande régularité et l'entre-aide des autres projets TX, IceCrawler et SIGL:CPAN.

Nous avons rencontré des difficultés particulières en trois points. Le premier est la recherche d'informations fiables sur le parseur JDT, utilisé par Eclipse. Cette étape a demandé plusieurs jours de collecte de documents, avant même de choisir entre lui et le générateur

d'analyseur de langage AntLR. Le deuxième point est l'état du logiciel Géphi, toujours en alpha, qui ne possède donc pas l'ensemble de ses fonctionnalités et n'est pas encore complètement stable. Ces problèmes furent surmontés grâce à la disponibilité de son responsable, Mathieu Bastian, qui a pallié en direct à nos besoins. Enfin l'évaluation de solutions technologiques et les choix cruciaux à effectuer ont nécessité beaucoup de travail.

3. Perspectives du projet CodeMiner

CodeMiner est un projet bien né. Capable de créer des usages répondant à des besoins non spécifiés à l'origine, il trouve son utilité dans la manière dont les gens se l'approprient. L'avenir du projet passera désormais par l'amélioration de la couche visualisation (ex: l'affichage de numérotation de séquences d'instructions), l'ajout d'autres langages de programmation comme le C++, l'ajout de fonctionnalités de statistiques et recherche opérationnelle à l'instar de la recherche de circuits, et la gestion de la comparaison de graphes évoluant dans le temps par une analyse de dépôts SVN. L'emploi d'un profiler pour animer la visualisation d'un graphe d'appels de méthode en temps réel est aussi un rêve que nous caressons. La publication du module de visualisation se fera lors de la sortie officielle de Géphi.

V. Annexes

1. *Glossaire*

Arbre AST (Abstract Syntax Tree) :

Représentation intermédiaire du code source utilisé par un analyseur syntaxique.

Eclipse :

Environnement de développement intégré. Supporte le Java, C++, PHP, Flex, etc.

Géphi :

Logiciel de visualisation et de manipulation de graphes (nœuds/arcs) dans un espace plan ou 3D.

JDT :

Parseur de langage utilisé par Eclipse.

Parseur :

Analyseur de texte (parcours et découpe en unités d'information).

SGBD :

Système de Gestion de Base de Données.

SVN (Subversion) :

Serveur de dépôt de code source et de contrôle de version.

UML (Unified Modeling Language) :

Langage graphique de modélisation des données et des traitements.

2. Annexe B : Contrat de licence GNU/GPL 3

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will

be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

3. Annexe C : Témoignage d'usage

« Avant d'utiliser codeminer, je n'avais aucune idée de l'usage que je pouvais en faire, n'ayant jamais utilisé d'outil de ce genre, ni d'outil de reverse engineering donc j'ai été surpris la première fois, ça m'a fait un peu peur même ;), je me suis demandé comment on (moi et les autres de mon équipe) arrivait à gérer un projet aussi complexe, vu le grand nombre de connexions, entre nos différentes parties..

De visualiser tous les connexions du code, ça surprend un peu donc au début, quand on est habitué à raisonner classe par classe, ou méthode par méthode (pendant la rédaction du code). J'aurais aimé te dire que ça m'a permis de voir les connexions "cachées", (qui n'existent officiellement dans notre UML, mais en pratique existent réellement : hacks, bug fixes, retouches de dernière minutes sans avoir eu le temps de mettre à jour l'UML etc..) mais... on a pas d'UML :D

Paradoxalement, ça m'a donc servi : puisque nous n'avons pas d'UML, nous n'avons pas de "référence" qui fait foi pour l'architecture, et on fait donc à l'arrache, en modifiant l'architecture en même temps qu'on ajoute les fonctionnalités et qu'on corrige les bugs. Mais avec codeminer j'ai pu suivre (bon, sur 2 "séances" :P) l'évolution de l'architecture et faire au moins une correction (suppression de code mort)

J'aurais bien aimé m'en servir comme référence graphique pour faire du code bien structuré (beau ? enfin.. faisons attention aux raccourcis un peu trop tentants ^^ les meilleurs design patterns ne rendent peut-être pas très bien sous forme de graphe :P) et voir si ça me permettait vraiment de déceler les problèmes architecturaux ("est-ce normal qu'il y ai bcp de connexion ici ? est-ce qu'on avait le choix ?" etc), mais j'ai pas eu le temps, dommage. »

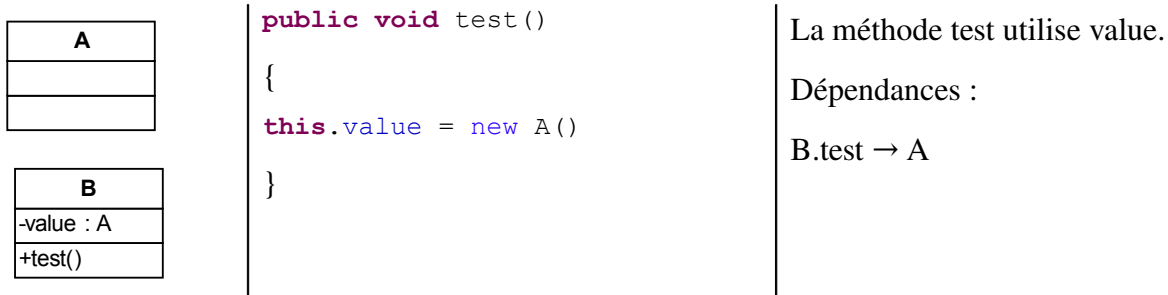
Julian Bilcke, à propos de son projet d'NF28, texte recueilli par email en réponse à la question « *En quelques mots, comment vous êtes-vous servi de codeminer (quels usages, pour voir quoi, pour décider quoi) et quelle a été votre réaction la première fois que vous avez vu la représentation de votre code ?* ».

4. Annexe D : Liste des relations entre éléments représentées

Les dépendances concernant les variables

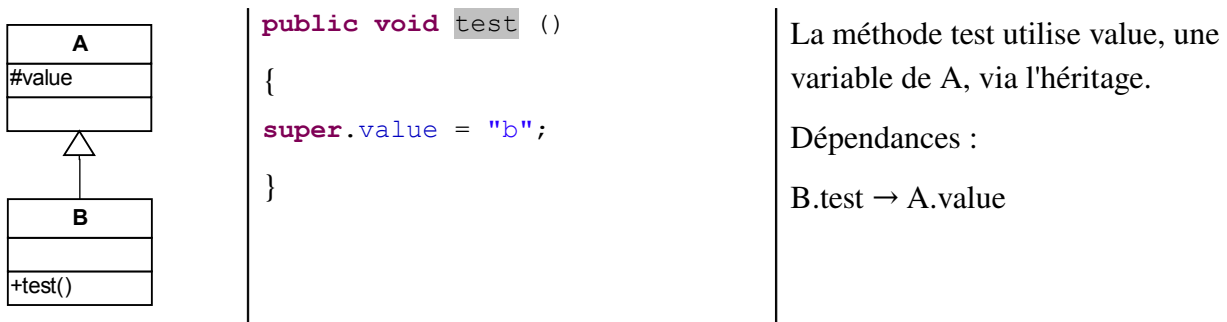
public boolean visit(FieldAccess node)

FieldAccess permet de représenter des expressions comme "this.foo"



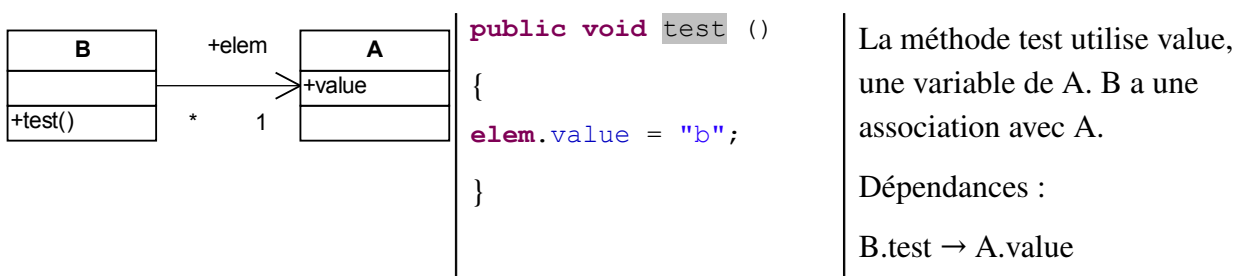
public boolean visit(SuperFieldAccess node)

SuperFieldAccess permet de représenter des expressions comme "super.foo",



public boolean visit(QualifiedName node)

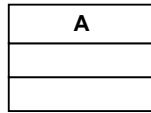
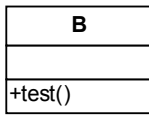
QualifiedName permet de représenter des expressions comme "foo.bar",



Les dépendances concernant les Méthodes

public boolean visit(ClassInstanceCreation node)

ClassInstanceCreation permet de représenter les dépendances qui résultent de la création d'instances.



```

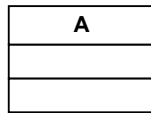
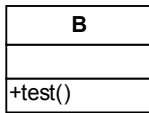
public void test ()
{
  new A ();
}
  
```

La méthode test crée une instance de A.

Dépendances :

B.test → A

L'exemple suivant montre une spécificité de Java qu'il faut prendre en compte pour les constructeurs par défaut. Ici, le constructeur de A n'est pas défini, et donc n'est pas connu à l'analyse. Nous choisissons alors de forcer la création de la dépendance de la méthode vers le type créé.



```

public void test ()
{
  new A ();
}
  
```

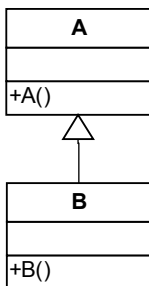
La méthode test crée une instance de A, mais A n'a pas de constructeur défini.

Dépendances :

B.test() → A

public boolean visit(SuperConstructorInvocation node)

SuperConstructorInvocation est une dépendance qui résulte de l'utilisation du constructeur avec super.



```

public B ()
{
  super ();
}
  
```

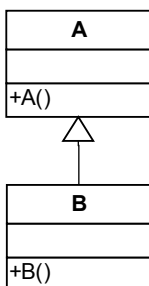
Le constructeur de A fait appel au constructeur de B.

Dépendances :

B.B ()→A. A()

public boolean visit(SuperMethodInvocation node)

SuperMethodInvocation est utilisée pour les appels de méthode utilisant le mot-clé super.



```

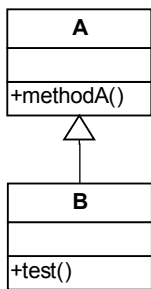
public B ()
{
  super ();
}
  
```

Le constructeur de A fait appel au constructeur de B.

Dépendances :

B.B ()→A. A()

public boolean visit(SuperMethodInvocation node)



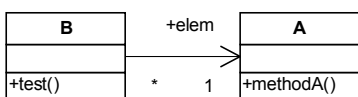
```

public void test()
{
    super.methodA();
}
  
```

Le constructeur de A fait appel au constructeur de B.
 Dépendances :
 B.test → A.methodA

Le dernier cas concerne les appels de méthodes usuelles sur les appels de fonction en général sans les cas particuliers présentés précédemment. On trouve les appels aux fonctions statiques, les appels de méthodes via variable membre ou à l'intérieur d'une même classe ainsi que les appels récursif.

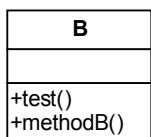
public boolean visit(MethodInvocation node)



```

public void test()
{
    elem.methodA();
}
  
```

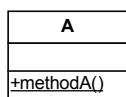
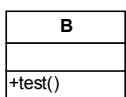
La fonction test appelle la fonction methodA de elem.
 Dépendances :
 B.test → A.methodA



```

public void test()
{
    methodB();
}
  
```

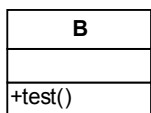
La fonction test de A appelle la methodB de A
 Dépendances :
 B.test → B.methodA
 Ou rien ?



```

public void test()
{
    A.methodA();
}
  
```

La fonction test appelle la fonction static methodA de A
 Dépendances :
 B.test() → A.methodA()



```

public void test()
{
    test();
}
  
```

La fonction test de B effectue un appel récursif sur elle-même. Aucune dépendance n'est ajoutée car un élément est toujours dépendant de lui-même.